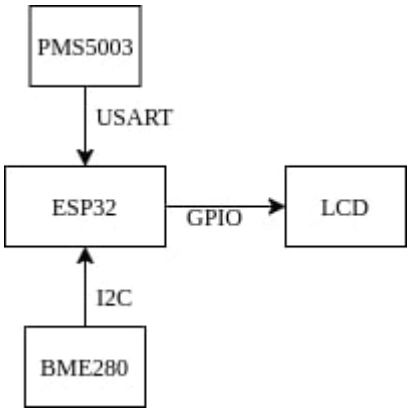
Smart air quality sensor

Introduction

This project aims to implement a smart air quality sensor capable of making predictions regarding air quality for the day. The purpose of it is to give recommendations on outdoor activities based off air quality, as well as display real-time sensor data to the user. This device is of use to those wary of their health and well-being who may wish to do various outdoor activities, such as running or going for a walk around the block.

General description

The device actively polls for air quality data from its sensors and displays it on its LCD screen. It stores hourly sensor data for up to 24 hours and makes a prediction of air quality for the following day on request. The predictive analysis is accessible to the user via an HTTP server.



Hardware Design

Components

The device consists of the following hardware components:

Component	Details	Interface	Source
ESP32	module with integrated Wi-Fi	-	emag
PMS5003	particulate matter (PM1, PM2.5, PM10) sensor	USART	emag
BME280	temperature, atmospheric pressure and humidity sensor	I2C	emag

LCD2004	4×20 character display	GPIO	emag

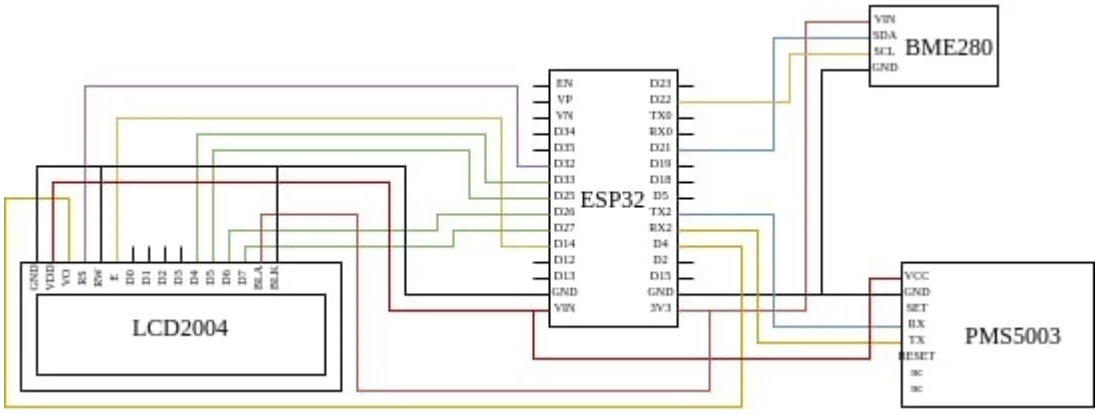
This choice of components was a result of various considerations pertaining to the project's specifications. The ESP32 is a powerful, dual core chip better fit for predictive analytics in contrast with its counterparts. It features on-board Wi-Fi, crucial for remotely sharing predictive data. The BME280 and PMS5003 sensors were picked for availability reasons. A large 4×20 character display was chosen over its more common 2×16 alternative to better fit all the sensor data.

The BME280 module interfaces with the ESP32 board using its default I2C SCL and SDA pins, which need not be set in software. Its power is supplied by the board using the 3V3 pin, according to specification.

The PMS5003 sensor communicates with the ESP32 module via the serial interface. The RX and TX pins of the sensor are connected to the TX2 and RX2 pins of the module, respectively. The RX2 and TX2 pins were chosen over the RX0 and TX0 for spacial convenience. As per the specification, it is powered via the VIN pin of the ESP32 module which, while powered via USB, outputs 5V. This voltage is necessary for the sensor to function properly.

