CS 437: Internet of Things
Final Project Report

Team

- Eric Schrock (ejs9)
- Devin Schmitz (devinms3)

Overview

We created a distributed network of CO2 sensors that report back via wifi to a central hub. The hub exposes a web interface on the LAN for accessing the data. This allows us to track variations in CO2 levels throughout a home and over time. This data could be used to understand and improve indoor air quality, leading to healthier lives.

Motivation

todo: "Talk about the importance of this problem. Pretend you are trying to convince someone to give you funding, or purchase what you developed. You may also want to give references/citations here."

todo: Rehash motivation section from the proposal. Maybe remove "low energy" as a design goal.

Technical Approach

Our overall system consists of a base station connected to a network of sensors over wifi. The base station is a Raspberry Pi 4B running a server. Sensors connect to the server and periodically report CO2 measurements. The base station then presents those measurements on a website available on the local area network.

Each sensor consists of a Raspberry Pi Pico WH connected to a CO2 sensor over I2C. Each sensor runs a client that is responsible for connecting to the server on the base station, reading CO2 measurements, and sending those measurements to the base station.

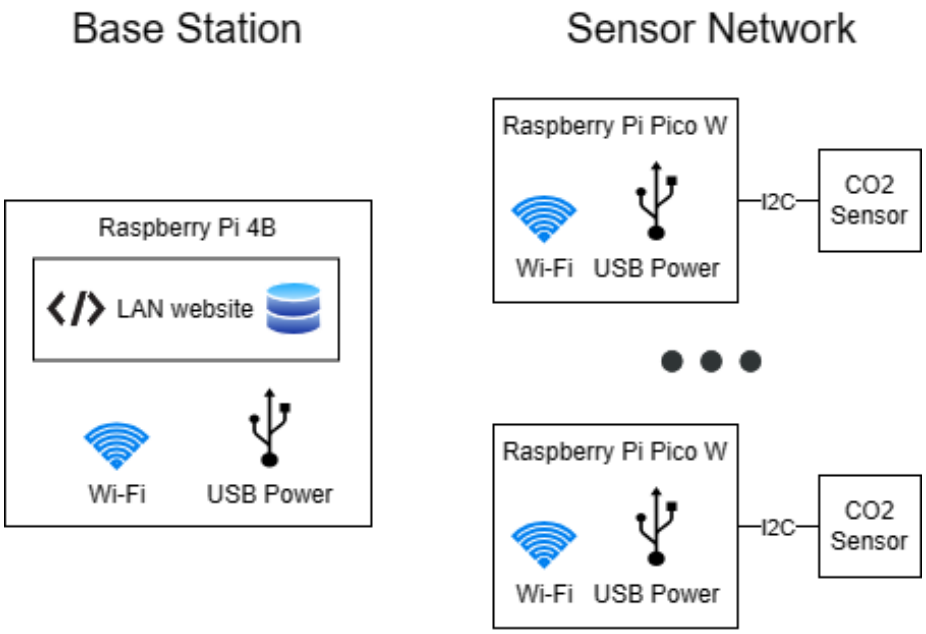

Figure 1: System Architecture

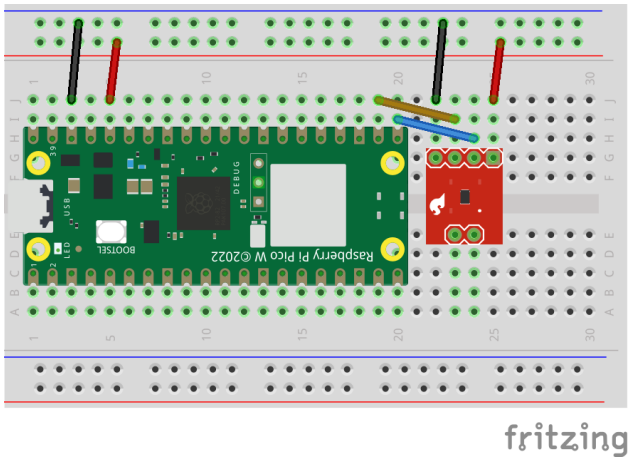

Figure 2: Sensor Layout

Note: The CO2 sensor shown in the Fritzing diagram above does not match the actual part and is only meant to give a general sense of the design.

