Documentation: AM2302 Sensor Integration with Flask-Based Water Cooling System

Overview

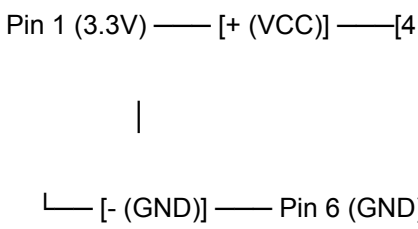
The project enhances a Raspberry Pi-based water cooling system to monitor temperature and humidity using an AM2302 (DHT22) sensor. The system uses a Flask web app (`relay_webapp.py`) to:

- Control a relay on GPIO 16 for 120 seconds to activate the cooling system.
- Display sensor data and relay status on a webpage (`index.html`) with bilingual labels (English/Amharic).
- Update sensor readings every 5 seconds via AJAX without disrupting the relay countdown.
- Log relay and sensor events to a Logstash server (192.168.6.100:5959).

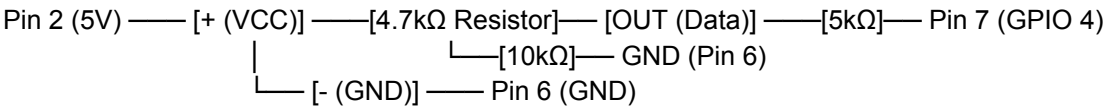
Hardware Setup

- **Components**:
  - Raspberry Pi (e.g., Pi 4 or 5, running Raspberry Pi OS, Python 3.11).
  - AM2302 (DHT22) sensor (3-pin: +, -, OUT).
  - Relay module on GPIO 16.
  - 4.7kΩ pull-up resistor (non-polarized).
  - Optional: 0.1µF capacitor for noise filtering, 5kΩ + 10kΩ resistors for 5V voltage divider (if needed).

- **Wiring**:
    - **+ (VCC)**: Connect to Pin 1 (3.3V) on Raspberry Pi.
    - **- (GND)**: Connect to Pin 6 (GND).
    - **OUT (Data)**: Connect to Pin 7 (GPIO 4).
    - **Pull-up Resistor**: 4.7kΩ between + (VCC) and OUT (Data). Either resistor end can connect to either point (no polarity).
    - **Diagram**:



- **Optional (for long cables >1m)**: Use Pin 2 (5V) for VCC with a voltage divider (5kΩ + 10kΩ) on Data to step down to 3.3V:



Software Setup

- **Dependencies**:
    - **System Packages**:
    - `bash`

```
sudo apt update
```

    - sudo apt install -y build-essential python3-dev git python3-pip python3-lgpio
    - **Virtual Environment** (due to PEP 668 restrictions):
    - `bash`

```
cd /home/cooling-automation/qrb/rpi
python3 -m venv venv
source venv/bin/activate
```

    - pip install flask python-logstash adafruit-circuitpython-dht
    - **Why Virtual Environment?**: Raspberry Pi OS (e.g., Bookworm, 64-bit) enforces PEP 668, preventing system-wide `pip` installs. A virtual environment (`venv`) isolates dependencies.
- **Verify Libraries**:
- `bash`

source venv/bin/activate

- pip list
  Ensure `flask`, `python-logstash`, and `adafruit-circuitpython-dht` are listed.