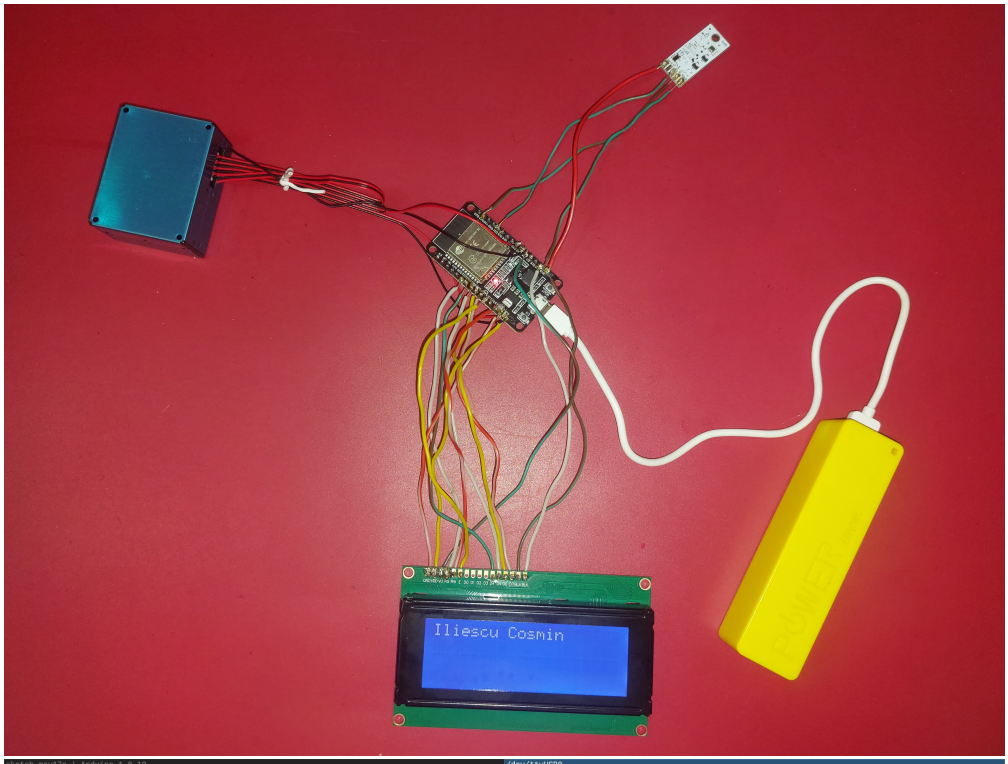
The LCD2004 display communicates with the ESP32 module via GPIO. The display can function under normal conditions while only using four data pins (D4:7). These four pins are connected to GPIO33, GPIO25, GPIO26, GPIO27, respectively for spacial convenience. The E (enable) and RS (cursor) pins of the LCD are connected to the GPIO14 and GPIO32 pins of the ESP32 module, respectively. The VO pin, which pertains to contrast control is also connected to a GPIO pin (GPIO4) which is capable of PWM. This enables contrast control from software, without the use of a potentiometer. The RW pin is pulled low via a connection to GND, which allows writing to the display. The display is powered via the VIN port of the ESP32 module, which outputs 5V while the board is powered using USB. The backlight, however, is undervolted to 3V3 (as opposed to its specified upper limit of 5V). This decision was made in consideration of power usage, despite the dimness.