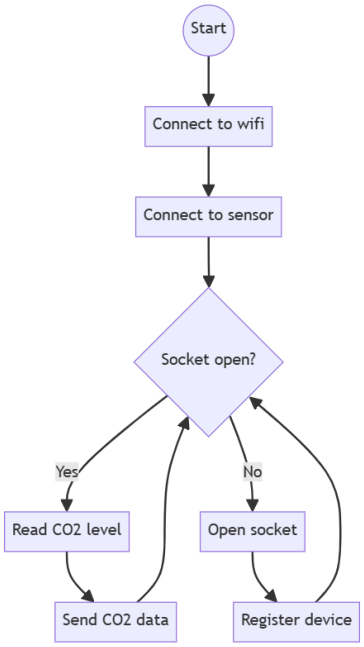

Figure 3: Sensor Flowchart

todo: Add a high-level diagram of the server code (details belong in the next section)

todo: Add a high-level diagram of the website code (details belong in the next section)

Implementation Details

Sensors

We spent a significant amount of time comparing[1] different options for the microcontroller on our sensors units. We started our search with six different Arduino variants (five Nano variants and the Micro). We looked at price, availability, breadboard compatibility, power and IO options, and connectivity (wifi, Bluetooth, and BLE). We were leaning towards the Arduino Nano 33 IoT[2] when we found the Raspberry Pi Pico WH[3] [4], which checked all our boxes but for a much cheaper price ($7 vs $27).

[1] https://github.com/EricSchrock/co2-monitor/blob/main/docs/microcontroller.md
[2] https://store-usa.arduino.cc/products/arduino-nano-33-iot-with-headers
[3] https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html
[4] https://www.pishop.us/product/raspberry-pi-pico-wh-pre-soldered-headers

The Pico comes with both C/C++[5] and Python[6] SDKs. We are both embedded C programmers for our day job, so we picked the Python SDK to get some experience with MicroPython[7]. Additionally, the simplicity of Python fit with a simple prototype on a tight timeline.

We also considered[8] multiple CO2 sensors, before settling on the ENS160[9]. Our criteria were price, availability, breadboard compatibility, and power and IO options.

Additionally, we had to consider whether to choose a true CO2 sensor or an equivalent CO2 (eCO2[10]) sensor. eCO2 sensors don't measure CO2 directly. Instead, they measure total volatile organic components (TVOC) and use that measurement to estimate the CO2 level. eCO2 sensors are much cheaper but are less accurate. We chose to go with an eCO2 sensor, as price is a big factor in our design. Having multiple sensor units allowed us to sanity check the reported values and notice anomalies. Given this, the eCO2 sensor was accurate enough to observe trends and draw conclusions. More on this in the CO2 Data Analysis portion of the Results section.

The SparkFun ENS160[11] came with the option to communicate over I2C or SPI. We chose I2C because it was simpler to implement in MicroPython.

The Pico supports wifi, Bluetooth, and BLE. We initially planned to use BLE to communicate with the server, but implementing wifi communication turned out to be much simpler. Additionally, power usage turned out to be less of a concern than we initially thought because we chose to power the sensor units off wall power instead of batteries.

We chose to connect up the Pico and ENS160 on a half sized breadboard and to power them both through the micro USB port on the Pico. We built both dev units and full prototypes.

[5] https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-c-sdk.pdf
[6] https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-python-sdk.pdf
[7] https://micropython.org/
[8] https://github.com/EricSchrock/co2-monitor/blob/main/docs/co2-sensor.md
[9] https://www.sciosense.com/wp-content/uploads/documents/SC-001224-DS-9-ENS160-Datasheet.pdf
[10] https://electronics360.globalspec.com/article/17986/what-are-eco2-sensors
[11] https://www.sparkfun.com/products/20844

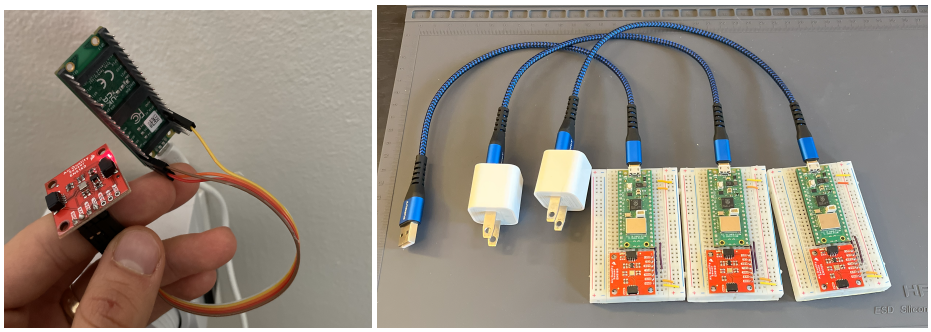

Figure 4: Development Unit (left) and Full Prototypes (right)

The software development for the sensor unit was relatively straight forward. We wrote simple programs to aid hardware bring up, first to blink the Pico LED and then to turn the Pico LED on when we breathed on the CO2 sensor. Then we created a simple program that connects to wifi, connects to the CO2 sensor via I2c, connects to the server, and then periodically reads and reports the CO2 level (see the sensor flow chart above in the Technical Approach section).

