UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Automated Sensor-Based Filtration System

Prithvi Saravanan, Omar Koueider, Karthik Talluri

05/01/2023

Agenda

Overview
Final Design
PCB Design
Results
Questions

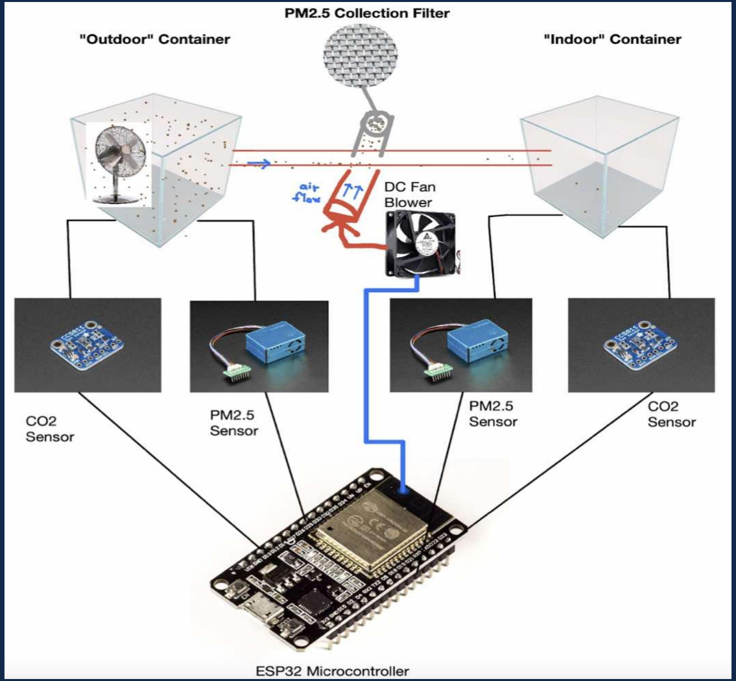
Overview

Objectives

- Burning fossil fuel, industrial emissions and forest fires cause air pollution

- There are more types of toxic gases and particulates in the air

Solution

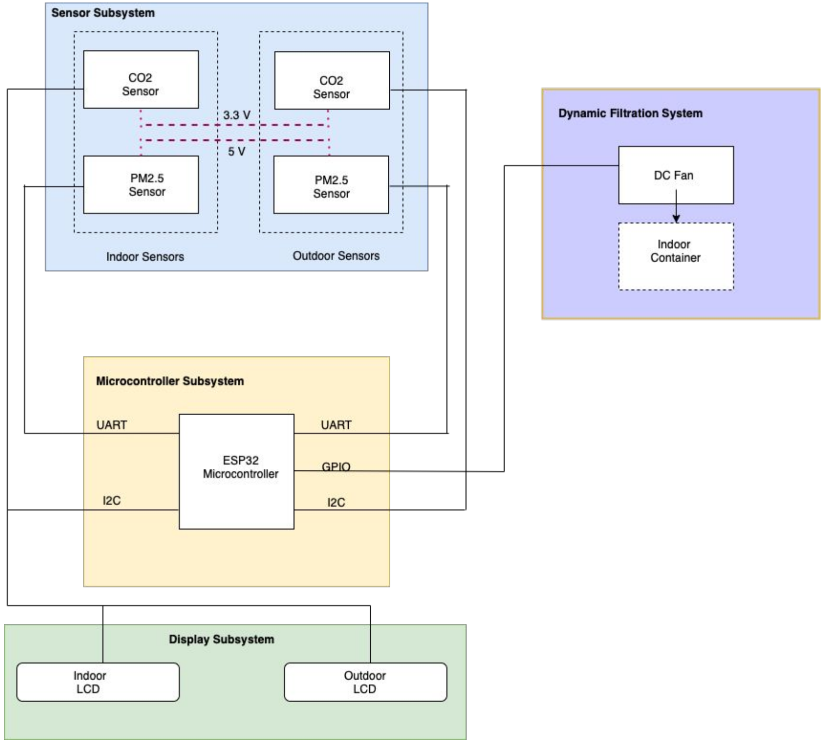


- A dynamic filtration system that adjusts according to particulate concentration
- Keep the indoor particulate concentration constant.

