EC535 Final Project Technical Report

“Smart Environment Monitor”

Konstantin Agrachev, Tanveer Dhilon

Link to GitHub repository: https://github.com/konstan10/EC535-Final-Project
Link to YouTube video: https://youtu.be/GnkTK4IswFQ

Abstract

Environmental monitoring is crucial in our modern day. Factors such as temperature and humidity can have huge effects on health, industry, agriculture, and even just daily life. However, many such systems are not so robust or quite costly, and concerns about quality linger across some of these products. This project aims to design and implement a lightweight real-time environmental monitoring system based on the BeagleBone Black platform. The system will take in readings from temperature/humidity and air quality sensor modules, perform calculations for measures such as heat index, and display said readings on an interface running on the BeagleBone, while also pushing that data to a web server for logging and further analysis. In doing so, this project will serve as a demonstration of a better alternative to traditional store-bought environmental monitors.

Introduction

The motivation for this project is driven by the need for precise environmental monitoring in various applications and everyday life. By monitoring environmental data such as temperature, humidity, eCO_2, and TVOC levels, users can increase weather forecasting accuracy, assess pollution levels, better prepare for extreme weather conditions, and make decisions based on current environmental conditions. Businesses and industries must also meet certain environmental and health standards to operate, highlighting the need for a robust environmental monitoring system. Monitoring environmental data can also allow the user to identify health risks by calculating heat index and air quality and can allow the user to take necessary protective measures. This project will design a low-cost and power-efficient environmental monitoring system using the BeagleBone Black to allow users to monitor and identify patterns in their environment.

Method

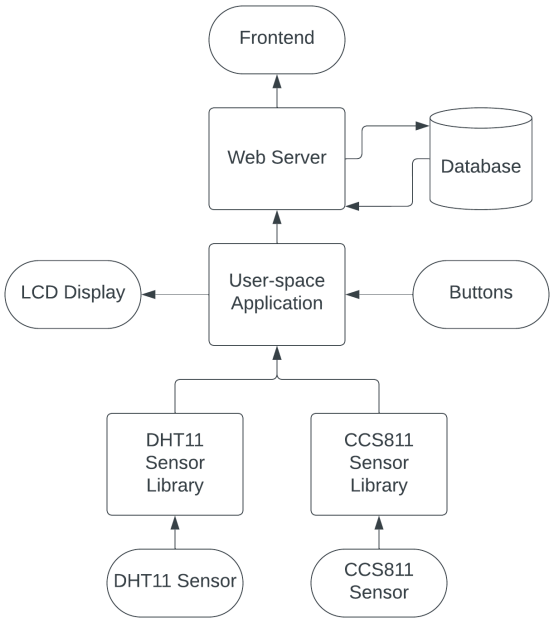

Figure 1: System Flowchart

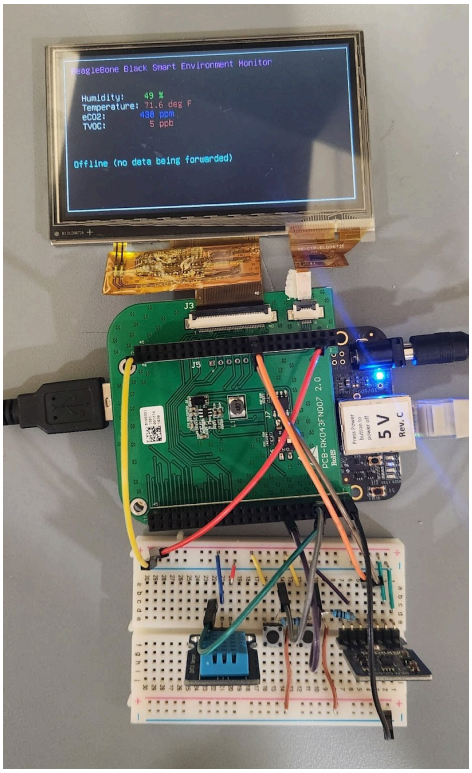

Figure 2: Physical Circuit

1. **Platform**

   This environmental monitoring system is based on the BeagleBone Black platform. We chose this platform as it was readily available to us and relatively simple to set up. It remains on the pricier side, although our code is reusable in other less expensive microcontrollers such as the Raspberry Pi. The image we used on top of our platform was the Debian 9.5 4GB SD IoT image, provided officially by the BeagleBoard Foundation [1]. We chose to use this image as it was the recommended image to use with our LCD (more information on this later), but being able to compile code directly on the platform using the built-in utilities as opposed to trying to cross-compile was an additional benefit.

2. **Components**

   Other than the BeagleBone Black, there were a few other components implemented into our project. To measure temperature and humidity, we used a DHT11 Temperature and Humidity Sensor, and to measure equivalent calculated carbon-dioxide and total volatile organic compound concentrations, we used a CCS811 Air Quality Sensor. Both sensors were readily available for local purchase and have decent documentation and libraries available on GitHub [2, 3]. For measurement display, we used a 4.3" BeagleBoard-compatible LCD provided to us for personal use. The only caveat of using this display was that it was only guaranteed to work with one image; using other images might have resulted in compatibility issues. Lastly, two 4-pin buttons were implemented: one to toggle the temperature unit (Fahrenheit/Celsius) and one to toggle data-sending to the web server.