Server

todo: "This is where you give the details in your implementation. Talk about specific software packages you used, hardware modules, any algorithms or research papers you referred to, data structure and protocol choices, etc. You should provide at least an informal list of citations of all these external materials that went into your project."

todo: Include picture of the Pi in a hub. Include a link to Pi website. Talk about how we chose the Pi (had it already through the class) (otherwise would have chosen cheaper model with less RAM). Talk about challenges with server process dying (solved with rc.local and cron job).

Website

todo: "This is where you give the details in your implementation. Talk about specific software packages you used, hardware modules, any algorithms or research papers you referred to, data structure and protocol choices, etc. You should provide at least an informal list of citations of all these external materials that went into your project."

todo: Talk through how we chose the web technology (Flask?). Include link to Flask website.

Results

Project Objectives

todo: "So, how did things turn out? You can provide performance results, experiences you had interacting with it, etc. Also talk about what the takeaway is - why should we care about your results? And, it is ok for things to go wrong - what did not go right in your project, what was hard and what lessons did you learn?"

Ideas

- Talk about price (compared to CO2 sensor mentioned in the motivation section) (see the BOM in the repo README)
  - Drive down price with economies of scale and resource usage optimization (e.g. could use a cheaper Pi with less RAM for the hub)
- Talk about proposal timeline vs reality
  - Initial design took longer than expect? (lots of options to choose between)
  - HW bring up went more smoothly than expected and no 2nd prototype was needed
  - Etc.
- See proposal motivation for other topics to hit on

CO2 Data Analysis

One question we had was whether our sensors would be accurate enough to provide useful data. We found that the data trends were reliable enough to interpret and act on. For example, below is around thirteen hours of CO2 data from three different rooms. Below that is a list tying trends in the data to my (Eric's) activity that day.

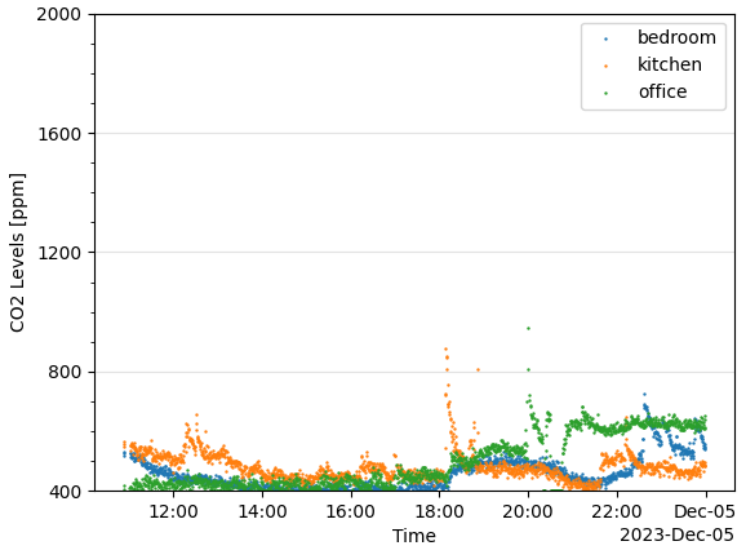

Figure 5: CO2 Data from Eric's Home (12/4/23)

1. 11 AM - 2 PM: The bedroom CO2 level drops from its nighttime high.
2. 12 - 1 PM: The kitchen CO2 level jumps as my wife and I eat lunch.
3. 1 - 6:15 PM: The bedroom sits empty and has the lowest CO2 level. I sit at my desk in my office and my wife moves around the kitchen and adjoining living room. The kitchen CO2 level is higher than the office CO2 level because humans breath out more CO2 when we are active.
4. 6:15 PM: The kitchen CO2 level spikes when we vent the pressure cooker and then dissipate to the rest of the house, leaving every room at a higher CO2 level.
5. 6:30 - 7 PM: The kitchen CO2 level jumps as my wife and I eat supper.
6. 7 - 9:30 PM: My wife leaves for time with friends and I return to my office. The kitchen CO2 level drops below that of the office.
7. 8:15 PM: I briefly open a window in my office. The office CO2 level drops to the minimum sensor value of 400 ppm, but jumps back up as soon as I close the window.
8. 8:30 - 8:45 PM: I open a window in my office for 15 minutes. The office

CO2 level again drops to the minimum, but jumps back as soon as the window is closed.

9. 9:30 - 10:15 PM: My wife returns and we spend time together in the living room, near the kitchen. The kitchen CO2 level jumps.
10. 10:15 PM: We go to bed. The kitchen CO2 level drops and the bedroom CO2 level jumps.