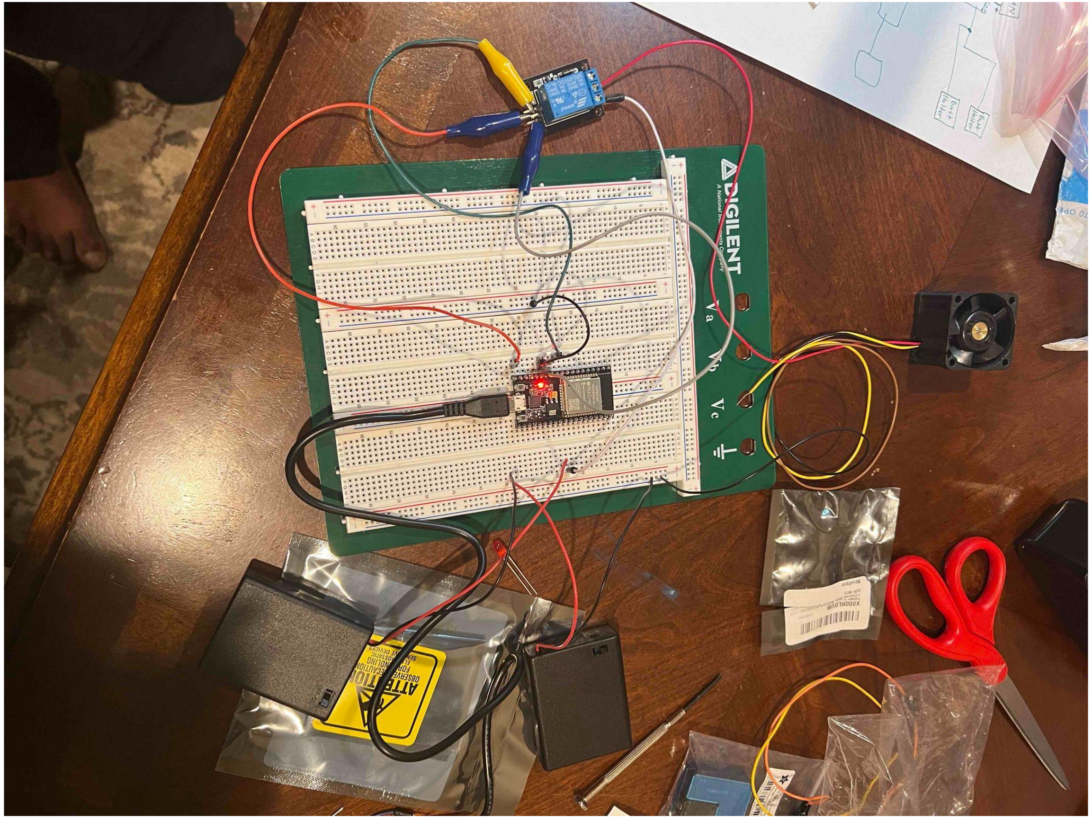
Block Diagram

There are 4 subsystems represented:

- Sensor Subsystem
- Microcontroller Subsystem
- Dynamic Filtration Subsystem
- Display Subsystem



FAN FIRST



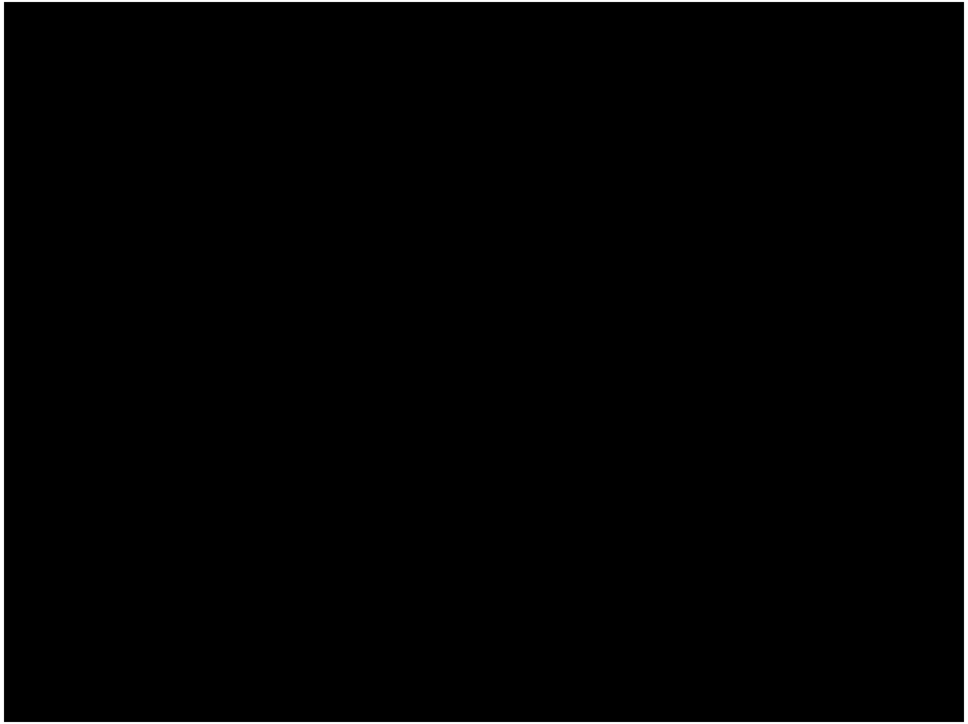
Started with Fan

- First device to be tested
- 12V Battery and a relay to protect ESP32
- Tested Pulse Width Modulation PWM to control speed

Ended with Fan

- Final piece was to add logic for fan speed control with data from sensors.

