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 final report for the Term Project for CS370-801 for our group 8B. The project objectives and justifications are introduced. The sections are Introduction, Problem Characterization, Proposed Solution and Implementation Strategy, and Conclusion. 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 discuss Problem Characterization, where the intent for the project and it’s requirements are elaborated. We move to the Proposed Solution and Implementation Strategy where we discuss the methodology including software environment and project design, followed by the libraries used and original work. We describe the project code, sources and reference code used in the project. We then close off with the conclusion. Figures and Tables contain images and artifacts described in the document. We close with acknowledgment to Adafruit and Ladyada and the bibliography. (*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

## A Remote Sensor for Monitoring Environment

Climate and environmental data is necessary for studying the environment and the effects that human activity has on it. There is often a shortage of manpower and equipment for studying the environment. This leads to areas of the map which have little to no data available for analysis. Scientists also face funding issues and tight constraints on budgets, which can affect their options for equipment. Using a low-cost data collection method, that allows remote access to the device is a challenge.

Our task was to communicate with a single-board computer that is using a sensor. The communication can be broken into two smaller tasks, communication with the single-board computer and communication between the single board computer and its sensor. In our case this would be to communication with an environmental sensor to obtain readings and then communicate that back to a user client on their computer.

# Proposed Solution and implementation strategy

## Methodology

## Libraries Used and Original Work

The libraries used are create-react-app created with the node package manager. This created the template code for our client-side.

We also used Springboot and Spring libraries for the server side, this was created using the Spring initializer which generates some base code with specified dependencies.

Pi4J library was also used by the backend server to interface with the GPIO header pins. Specifically, it uses I2C communications to send and receive signals from the I2C GPIO header pins. Accessing the GPIO headers of the Raspberry Pi, natively, requires writing device drivers which is beyond the scope of this course, and thus we require a 3rd party library to perform this task.

A derived work, which is an original work based on an existing work, was created for interfacing with the sensor board. The Adafruit BME688 board has a setup guide and detailed tutorial, which links to their Adafruit CircuitPython BME680 library, written in Python. There are no other resources other than code samples and the setup guide. Our derived original work is based on the Adafruit CircuitPython BME680 library and is compliant under MIT and BSD-3-Clause license to be modified and redistributed. What is original about our work is that it is written in Java instead of Python, and uses the Java language and object oriented design features of Java language. The original design was adapted to Java, separate class files were created for the original Python classes, code was rewritten to move large code blocks into smaller functions with single responsibility principle, and new classes were added. New classes in the package com.sensor are Sensor class and com.sensor.demo contains SensorDemo class. The Sensor class is used by external applications to control the sensor, retrieving data directly, and the SensorDemo contains a main function for demonstration of the Adafruit code within the library. Our project library is called “adafruit\_bme680” and resides in its own project as a JAR dependency for the server. This is necessary because we are implementing Adafruit’s sensor board, and the only resources that are available are code sources open to modification, as opposed to technical specifications, API, SDK, or other technical references. Initially, it was intended to perform I2C and SPI communications, but due to time constraint the I2C is the only tested and working part of our library.

Our original code in the backend server includes the classes for creating the REST API, the sensor controller, and the data transfer object for the sensor data. There are two packages com.cs370.project.weather and com.cs370.project.sensor.

WeatherApplication class contains the man function for the application. WeatherController is the Spring Framework REST API controller, where endpoints were added to allow communication. The endpoint “/sensordata” is used to retrieve the data which returns SensorData using and instance of SensorController. Sensor Controller class is uses the class com.sensor.Sensor, from adafruit\_bme680 project, to retrieve the reading data from the sensor device, and

The client-side frontend includes original code for retrieving and displaying the sensor readings. The web page is very simple and contains a display and a button to retrieve the data. It uses the src/components/Page.js to display the UI elements in the main page.

There is a Python code file written which will communicate with the sensor and write data to a database. It uses board, digitalio, busio, adafruit\_bme680, mysql.connector as import dependencies. There is a SensorDB\_Setup file for how to setup the code to run as a system in the project called Database. Unfortunately, there is currently no interface for the database to the server backend and thus, not communication to the client side from the database stored values. Initially we thought we had until the end of November to complete the assignment, and if given that time, we would be able to complete that part. Essentially the Database project currently operates as an autonomous service that reads data from the sensor and writes that data to a database.

# conclusion

The primary objective of the assignment, to use a single board computer to communicate with a sensor device and at least one external computer communicate with the single board computer was completed successfully. We had success in communication with the sensor device using I2C communication protocol, which is currently implemented in the server backend. SPI communications protocol was not completed, although we intended to solely use I2C for our communications. SPI was something that if this project continued to market, we would like to provide. We would like to provide historical data that would allow analysis of the data, and package it for transport. If we had had more time in the project to complete this task, we are close to delivering this functionality. Our team’s own goals in this project, to provide extra functionality are still under development.

# 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>
14. Adafruit, “Adafruit BME680 detailed tutorial,” [Online] Available: <https://learn.adafruit.com/adafruit-bme680-humidity-temperature-barometic-pressure-voc-gas>
15. Adafruit, “Adafruit CircuitPython BME680,” [Online] Available: https://github.com/adafruit/Adafruit\_CircuitPython\_BME680