3. **Code Overview**

   Our program lives in the userspace and was written in the C programming language. The main thread runs in a loop and serves the purpose of retrieving data from the sensor, displaying that data in a lightweight textual user interface using the ncurses library, and then optionally choosing to spawn a new thread to send that data to the web server using the curl library. As the sensors refresh every second, the main thread sleeps for one second at the end of each loop. The main thread also spawns another thread before it enters the loop, which polls the button states, checking for value changes in the GPIO files. The data gets sent through a tunneling service called ngrok, which is useful for opening up endpoints to the public. In the FastAPI backend, a POST request is initiated by the BeagleBone containing temperature, unit, humidity, eCO_2, and TVOC data to /data. This data is also stored into a MongoDB database. A GET request is then sent to fetch the latest data from /latest-data and host the index.html page. Within the index.html, a JavaScript program is written to fetch the

temperature, humidity, eCO_2, and TVOC data every 5 seconds and plotted on four charts created using Chart.js. In addition to this, functions are written to calculate the heat index based on temperature and humidity, and to classify the effects the environmental conditions can have on the user [4]. The air quality of the room is also classified based on eCO_2 and TVOC data. The average temperature and standard deviation of the temperature values stored in the database are also fetched using a GET from /average-temperature and /temperature-stddev respectfully.

```
function calculateHeatIndex(T, RH) {
  T = T * 9 / 5 + 32;
  let HI_simple = 0.5 * (T + 61.0 + ((T - 68.0) * 1.2) + (RH * 0.094));
  let HI = (HI_simple + T) / 2;

  if (HI >= 80) {
    HI = -42.379 + 2.04901523 * T + 10.14333127 * RH - 0.22475541 * T * RH
      - 0.00683783 * T * T - 0.05481717 * RH * RH
      + 0.00122874 * T * T * RH + 0.00085282 * T * RH * RH
      - 0.00000199 * T * T * RH * RH;

    if (RH < 13 && T >= 80 && T <= 112) {
      HI -= ((13 - RH) / 4) * Math.sqrt((17 - Math.abs(T - 95)) / 17);
    }

    if (RH > 85 && T >= 80 && T <= 87) {
      HI += ((RH - 85) / 10) * ((87 - T) / 5);
    }
  }

  HI = (HI - 32) * 5 / 9;
  return HI.toFixed(1);
}
```

Figure 3: Heat Index Formula

```
if (HI >= 52) {
  classification = 'Extreme Danger';
  effect = 'Heat stroke highly likely';
  backgroundColor = 'darkred';
} else if (HI >= 39) {
  classification = 'Danger';
  effect = 'Heat cramps/exhaustion likely';
  backgroundColor = 'red';
} else if (HI >= 32) {
  classification = 'Extreme Caution';
  effect = 'Possible cramps/exhaustion';
  backgroundColor = 'orange';
} else if (HI >= 27) {
  classification = 'Caution';
  effect = 'Fatigue possible';
  backgroundColor = 'yellow';
} else {
  classification = 'Normal';
  effect = 'No immediate danger';
  backgroundColor = 'lightgreen';
}
```

```
if (eco2 > 2000 || tvoc > 500) {
  status = 'Very Unhealthy';
  backgroundColor = 'darkred';
} else if (eco2 > 1500 || tvoc > 400) {
  status = 'Unhealthy';
  backgroundColor = 'red';
} else if (eco2 > 1000 || tvoc > 300) {
  status = 'Moderate';
  backgroundColor = 'orange';
} else {
  status = 'Good';
  backgroundColor = 'lightgreen';
}
```

Figure 4: Classification of Heat Index and Air Quality

Results

1. Physical Monitor

Every second, the monitor would successfully update the measurement fields with new data. A simple test we ran to see if the sensors worked was breathing hot breath into the

sensors. In doing so, we could see all measured values spike temporarily on the monitor, and then drop seconds after breathing, as expected. Occasionally, the sensors would return null values, but a simple check for that in the code solved that issue, instead displaying previous values.

2. **FastAPI and Uvicorn Server**
   The Uvicorn server was successfully able to host the index.html file, containing a dashboard to monitor the environmental data received from the BeagleBone Black. Within the webpage, the current temperature, humidity, eCO_2 in ppm, and TVOC in ppb are displayed, along with four charts for each data value. These values were successfully fetched every five seconds and plotted onto their respective graphs. All values received by the server were also stored into a MongoDB database. When running the FastAPI backend code, a 200 OK HTTP message was displayed showing that the HTTP requests were successful, and the data being sent from the BeagleBone Black was also printed. Along with this, the average temperature and standard deviation of the values stored in the MongoDB database are also displayed on the webpage. The heat index, classification of the heat index, effect of heat index, and air quality status are also displayed. The background colors of the classification, effect of heat index, and air quality status messages are also updated depending on the severity of the effects on the user's health.

Limitations and Future Work

Limitations regarding our system have to do with the level of its current scalability and functionality. For example, the web server is only designed to show results from one monitor, but the user might want to see results from multiple monitors. Data is also currently stored in a local database as opposed to a remote one, which limits access from multiple machines. The BeagleBoard Black is a rather pricey platform; we would want to see if we can make the system work on more budget platforms such as the Raspberry Pi, so as to cut costs but without heavily affecting performance.

In terms of functionality, we would want to integrate more types of sensors, as users might want other specific measurements depending on their application; this could be light, sound, wind, and so on. Sensor data can be used to derive all sorts of advanced measurements, and so, we would want to look into said measurements that can be derived, perhaps even making the use of machine learning models to do so. Addressing these scalability and functionality issues would make our project more viable and appealing for consumer use.

References / Resources

[1] https://www.beagleboard.org/distros/debian-9-5-2018-10-07-4gb-sd-iot

[2] https://github.com/bitbank2/CCS811

[3] https://github.com/ChadLefort/beaglebone-dht

[4] https://www.weather.gov/ama/heatindex