PM2.5 Sensor - UART

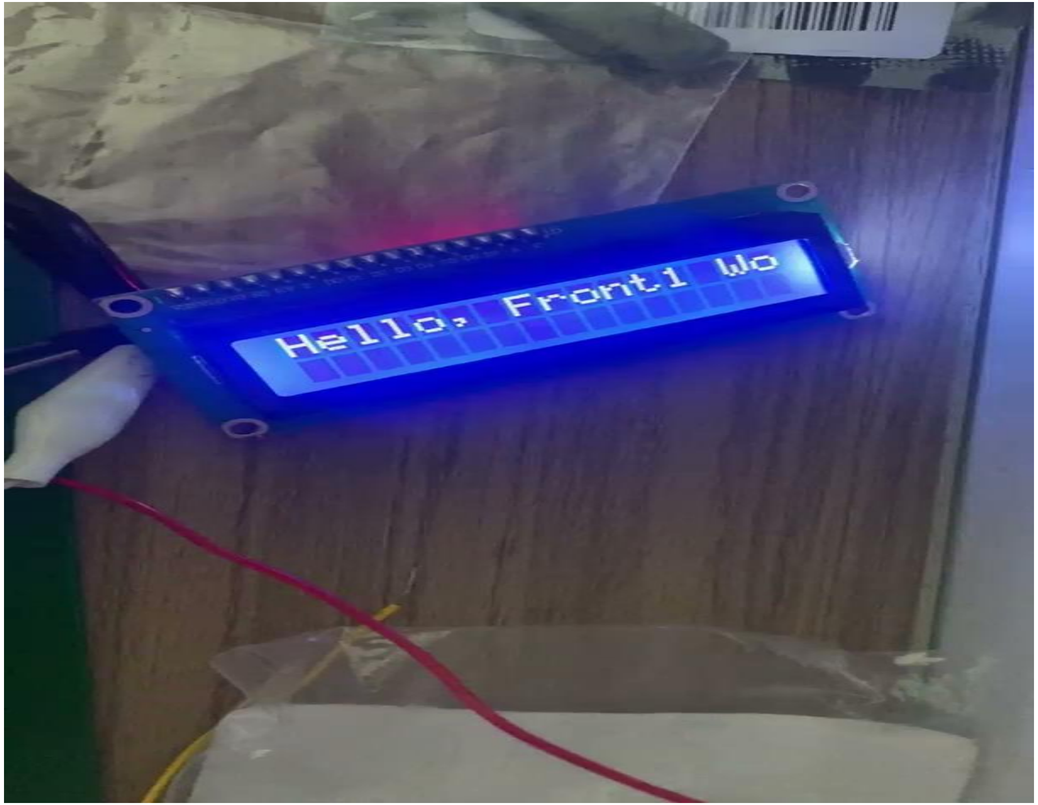


- 2 GPIOs for UART Rx and Tx per PM2.5
- Video shows running standalone program first time

Deadline Day - 1 Issue - Checksum Error

- PM2.5 streams data almost every 1 second (0.95 secs approx)
- If loop doesn’t read on time - will result in checksum error
- Took a while to realize that we had added a delay to the integrated code
- Could have done polling

Hello Display - I2C



Standalone Display program for 2 LCD displays.

Messages randomly goes to both LCDs

- LCDs come with default I^2C address 0x27
- Needed to hook up 2 LCDs to Default Serial Data(SDA) and Serial Clock(SCL) of ESP32
- On an I^2C bus, addresses have to be different for each device
- Shorted address solder pads to change one LCD address