Code Implementation

Python Code: relay_webapp.py

Located at `/home/cooling-automation/qrb/rpi/relay_webapp.py`, this Flask app controls the relay, reads the AM2302 sensor, and serves the webpage.

```
python
# -*- coding: utf-8 -*-
from flask import Flask, render_template, Response, jsonify
import RPi.GPIO as GPIO
import time
import threading
from logstash import TCPLogstashHandler
import logging
import adafruit_dht
import board

# Setup logging
mylogger = logging.getLogger(__name__)
handler = TCPLogstashHandler(host='192.168.6.100', port=5959)
mylogger.addHandler(handler)

app = Flask(__name__)

# GPIO Setup for relay
RELAY_PIN = 16
GPIO.setmode(GPIO.BCM)
GPIO.setup(RELAY_PIN, GPIO.OUT)
GPIO.output(RELAY_PIN, GPIO.LOW)  # Relay off initially

# Sensor setup (AM2302 = DHT22, on GPIO 4)
DHT_SENSOR = adafruit_dht.DHT22(board.D4, use_pulseio=False)
DHT_PIN = 4

# Global variable
relay_busy = False

def read_sensor():
    """Read temperature and humidity from AM2302 with retries."""
    for i in range(3):  # Retry up to 3 times
        try:
            temperature = DHT_SENSOR.temperature
```

```
            humidity = DHT_SENSOR.humidity
            if humidity is not None and temperature is not None:
                mylogger.info(f"Sensor reading: Temp={temperature:.1f}°C, Humidity={humidity:.1f}%")
                return round(temperature, 1), round(humidity, 1)
            time.sleep(2)
        except RuntimeError as e:
            mylogger.error(f"Sensor read error (attempt {i+1}): {e}")
            time.sleep(2)
    mylogger.error("Failed to read AM2302 sensor after retries")
    return None, None

def toggle_relay():
    """Run relay for 120 seconds."""
    global relay_busy
    relay_busy = True
    try:
        mylogger.warning("Turning ON")
        print("Turning ON normally-off outlets")
        GPIO.output(RELAY_PIN, GPIO.HIGH)  # Turn on relay
        time.sleep(120)  # Keep on for 120 seconds
        mylogger.warning("Turning OFF")
        print("Turning OFF normally-off outlets")
        GPIO.output(RELAY_PIN, GPIO.LOW)  # Turn off relay
    finally:
        relay_busy = False

@app.route('/')
def index():
    temp, hum = read_sensor()
    return render_template('index.html', current_temp=temp, current_hum=hum)

@app.route('/trigger')
def trigger():
    global relay_busy
    if not relay_busy:
        threading.Thread(target=toggle_relay, daemon=True).start()
        return Response("Water is running for 120 seconds!", status=200)
    return Response("Relay is busy, please wait!", status=429)

@app.route('/get_status')
def get_status():
    temp, hum = read_sensor()
    return jsonify({
        'temp': temp,
        'humidity': hum,
        'relay_busy': relay_busy
    })

if __name__ == '__main__':
```

```
try:
    app.run(host='0.0.0.0', port=5000, debug=False)
except KeyboardInterrupt:
    print("\nProgram terminated by user")
    GPIO.cleanup()
```

Key Features:

- Uses `adafruit-circuitpython-dht` for AM2302 on GPIO 4.
- Reads temperature and humidity with retries to handle timing issues.
- Logs sensor readings and relay events to Logstash.
- Serves `/` for the webpage, `/trigger` for relay control, and `/get_status` for AJAX updates.
- UTF-8 encoding (`# -*- coding: utf-8 -*-`) supports special characters (e.g., °C).

HTML Code: index.html

Located at `/home/cooling-automation/qrb/rpi/templates/index.html`, this displays the interface with bilingual labels and AJAX updates.

`html`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Water Cooling System</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            margin: 0;
            background-color: #f0f0f0;
            padding: 10px;
        }
        .container {
            text-align: center;
            background-color: white;
            padding: 2rem;
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
            width: 90%;
            max-width: 500px;
            margin: auto;
        }
        h1 {
            color: #333;
```

```
    margin-bottom: 1.5rem;
    font-size: 2rem;
}
.status {
    font-size: 1.5rem;
    font-weight: bold;
    margin: 1rem 0;
}
.status.on {
    color: #4CAF50;
}
.status.off {
    color: #e74c3c;
}
.countdown {
    font-size: 2rem;
    color: #333;
    margin: 0.5rem 0;
}
.countdown-circle {
    width: 100px;
    height: 100px;
    margin: 0 auto;
    position: relative;
}
.countdown-circle svg {
    transform: rotate(-90deg);
}
.countdown-circle circle {
    fill: none;
    stroke: #4CAF50;
    stroke-width: 10;
    stroke-dasharray: 283;
    stroke-dashoffset: 0;
    transition: stroke-dashoffset 120s linear;
}
.countdown-circle.running circle {
    stroke-dashoffset: 283;
}
button {
    padding: 0.75rem 1.5rem;
    font-size: 1.125rem;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    margin-top: 1rem;
    width: 100%;
```