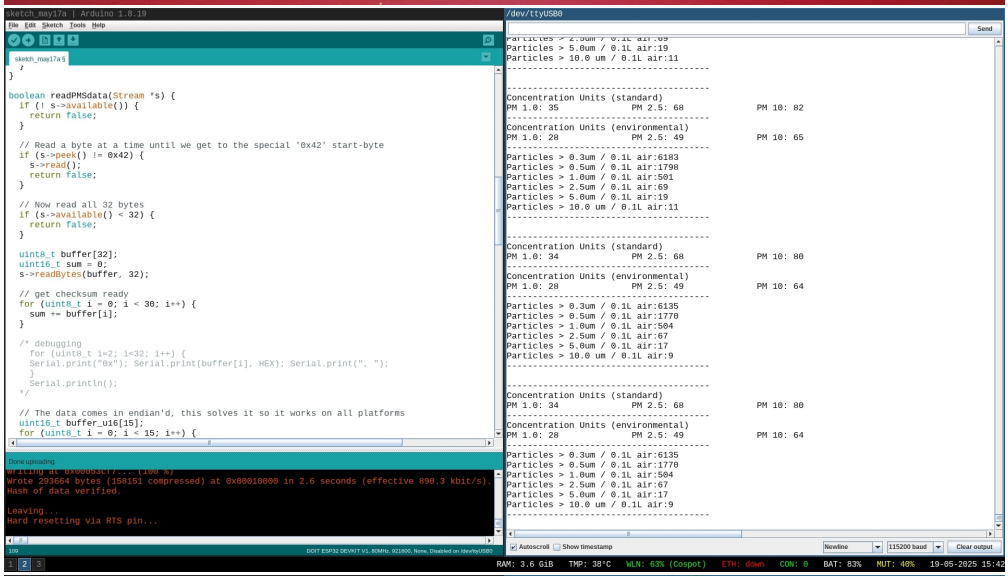
Circuit diagram



Proof of work

LCD output and data readings from PMS5003 and BME280 respectively:





```
/*
}

boolean readPMSdata(Stream *s) {
  if (! s->available()) {
    return false;
  }

  // Read a byte at a time until we get to the special '0x42' start-byte
  if (s->peek() != 0x42) {
    s->read();
    return false;
  }

  // Now read all 32 bytes
  if (s->available() < 32) {
    return false;
  }

  uint8_t buffer[32];
  uint16_t sum = 0;
  s->readBytes(buffer, 32);

  // get checksum ready
  for (uint8_t i = 0; i < 30; i++) {
    sum += buffer[i];
  }

  /* debugging
    for (uint8_t i=2; i<32; i++) {
    Serial.print("0x"); Serial.print(buffer[i], HEX); Serial.print(", ");
    }
    Serial.println();
  */

  // The data comes in endian'd, this solves it so it works on all platforms
  uint16_t buffer_u16[15];
  for (uint8_t i = 0; i < 15; i++) {
```

```
Done uploading.
Writing at 0x00053c17... (100 %)
Wrote 293664 bytes (158151 compressed) at 0x00010000 in 2.6 seconds (effective 890.3 kbit/s).
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

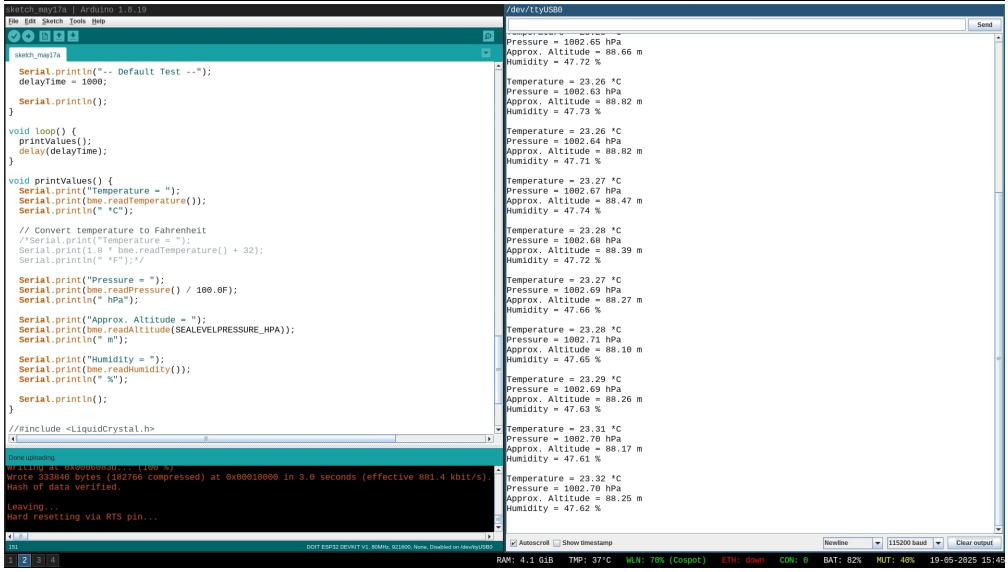
```
/dev/ttyUSB0
Particles > 5.0um / 0.1L air:19
Particles > 10.0 um / 0.1L air:11
---------------------------------------