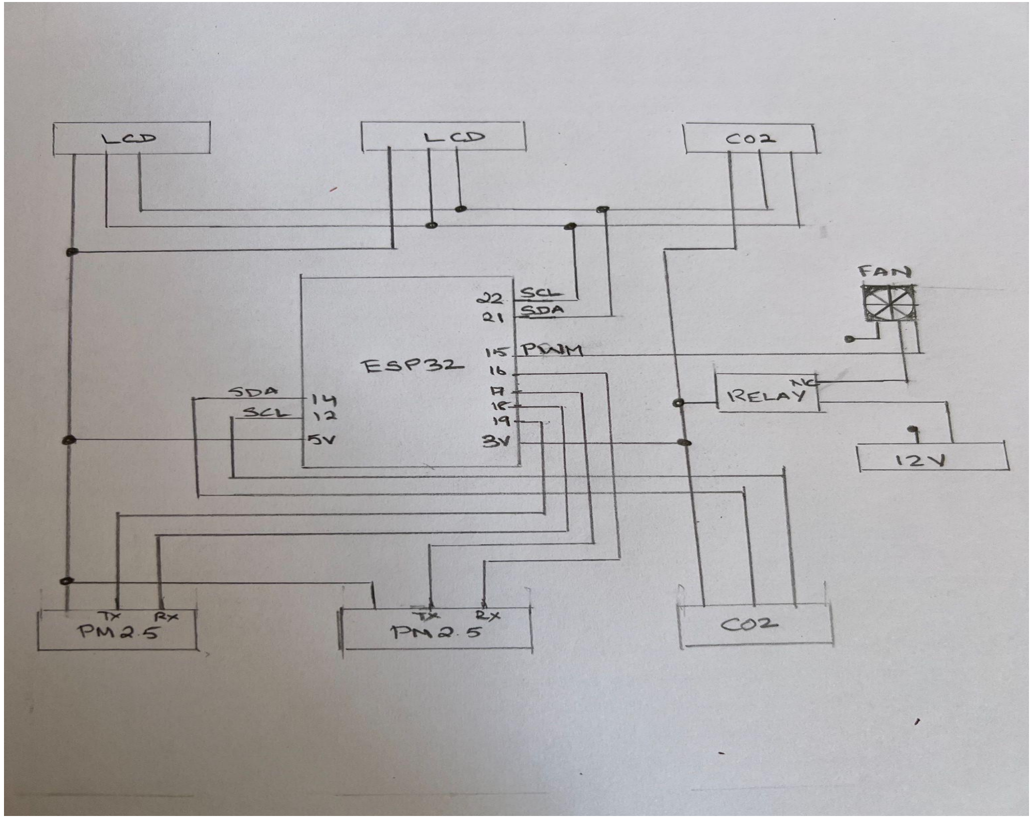
CO2 Sensor - I2C

- Library support for 1 SCD30 CO_2 sensor in the system.
- Modified library to support multiple SCD30s.

Two CO_2 on default I^2C ports not possible

- Default I^2C address 0x61 but no solder pads to modify.
- I hooked one CO_2 sensor with LCDs
- Converted couple of GPIOs to act as SDA and SCL for the other CO_2 sensor.

