Option B: Weather Client - Final Report

John Maksuta   
Compute Science Department CS370-801  
Colorado State UniversityFort Collins, CO  
jmaksuta@colostate.edu  
  
  
Ali Fayed  
Computer Science Department CS370-801  
Colorado State UniversityFort Collins, CO  
alifayed@colostate.edu  
  
  
Matthew Boin  
Computer Science Department CS370-801  
Colorado State UniversityFort Collins, CO  
mboin@colostate.edu

*Abstract*—This is the first progress report for the Term Project for CS370-801 for our group 8B. The project objectives and justifications are introduced. Description of the Hardware, Software Environment, Project Design, and Development. We discuss what hardware we are using, obtaining the hardware and other materials with costs associated, and describe the project's intended design. We then briefly describe our software development environment. The project design is elaborated, with the primary and secondary implementations for accessing the device and data, directly and indirectly. Then the development section is introduced where we describe our current progress and efforts. This is the current state of the project at the time of this report. We then discuss the two quantitative attributes of the project that we intend to analyze, power consumption, estimates, and measurements, and the second attribute cost and marketability analysis. Figures and Tables contain images and artifacts described in the document. We close with acknowledgment to Adafruit and Ladyada. (*Abstract*)

Keywords—Raspberry Pi, single board computer, GPIO headers, sensor board, I2C communication, React, Spring-boot, Pi4J, Java, Python (key words)

# Introduction

The project objective is to develop and evaluate a system built using a single-board computer. The hardware requirement specifications are a single board computer, a sensor device, and communicating with at least one other computer. The single-board computer must have both Wi-Fi and OS boot capability.

Our project will measure, monitor, and report environmental factors, such as temperature, air pressure, humidity, and air quality over the Internet to a client machine. This progress report describes the current status of the project and our intent to evaluate quantitative information about the project. At the end of the project, we will submit a final report and a demonstration of the application we have produced.

We have chosen the Raspberry Pi 3 Model B+ for our single board computer [1]. It is the final revision of the third-generation single-board computer with Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC 1.4GHz processor, and 1GB LPDDR2 SDRAM. They are running the Raspberry Pi OS, formerly “Raspbian”, which is a “Debian-based operating system optimised for the Raspberry Pi hardware” [2].

For our sensor, we have chosen Adafruit BME688 - Temperature, Humidity, Pressure, and Gas Sensor - STEMMA QT [3]. The sensor features the Bosch BME680 Low power gas, pressure, temperature & humidity sensor [4]. Which is a precision device capable of measuring “humidity with ±3% accuracy, barometric pressure with ±1 ℎ𝑃𝑃𝑃𝑃 absolute accuracy, and temperature with ±1.0°𝐶𝐶 accuracy,” and can be used as “an altimeter with ±1 meter or better accuracy!” [3].

# Problem Characterization

## Obtaining the Hardware

Due to our remote team location, our team chose to obtain hardware separately rather than having a shared device to develop with. Although working individually, we have the same hardware board and sensor see Fig. 1-3, and Fig. 7. We chose the Raspberry Pi 3 B+ model computer board and the Adafruit BME688 sensor board [1][3]. All major hardware was already owned by the team members or was purchased through Adafruit’s website. The Raspberry Pi 3 B+ costs $35.00, and the Adafruit BME688 sensor board costs $19.95. The major hardware components totaled $54.95. There was an additional cost of $0.95 for STEMMA QT / Qwiic JST SH 4-pin Cable with Premium Female Sockets - 150mm Long, which is an adapter cable to the Adafruit BME688 board [5]. The STEMMA QT is a proprietary connector socket for connecting to Qwiic connections, that has leads with female jumper sockets for easy connection to the GPIO header pins. This cable was not necessary for function, but allowed our team to use the board without soldering an additional header onto the sensor board, and easy quick connection to the GPIO pins. Additional miscellaneous hardware was necessary, such as a micro SD card to contain the Raspberry Pi OS, although the OS is distributed freely through the Raspberry Pi foundation [6]. Members could also choose to obtain miscellaneous equipment such as HDMI cable to use a monitor and/or external peripheral devices like a mouse and keyboard, but this was not necessary for development.

## Software Environment

We are developing individually using Visual Studio Code IDE, and our code is maintained in a repository on GitHub (<https://github.com/CS-370-801-Fall24-Term-Project/Fall2024TermProject.git>). We are developing directly on the board, through ssh protocol from our development machines.

## Project Design

Our design has two components, primary and secondary. The primary component of the design is to have a website using a Node.js React client-side with a Java Spring-boot server-side. The server-side code will have direct communication to the sensor using I2C communication to the Adafruit BME688 sensor board. The Java interface to the sensor will use Pi4J to gain access to the GPIO headers on the Raspberry Pi and use I2C communication pins to communicate directly with the sensor board. A client will use the React UI to make a REST API request to the server side that will communicate directly with the sensor and obtain live data, returning it to the client and displaying it to the user. The user will be able to connect to the Raspberry Pi host using HTTP protocol through their web browser application.