```
    max-width: 200px;
}
button:hover {
    background-color: #45a049;
}
button:disabled {
    background-color: #cccccc;
    cursor: not-allowed;
}
.sensor {
    font-size: 1.25rem;
    color: #333;
    margin: 1rem 0;
    padding: 0.5rem;
    border: 1px solid #ddd;
    border-radius: 5px;
    background-color: #f9f9f9;
}
footer {
    margin-top: 2rem;
    font-size: 0.875rem;
    color: #666;
    text-align: center;
}
@media (max-width: 600px) {
    .container {
        padding: 1rem;
    }
    h1 {
        font-size: 1.5rem;
    }
    .status {
        font-size: 1.25rem;
    }
    .countdown {
        font-size: 1.75rem;
    }
    .countdown-circle {
        width: 80px;
        height: 80px;
    }
    .countdown-circle svg {
        width: 80px;
        height: 80px;
    }
    .countdown-circle circle {
        cx: 40;
        cy: 40;
        r: 35;
```

```
        }
        button {
            padding: 0.5rem 1rem;
            font-size: 1rem;
            max-width: 150px;
        }
        .sensor {
            font-size: 1rem;
        }
        footer {
            font-size: 0.75rem;
        }
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Water Cooling system - በውሀ የማቀዝቀዝ ስርዓት</h1>
    <div class="status off" id="status">Powered Off - ዉሀው መፍሰስ አቁሟል</div>
    <div class="sensor">
      Temperature - ሙቀት: <span id="current-temp">{{ '%.1f' % current_temp if current_temp is not
none else 'N/A' }}°C</span><br>
      Humidity - እርጥበት: <span id="current-hum">{{ '%.1f' % current_hum if current_hum is not none
else 'N/A' }}%</span>
    </div>
    <div class="countdown" id="countdown" style="display: none;">120</div>
    <div class="countdown-circle" id="circle" style="display: none;">
      <svg width="100" height="100">
        <circle cx="50" cy="50" r="45" />
      </svg>
    </div>
    <button id="triggerButton" onclick="triggerRelay()">Power On - ማቀዝቀዝ ጀምር (120s)</button>
    <p><em><small>2025 © QRB Labs</small></em></p>
  </div>
  <footer></footer>

  <script>
    let countdownInterval = null;

    async function triggerRelay() {
      const button = document.getElementById('triggerButton');
      const status = document.getElementById('status');
      const countdown = document.getElementById('countdown');
      const circle = document.getElementById('circle');

      button.disabled = true;
      status.textContent = 'Powered On - ዉሀው እየፈሰሰ ነዉ';
      status.className = 'status on';
      countdown.textContent = '120';
```

```
            countdown.style.display = 'block';
            circle.style.display = 'block';
            circle.classList.add('running');

            try {
                const response = await fetch('/trigger');
                const message = await response.text();

                if (response.status === 200) {
                    let seconds = 120;
                    countdownInterval = setInterval(() => {
                        seconds--;
                        countdown.textContent = seconds;
                        if (seconds <= 0) {
                            clearInterval(countdownInterval);
                            status.textContent = 'Powered Off - ዉሀው መፍሰስ አቁሟል';
                            status.className = 'status off';
                            countdown.style.display = 'none';
                            circle.style.display = 'none';
                            circle.classList.remove('running');
                            button.disabled = false;
                        }
                    }, 1000);
                } else {
                    status.textContent = message;
                    status.className = 'status off';
                    countdown.style.display = 'none';
                    circle.style.display = 'none';
                    circle.classList.remove('running');
                    button.disabled = false;
                }
            } catch (error) {
                status.textContent = 'Error: ' + error.message;
                status.className = 'status off';
                countdown.style.display = 'none';
                circle.style.display = 'none';
                circle.classList.remove('running');
                button.disabled = false;
            }
        }

        // Update sensor readings every 5 seconds
        function updateSensor() {
            fetch('/get_status')
                .then(response => response.json())
                .then(data => {
                    document.getElementById('current-temp').textContent = data.temp !== null ?
data.temp.toFixed(1) + '°C' : 'N/A';
```

```
                document.getElementById('current-hum').textContent = data.humidity !== null ?
data.humidity.toFixed(1) + '%' : 'N/A';
            })
            .catch(error => console.error('Sensor update error:', error));
        }