Circuit

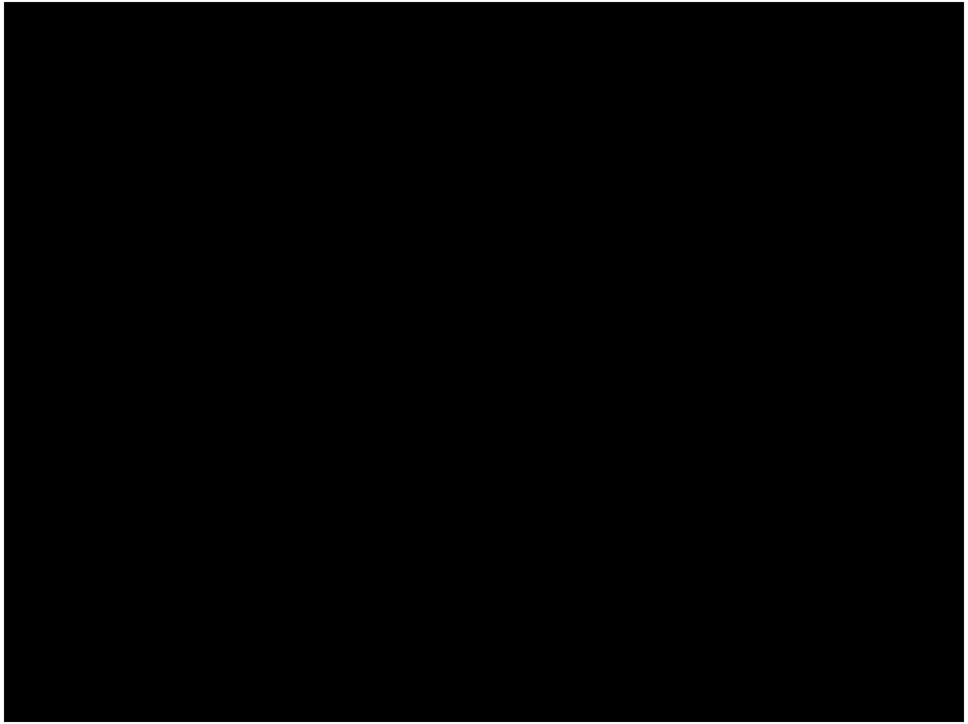


LCD
LCD
CO2
FAN
ESP32
22 SCL
21 SDA
15 PWM
16
17
18
19
3V
SDA 14
SCL 12
5V
RELAY
NC
12V
TX RX
PM2.5
TX RX
PM2.5
CO2

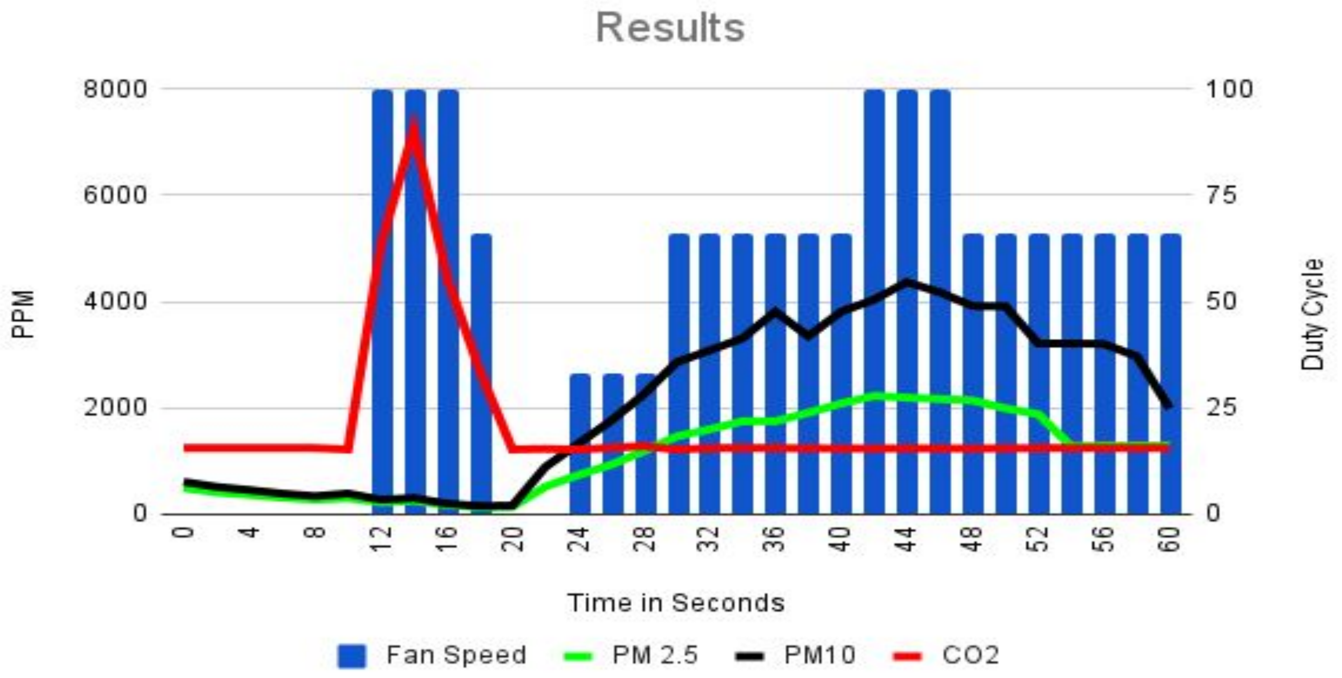
Integrated Microcontroller System

- Sensors fill up global data structures
- 4 ranges for PM2.5, PM10 and CO_2
    - MODERATE → DUTY CYCLE 0 → FAN SPEED OFF
    - UNHEALTHY → DUTY CYCLE 33 → FAN SPEED LOW
    - VERY UNHEALTHY → DUTY CYCLE 66 → FAN SPEED MEDIUM
    - HAZARDOUS → DUTY CYCLE 100 → FAN SPEED HIGH
- PM2.5, PM10, CO_2 and FAN DUTY CYCLE are displayed on the LCD
- Power efficiency is achieved by turning the fan off at moderate levels

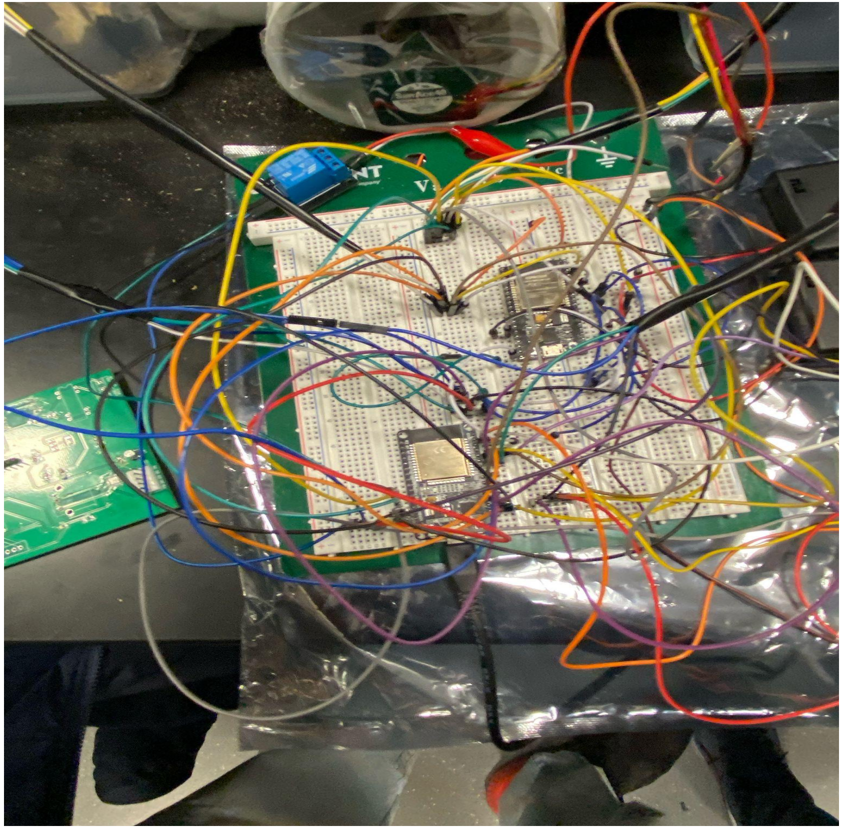
System in Action



Graph from Sensor Data and Fan Speed



Deadline Day Chaos



- On demo day ESP32 fried
- Relay protecting ESP32 from 12V battery failed
- Borrowed an ESP32 and rewired in an hour
- Fried ESP32 is still on the breadboard hence the mess