---------------------------------------
Concentration Units (standard)
PM 1.0: 35                PM 2.5: 68              PM 10: 82
---------------------------------------
Concentration Units (environmental)
PM 1.0: 28                PM 2.5: 49              PM 10: 65
---------------------------------------
Particles > 0.3um / 0.1L air:6103
Particles > 0.5um / 0.1L air:1798
Particles > 1.0um / 0.1L air:501
Particles > 2.5um / 0.1L air:69
Particles > 5.0um / 0.1L air:19
Particles > 10.0 um / 0.1L air:11
---------------------------------------

---------------------------------------
Concentration Units (standard)
PM 1.0: 34                PM 2.5: 68              PM 10: 80
---------------------------------------
Concentration Units (environmental)
PM 1.0: 28                PM 2.5: 49              PM 10: 64
---------------------------------------
Particles > 0.3um / 0.1L air:6135
Particles > 0.5um / 0.1L air:1770
Particles > 1.0um / 0.1L air:504
Particles > 2.5um / 0.1L air:67
Particles > 5.0um / 0.1L air:17
Particles > 10.0 um / 0.1L air:9
---------------------------------------

---------------------------------------
Concentration Units (standard)
PM 1.0: 34                PM 2.5: 68              PM 10: 80
---------------------------------------
Concentration Units (environmental)
PM 1.0: 28                PM 2.5: 49              PM 10: 64
---------------------------------------
Particles > 0.3um / 0.1L air:6135
Particles > 0.5um / 0.1L air:1770
Particles > 1.0um / 0.1L air:504
Particles > 2.5um / 0.1L air:67
Particles > 5.0um / 0.1L air:17
Particles > 10.0 um / 0.1L air:9
---------------------------------------
```

```
Autoscroll  Show timestamp  Newline  115200 baud  Clear output
RAM: 3.6 GiB  TMP: 38°C  WLN: 63% (Cospot)  ETH: down  CON: 0  BAT: 83%  MUT: 40%  19-05-2025 15:42
```



```
  Serial.println("-- Default Test --");
  delayTime = 1000;

  Serial.println();
}

void loop() {
  printValues();
  delay(delayTime);
}

void printValues() {
  Serial.print("Temperature = ");
  Serial.print(bme.readTemperature());
  Serial.println(" *C");

  // Convert temperature to Fahrenheit
  /*Serial.print("Temperature = ");
  Serial.print(1.8 * bme.readTemperature() + 32);
  Serial.println(" *F");*/

  Serial.print("Pressure = ");
  Serial.print(bme.readPressure() / 100.0F);
  Serial.println(" hPa");

  Serial.print("Approx. Altitude = ");
  Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
  Serial.println(" m");

  Serial.print("Humidity = ");
  Serial.print(bme.readHumidity());
  Serial.println(" %");

  Serial.println();
}

//#include <LiquidCrystal.h>
```

```
Done uploading.
Writing at 0x00060d3d... (100 %)
Wrote 333840 bytes (182766 compressed) at 0x00010000 in 3.0 seconds (effective 881.4 kbit/s).
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

```
/dev/ttyUSB0
Pressure = 1002.65 hPa
Approx. Altitude = 88.66 m
Humidity = 47.72 %

Temperature = 23.26 *C
Pressure = 1002.63 hPa
Approx. Altitude = 88.82 m
Humidity = 47.73 %

Temperature = 23.26 *C
Pressure = 1002.64 hPa
Approx. Altitude = 88.82 m
Humidity = 47.71 %

Temperature = 23.27 *C
Pressure = 1002.67 hPa
Approx. Altitude = 88.47 m
Humidity = 47.74 %

Temperature = 23.28 *C
Pressure = 1002.68 hPa
Approx. Altitude = 88.39 m
Humidity = 47.72 %

Temperature = 23.27 *C
Pressure = 1002.69 hPa
Approx. Altitude = 88.27 m
Humidity = 47.66 %

Temperature = 23.28 *C
Pressure = 1002.71 hPa
Approx. Altitude = 88.10 m
Humidity = 47.65 %

Temperature = 23.29 *C
Pressure = 1002.69 hPa
Approx. Altitude = 88.26 m
Humidity = 47.63 %

Temperature = 23.31 *C
Pressure = 1002.70 hPa
Approx. Altitude = 88.17 m
Humidity = 47.61 %

Temperature = 23.32 *C
Pressure = 1002.70 hPa
Approx. Altitude = 88.25 m
Humidity = 47.62 %
```

```
Autoscroll  Show timestamp  Newline  115200 baud  Clear output
RAM: 4.1 GiB  TMP: 37°C  WLN: 70% (Cospot)  ETH: down  CON: 0  BAT: 82%  MUT: 40%  19-05-2025 15:45
```