Additionally, as part of the server-side implementation, the user will be able to connect using Java Socket connection, to a specified port listening for incoming communications.

The secondary design component is an application or service running in the background, separate from the primary component website client-server environment, that polls the sensor periodically and saves data to a database. This allows the client side to make requests to the server for historical data. For historical data client requests, the server side will directly access the same database that the background service inserts records into. The user can thus obtain historical data that was collected while the device was running, but the user was not currently logged onto the device. At minimum, the user can collect a .csv format file. We would like to display this using a graphical display as well on the client side UI.

# Proposed Solution and implementation strategy

We created the base code for our website using NPX create-react-app template. The server-side base code was created using the Spring Initializer [7]. We added a basic API endpoint using the Spring Framework to demonstrate the technology [8][9]. We created the project for the Adafruit BME680 interface using Pi4J library for communicating using the I2C communication protocol through the computer’s onboard GPIO header pins [10]. This project will be packaged as a JAR and referenced as a dependency in the server-side backend. Our code dependencies on the server are maintained using Maven [11].

Sensor Integration - Python works with the BME688 sensor wired to the Raspberry Pi through I2C protocol and successfully printing out temperature, humidity and air pressure readings.

Communication Setup - We verify that the Raspberry Pi will talk to client machine through HTTP and socket programming.

UI Development – So far, we have not made much progress just yet, we have a basic frontend webpage with mock data displayed, see Fig.6, and we plan to incorporate live data visualization on the web interface using Java-based Spring Boot.

# conclusion

## Methodology

## Libraries Used and Original Work

# Figures and Tables

A black rectangular object with wires

Description automatically generated

1. John Maksuta’s Adafruit BME680 Sesnor Board and Pi Case.

A black electronic device with wires and wires

Description automatically generated

1. John Maksuta’s Raspberry Pi Board and case, with GPIO header exposed and sensor leads connected.

A green circuit board with red and blue wires

Description automatically generated

1. Ali Fayed’s Raspberry Pi Board, with GPIO header exposed and sensor leads connected.

A green circuit board with many wires

Description automatically generated

1. Adafruit BME680 Sesnor Board and Raspberry Pi Board wired with I2C [12]

A screenshot of a computer

Description automatically generated

1. Ali Fayed’s testing environment using Python.

A screenshot of a weather dashboard

Description automatically generated

1. UI Development progress.

A green circuit board with red and blue wires

Description automatically generated

1. Matt Boin’s Raspberry Pi Board with GPIO expansion header and sensor wires connected.

##### Acknowledgment

We would like to acknowledge Adafruit for providing documentation and code samples for interfacing with their BME680 sensor product. Specifically to Ladyada from Adafruit who wrote the original Python code work “ adafruit\_bme680” of which the Java interface for the BME680 that we wrote is a derived work from their Python language source [13]. This is permitted by 2017 Ladyada for Adafruit Industries under license MIT AND BSD-3-Clause.

# Bibliography

1. Raspberry Pi, “Raspberry Pi 3 Model B+,” [Online] Available: <https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/>
2. Raspberry Pi OS. n.d. September 2024. <https://www.raspberrypi.com/documentation/computers/os.html>
3. Adafruit, “Adafruit BME688 – Temperature, Humidity, Pressure and Gas Sensor,” Adafruit, September 2024, <https://www.adafruit.com/product/5046>
4. Bosch. "Bosch BME680 Low power gas, pressure, temperature & humidity sensor datasheet." n.d. September 2024. <https://cdn-shop.adafruit.com/product-files/3660/BME680.pdf>.
5. Adafruit, “Qwiic JST SH 4-pin Cable with Premium Female Sockets,” Adafruit stemma, <https://www.adafruit.com/product/4397>
6. Raspberry Pi OS. n.d. September 2024. <https://www.raspberrypi.com/software/>.
7. Broadcom, “Spring Initializer,” 2005-2024. [Online] Available: <https://start.spring.io/>
8. Paraschiv, Eugen, "Spring RequestMapping," Baeldung, May 11 2024, [Online] Available: https://www.baeldung.com/spring-requestmapping
9. Code With Arjun, "Spring Boot Using VSCode," Youtube, youtube video, Aug 7, 2021, [Online] Available: <https://www.youtube.com/watch?v=dq1z9t03mXI>
10. Pi4J, “Pi4J,” [Online] Available: <https://www.pi4j.com/>
11. Maven, [Online] Available: https://maven.apache.org/download.cgi
12. Adafruit, "Adafruit BME680 Humidity, Temperature, Barometric Pressure & VOC Gas," Adafruit Learning System. [Online]. Available: <https://learn.adafruit.com/adafruit-bme680-humidity-temperature-barometic-pressure-voc-gas/python-circuitpython>
13. Adafruit, “Adafruit\_BME680,” Github repository, <https://github.com/adafruit/Adafruit_BME680> Broadcom, “Spring Quickstart Guide,” [Online] Available: <https://spring.io/quickstart>