Final Design

High Level Requirements

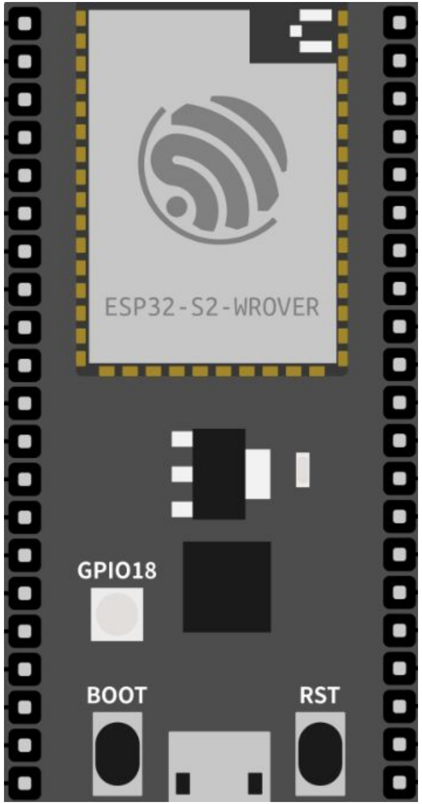
- The concentration of PM2.5 particles in the clean enclosure should be less than that of the outdoor enclosure by approximately 75%
- The concentration of CO_2 will determine whether air flow will speed up or slow down based on circulation
- The dynamic filtration mechanism must only start running when the PM2.5 or CO_2 particles reach a certain threshold

Components

- 2 PM2.5 Sensors - one indoor and one outdoor
- 2 CO_2 Sensors - one indoor and one outdoor
- ESP32 Microcontroller
- 1 Airflow Fan
- 1 Filter Fan blower supporting PWM function and relay
- 2 LCDs - one for indoor sensor data and one for outdoor sensor data

Microcontroller subsystem

- The ESP32 Microcontroller (MCU) is central to the filtration system
- Initializes UARTs for PM2.5 sensors
- Initializes I^2C with device addresses to drive CO_2 sensors and LCD displays
- Initializes fan frequency and PWM to control speed
- In a loop reads data from the sensors
- Determines the fan speed based on the readings
- Ramps up or ramps down the fan speed
- Displays the sensor data on the LCD

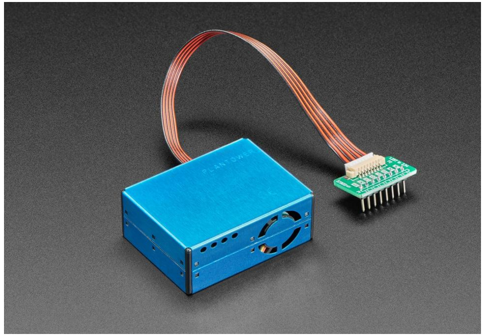


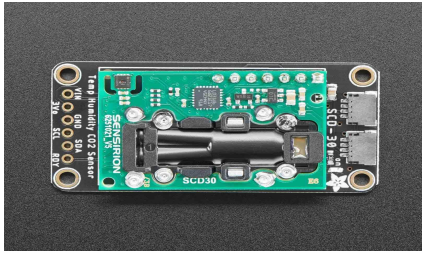
Filtration subsystem

- The Sanyo Denki fan is the key driver for the filtration subsystem
- Supports **Pulse Width Modulation** (PWM) for speed control
- Based on the sensor readings the MCU controls the fan's duty cycle
  - Thresholds:
    - PM2.5: 250, 500, 750, 1000 [mg/m^3]
    - CO2: 1400, 1700, 1900, 2200 [parts/million]
- The fan redirects the air inflow to the filter
- A relay is used between MCU and fan due to the voltage difference

Sensor subsystem

- PM2.5 Air Quality Sensor that senses particulate matter
- One PM2.5 outdoor and one indoor
- 2.5µm and 10µm readings of outdoor PM2.5 used by microcontroller subsystem
- Connected to MCU using UART