Software Design

The program running on the microcontroller was developed and uploaded using ESP-IDF - Espressif's official IoT Development Framework for the ESP32. The framework provides wrappers and abstractions for various features supported by the ESP32 family of microcontrollers. These streamline development and translate focus onto actual functionality. The following abstractions were used during development:

- LED Control: peripheral library for PWM control targeted towards, but not limited to LEDs. This is used for contrast control on the LCD display, as the hardware design does not feature a potentiometer for this use case.
- Wi-Fi driver for establishing wireless connections to send predictive data over. Additionally, the event handling library was used to gracefully handle connection establishment.
- UART library for interfacing with the PMS5003 particulate matter sensor. The driver for this sensor is self-written, in accordance with its datasheet. The payload regularly sent by the sensor consists of 32 bytes - 2 start bytes (0x42 and 0x4D), 2 bytes for the data length (0x00 and 0x1C if not tructated), 26 bytes of measured data and 2 last bytes for the payload's checksum. Of these, only bytes 10 through 15 are of use, covering PM2.5 and PM10 readings.
- HTTP Server implementation, which responds to GET requests with predictive AQI data for the next hour.

Aside from the libraries available inside ESP-IDF, the following third-party libraries were used for peripheral communication:

- BME280 I2C driver easily integrated into the project via the ESP Component Registry. It streamlines use of I2C and BME280 sensor readings.
- HD44780 driver plucked from its parent library. This controller is compatible with various character display controllers, including the one present in the LCD2004.

Predictions of AQI are made using a highly accurate regression model trained and deployed using the Edge Impulse platform. Edge Impulse provides tools and APIs for training and deploying machine learning models specifically on edge devices, including embedded systems. The model was trained on two years' worth of hourly readings of PM2.5 and PM10, starting from 2023 and ending in 2024. The readings were translated into an AQI following this US Environmental Protection Agency paper. The dataset was split into samples covering 24 hours, each labled as the AQI reading of hour 25. The readings are publicly available on www.calitateaer.ro.

All the previous components interact as follows:

1. A connection is established via Wi-Fi and the HTTP server is started.
2. The two sensors are the LCD are initialized.
3. The device waits for 30 seconds, while the PMS5003 sensor warms up (in accordance with its specifications)
4. Every minute, data is read from the sensors and displayed on the LCD. The BME280 should be read at least every 60 seconds, according to its specification.
5. Every hour, the data read from the PMS5003 sensor is turned into an AQI and saved in a buffer of size 24.
6. Upon receiving a GET request, the HTTP server responds with the AQI for the following 24 hours predicted using the regression model uploaded onto the microcontroller. The device should ideally run for at least 24 hours before a prediction is requested, so 24 hours of predictive data may be available in the buffer.

Results

A demonstration of the device is available at https://youtu.be/Bxe3qhWADMU.

Conclusions

Download

The software implementation is available at https://github.com/Cosminovici22/pm_project.

Journal

- 13th to 16th of May: arrival of components
- 17th and 18th of May: assembly of device components
- 23th to 25th of May: development of device software
- 28th and 29th of May: refinement and packaging of device
- 30th of May: device demonstration at PM fair

Bibliography

- LCD2004 pin summary
- PMS5003 datasheet
- ESP-IDF programming guide
- BME280 I2C driver usage examples
- AQI dataset source
- US Environmental Protection Agency technical assistance document for the reporting of daily air quality

Export to PDF

From:
http://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
http://ocw.cs.pub.ro/courses/pm/prj2025/avaduva/cosmin.iliescu

Last update: **2025/05/30 06:33**