        // Start auto-updates
        setInterval(updateSensor, 5000);
        // Initial update
        updateSensor();
    </script>
</body>
</html>
```

Key Features:

- Displays relay status (Powered On/Off with Amharic translation).
- Shows temperature and humidity in a styled `.sensor` div, updating every 5 seconds via AJAX (`/get_status`).
- Includes a 120-second countdown with SVG animation for relay activation.
- Responsive design for mobile devices.

Running the Application

1. **Activate Virtual Environment**:
2. `bash`

cd /home/cooling-automation/qrb/rpi

3. source venv/bin/activate
4. **Run the App**:
5. `bash`
6. sudo venv/bin/python relay_webapp.py
    - `sudo` is required for GPIO access (relay and sensor).
7. **Access Webpage**:
    - Open `http://<pi-ip>:5000` (find IP with `hostname -I`).
    - Expect:
        - Temperature (20-30°C) and humidity (30-60%) updating every 5 seconds.
        - Relay button triggering 120s countdown with SVG animation.
        - Logstash logs at 192.168.6.100:5959 (e.g., `Sensor reading: Temp=25.0°C, Humidity=50.0%`).

Troubleshooting

Issues encountered and resolved during integration:

1. **SyntaxError: Non-UTF-8 code**:
    - **Issue**: Degree symbol (°C) in `mylogger.info` caused parsing error.

    - **Fix**: Added `# -*- coding: utf-8 -*-` at the top of `relay_webapp.py`.
2. **ModuleNotFoundError: No module named 'Adafruit_DHT'**:
    - **Issue**: `Adafruit_DHT` library wasn't installed.
    - **Fix**: Initially attempted `Adafruit_Python_DHT`, but it failed with `Unknown platform` error.
3. **RuntimeError: Unknown platform**:
    - **Issue**: `Adafruit_Python_DHT` (v1.4.0) didn't recognize the Raspberry Pi (aarch64, Python 3.11).
    - **Fix**: Switched to `adafruit-circuitpython-dht`, which supports modern Pis.
4. **PEP 668 Externally Managed Environment**:
    - **Issue**: System-wide `pip` installs were blocked.
    - **Fix**: Created a virtual environment (`venv`) and installed `flask`, `python-logstash`, and `adafruit-circuitpython-dht`.
5. **ModuleNotFoundError: No module named 'flask'**:
    - **Issue**: Flask wasn't installed in the virtual environment.
    - **Fix**: Installed `flask` and `python-logstash` with `pip install flask python-logstash`.
6. **"N/A" on Webpage**:
    - **Issue**: Sensor readings failed, showing "N/A".
    - **Fix**: Verified wiring (+ to Pin 1, - to Pin 6, OUT to Pin 7, 4.7kΩ resistor) and switched to `adafruit-circuitpython-dht`.

General Debugging:

- **Standalone Sensor Test**:
- `python`

```
import adafruit_dht
import board
sensor = adafruit_dht.DHT22(board.D4, use_pulseio=False)
```

- print(f'Temperature: {sensor.temperature}°C, Humidity: {sensor.humidity}%')
  Run: `sudo venv/bin/python test_dht.py`. Expect valid readings.
- **Wiring Issues**:
    - Recheck connections and resistor.
    - Try 5V (Pin 2) with voltage divider or add 0.1µF capacitor for stability.
- **Logs**: Check console or Logstash for errors (e.g., `Sensor read error`).
- **Debug Mode**: Enable `app.run(debug=True)` for detailed Flask errors.

Future Enhancements

- **Alerts**: Add JavaScript alerts for high temperature (>30°C) or humidity (>70%).
- **Charting**: Integrate Chart.js for temperature/humidity trends.
- **Automation**: Trigger relay based on sensor thresholds.
- **Persistence**: Log sensor data to a file or database.

Conclusion

The AM2302 sensor was successfully integrated into the Flask app, displaying temperature and humidity on a webpage with 5-second updates, alongside relay control for the water cooling system. Using a virtual environment resolved PEP 668 restrictions, and `adafruit-circuitpython-dht` fixed platform compatibility issues. The system is now operational, with robust error handling and logging.

For further assistance or enhancements, contact the developer with error logs or feature requests.