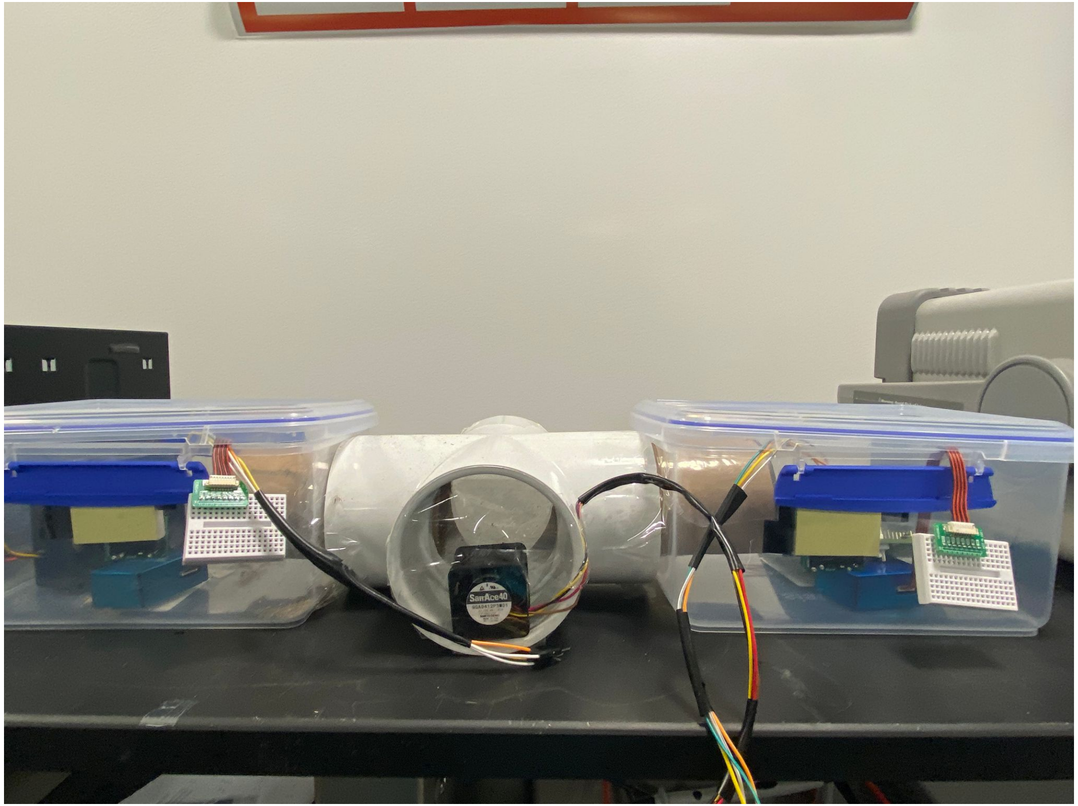


- SCD30 CO_2 sensor that senses Carbon Dioxide
- One SCD30 outdoor and one indoor
- The SCD30 readings is used by microcontroller subsystem
- Connected to MCU using I^2C

Display subsystem

- There are two LCD displays
- One displays outdoor CO_2 value, PM2.5 particulate and PM10 particulate values and Fan duty cycle
- The other displays CO_2, PM2.5, and PM10 values of indoor
- The LCDs are driven from MCU through I^2C