The sensors do have a couple issues to keep in mind. First, as explained in the section on implementation details, they are measuring eCO2 which is an estimate of CO2 levels derived from TVOC levels. TVOCs can spike without CO2 spiking. For example, the data above shows a spike in the office CO2 levels at around 8 PM. This was from running a paper shredder near the sensor.

Second, the sensor results can occasionally get stuck at a high offset until they are power cycled. The CO2 level in the office jumped significantly the evening of December 4th (above) and remained high into the next day (below). I first tried opening windows throughout the house from 10:15 - 10:30 AM. The other rooms responded to this, but the office CO2 level jumped back up as soon as the windows were closed. At around noon, I power cycled the office sensor.

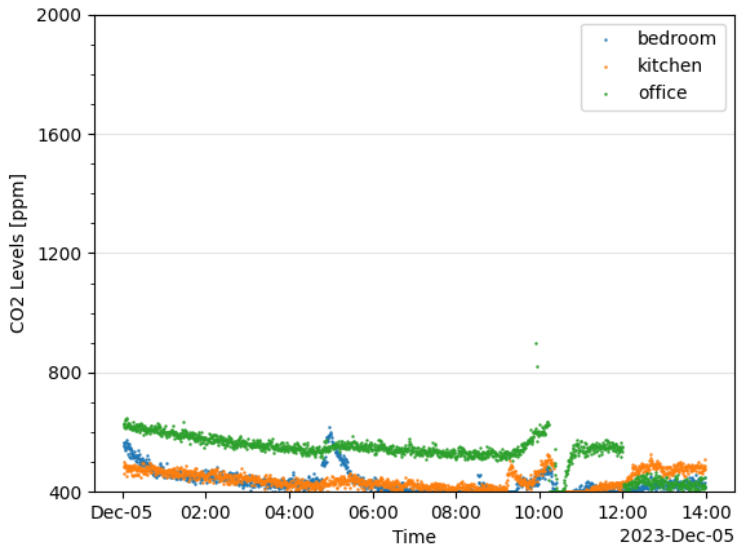

Figure 6: CO2 Data from Eric's Home (12/5/23)

Despite these issues, we believe that the sensor data captured is accurate enough to drive decisions, such as whether to open windows or whether to invest in a new HVAC that cycles in outdoor air in response to high indoor CO2 levels, as long as the sensor limitations are kept in mind.

todo (Devin): Do you have any interesting data or observations to add? How does your data compare to mine?

Demo Videos

We recommend watching these videos on a large screen and/or setting your video player to HD/1080p.

- Hardware Demo
- Website Demo

todo (Eric): Record a demo video of how to setup the hardware.

- Sensor and base station hardware
- Installation (https://github.com/EricSchrock/co2-monitor/blob/main/src/README.md)
- Talk through LED transitions
- Show server log and CO2 data files

todo (Devin): Record a demo video of how to use the website.

- Installation (https://github.com/EricSchrock/co2-monitor/blob/main/src/README.md)
- Talk through and demo website features
  - Date selection
  - Sensor/room selection
  - Refresh for latest data
- Talk through interesting trends observed in the data

Project Repository

https://github.com/EricSchrock/co2-monitor

What We Learned

Eric

Two of the key skills I learned in this project were networking via the Python sockets library and how to configure Linux using the `/etc/rc.local` file and `cron` jobs. Both were completely new to me and are valuable experiences for the future.

On a different note, I was surprised by how easy it was to get the project up and running on the Raspberry Pi Pico with MicroPython. I'm used to embedded boards being much more painful to bring up. Perhaps some personal projects I thought would be too time consuming are actually in reach!

This project also taught me about the health impacts of high concentrations of CO2 in indoor spaces. This is important knowledge to help me stay sharp. As a remote worker, I can put what I've learned to use by opening my office windows at key times.

Devin

todo (Devin): Add a paragraph on things learned. Compare to the things you expected to learn in the project proposal.

If We Had More Time

If we were to take this project further, we would have three main goals. The first would be to increase the accuracy of the sensor readings, either by tuning them with temperature and humidity readings reported to them by the base station or by automatically power cycling the ENS160 CO2 sensor periodically.

Second, we would expand the functionality of the website. We would add the ability to save timestamped notes to mark and explain phenomena in the data. We would also make the time interval for the display configurable. Additionally, we would add simple statistics, such as the max and average over the selected time interval.

The third goal would be to condense the sensor unit into a wall wart with a protective housing. We would look into 3D printing for the case and into a custom PCB to fit in a smaller form factor.

Conclusion

In conclusion, we accomplished the high level goal of our project, to prototype a distributed, networked, and affordable CO2 monitoring solution. Along the way, we tried new technologies, built new skills, and gained a deeper understanding of the quality of the air in our homes and of how it impacts our health.