Mechanical Design



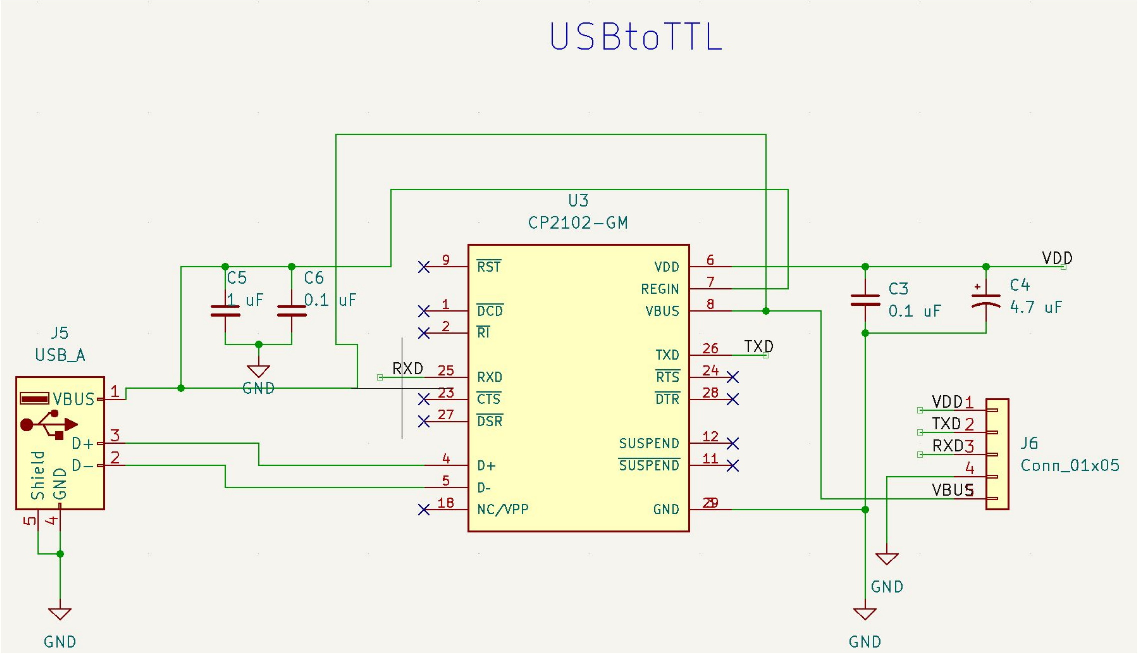
PCB Design

USB Power/Programming Schematic

Power

- 5V is supplied to the ESP32 microcontroller

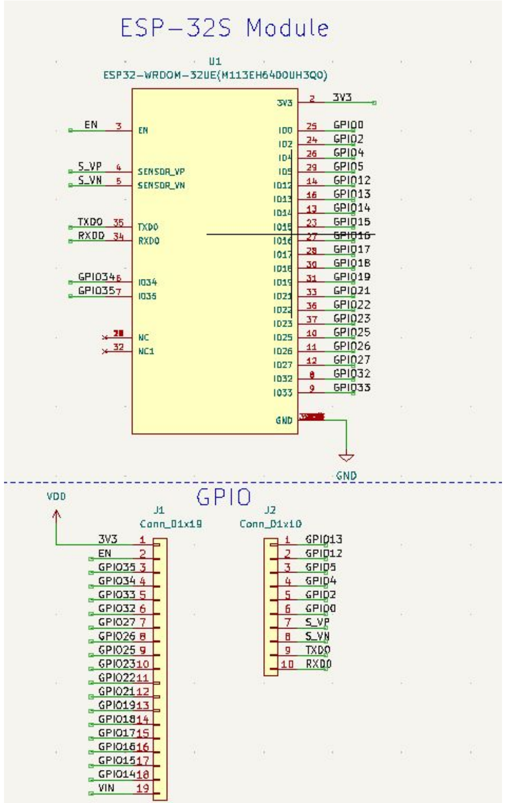
- We initiate a step down from 24V to 5V



ESP32 Schematic

Microcontroller

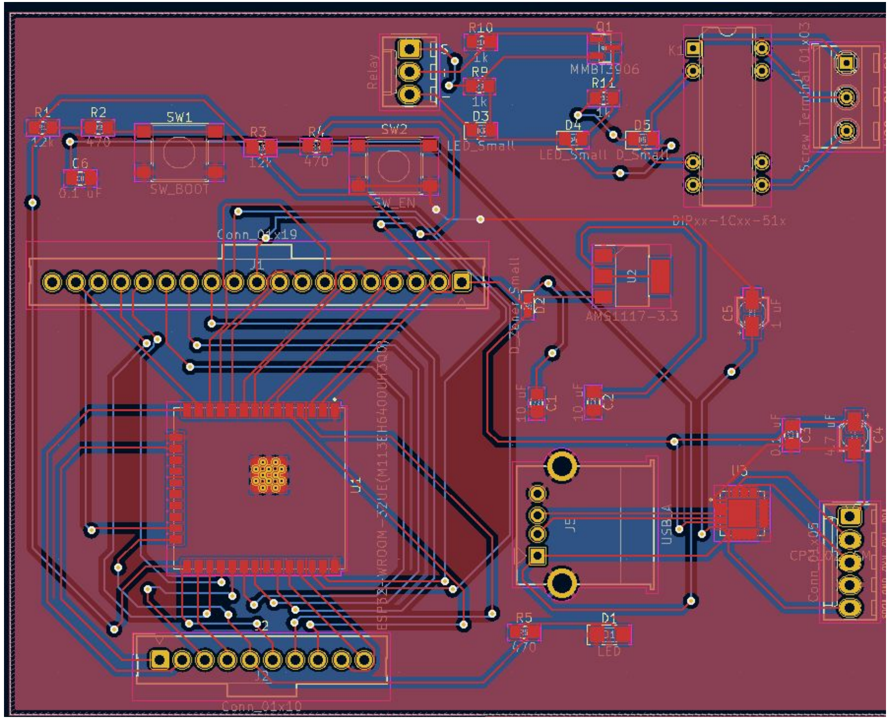
- We used an ESP32 microcontroller for the microcontroller subsystem
- The GPIO pins for UART, TX, and RX in particular were used to communicate with the subsystems
  - Each of them are used for the sensors
  - Specific GPIO pins were used to relay fan speed and communicate with the voltage regulator (relay)



PCB

MISC

- USB-A for ease of programming
- Went unused in final project



Results

What Went Well?

- Microcontroller/Sensor subsystems went very well with some minor hiccups
    - Assigning different I^2C address to LCDs - shorted
    - Modified CO_2 sensor library code to accommodate 2 sensors and on 2 different pins
- Filtration subsystem performed extremely efficiently
    - Fan was changing according to data from both sensors
    - Clean environment stayed clean
- Display subsystem did was constantly displaying correct data

What Went Wrong?

- **D Day -1** Adding an inadvertent delay caused checksum failures when reading PM2.5 streaming data through UART
- PM2.5 streams data close to every second. If UART read is not done every second, it causes checksum errors
- **D Day 9:00 AM** The relay that is between the fan and ESP32 microcontroller - protecting the ESP32 from 12V battery - gave up and ESP32 died
    - We borrowed an ESP32 at 12:00 PM and wired all the connections for our demo at 1:00 PM
    - The burnt ESP32 with its connections was left on the board (breadboard looked messy)
- USB to TTL PCB did not work - we couldn't get the exact components and the board arrived late making it difficult to debug

Next Steps

- Implement dynamic filtration capabilities with the second CO_2 sensor
- Use forced convection and create consistent air flow through the clean enclosure so that stale air is removed even more consistently
- Extend capabilities to PM1.0 particles

Questions?

The Grainger College of Engineering

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN