Citizen Air Quality Sensor

Joshua Egwuatu, Kaleb Irwin, Cameron Smith, Alejandro Moore, Marvo Odds

Computer and Electrical Engineering Department at Tennessee Technological University

115 W. Tenth St., Cookeville, TN 38505

Jiegwuatu42@tntech.edu, kjirwin42@tntech.edu, casmith51@tntech.edu, aamoore42@tntech.edu, modds42@tntech.edu

I. INTRODUCTION

A. PROBLEM DEFINITION

The goal of this project is a replicable device that accepts and operates multiple air quality sensors independently with a selection of interchangeable sensor options for local data collection. Air quality is vital to maintaining life on Earth [1]. Poor air quality can result in a slew of health problems such as heart disease and lung cancer [2]. In the modern age, humanity pushes the levels of pollution in the world to unprecedented levels as compared to the pre-industrial era [3]. This pollution affects the very air that all living things rely on to survive. This perspective on the negative repercussions of polluting the air and Earth is what led to this team's proposition of a device that allows everyday citizens to see what pollutants blight their communities.

There are already services that exist to alleviate this problem, however, the services do not fulfill the solution to a high degree. The biggest of these services would be the Air Quality Index (AQI) provided by the United States Environmental Protection Agency [4]. The AQI takes in data from sensors placed systematically all across the country and in highly populated areas [5]. This data is then aggregated together to fill in any regions that were not explicitly sampled from a sensor directly [5]. The issue derives from that last sentence. Because it is not reasonable or cost-effective for these services to place sensors everywhere, the readings that are received from these sensors may not be accurate in any given place. For a community wanting a detailed view into the air quality of their chosen area, they would require more concrete sources of data. It is here where the team's proposed device comes in to provide an easy-to-use, solar-powered, easily replicable, and low-cost modular system that uses multiple sensors to gather data on different pollutants.

B. KEY DESIGN POINTS

There are a variety of specifications and constraints placed upon the project for the purposes of functionality, safety, integrity, and longevity. The team will provide a selection of tested and approved sensors for use. Common qualities to measure are carbon monoxide, lead, nitrogen oxides, ozone, particulate matter, and sulfur dioxide. The device will be designed with mostly, if not entirely, off-the-shelf purchases to lower the technical knowledge required for construction. The software should be available for free online on GitHub. To further help in ease of use, thorough documentation will be available online in GitHub for free to answer common answers. The device will be relatively cost-effective including the microcontroller, additional components, and software remaining under $300. These are all specifications given by the customer.

Constraints are also included based on engineering standards that may be associated. Given that the device will be operating in a variety of environments, it will need to be somewhat durable, so the chosen standard is IP34 to resist water from the rain from all directions and protection from small objects greater than 2.5 mm thick. The range of temperature operations will need to include 10°F to 90°F. The power source(s) will be portable and last at least 20 hours to allow for sufficient data collection. The project will use Transmission Control Protocol (TCP) and IEEE 802.11 for communication with the website server.

This design team shall follow National Electrical Code safety procedures, associated National Fire Protection Agency procedures, and abide by piracy laws. There are no intentions to break any laws established by the United States of America Government. All patents, licenses, and copyrighted material will be used under legal fair use with citations and credit when appropriate. This device will be designed to prevent risk and harm to the users, communities, and environment. The team will follow IEEE code of Ethics and common electrical concepts/standards: resistor color codes, design schematics, communication between devices, etc.

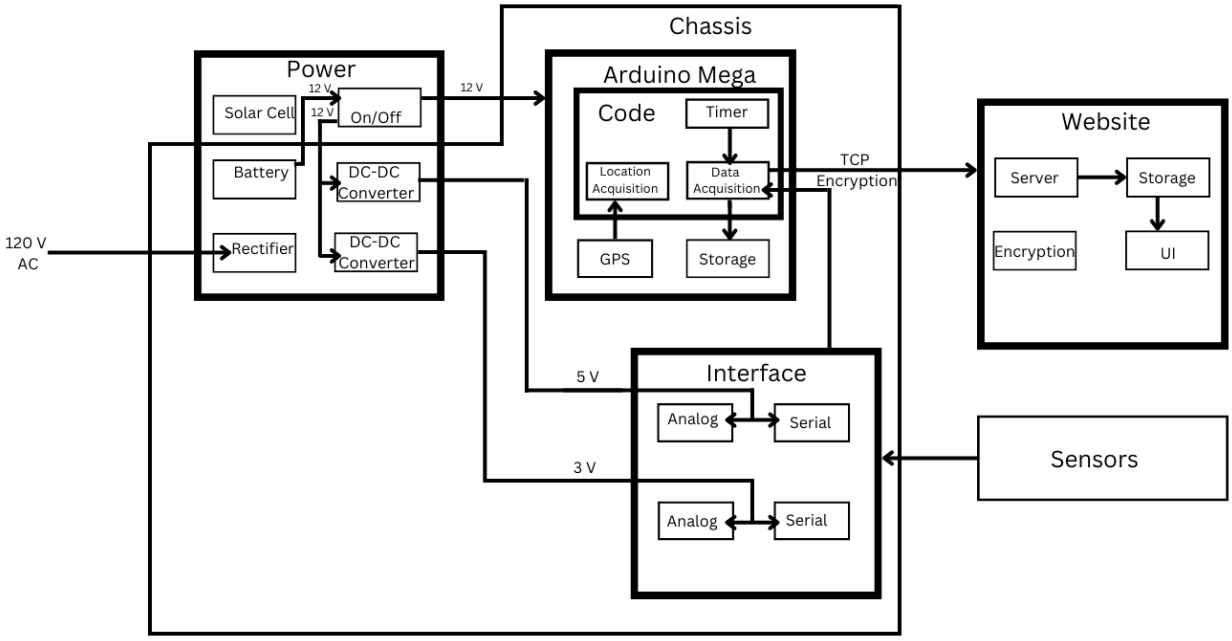

Figure 1. Block Diagram

II. POWER SYSTEMS

The power system will center around a 12V battery that will provide power to the device. A 12V battery was selected as it is fairly common and will be able to power the Arduino without needing to change the voltage as its recommended input voltage is 7-12V. The battery can be charged either by an onboard solar panel or via a 120VAC source. The battery's output will be connected to an on/off switch and then routed directly to the Arduino and to two DC-DC converters. These will convert the ~12V to 3V and 5V. Further filtering of these voltages may be required depending on what converters we use. These two voltages will be used to power the various sensors on the interface. The choice to use converters and the battery overpowering the sensors directly off the Arduino's 5V and 3.3V pins were made due to current considerations. The Arduino says it can output a maximum of 50mA and some of the sensors we found required more current than that. The power system will be tested using LTSpice as well as hands-on testing. We will look at the output voltages as well as the current capabilities of all components and ensure the Spice model shows all necessary metrics are within spec.

III. MICROCONTROLLER SYSTEMS

The microcontroller is an Arduino Mega REV 3 with 54 digital input/output pins,16 analog inputs and will be powered by a 12 volt battery. Using the Arduino Mega allows for more sensors to be added and accounts for types of sensors that might take up more pins. The timer will set a determined time interval for the Arduino to wait before acquiring and sending data gathered from the sensors to account for unstable data given by the sensors due to warmup and cooldown times. The data acquisition system will receive input from the interface and output data to two places: the storage within the arduino and to the website database. Location acquisition will set up the Arduino to be able to interface with an outside GPS module and allow the user to see the location of the data readings.

IV. INTERFACE AND PORT CONTROLS

The interface is a specified area on the chassis that is available to the user for their sensor selection. The targeted area will be able to house up to 6 different sensors at a time and will send the received data to the microcontroller. It is important that the interface should be able to take in multiple types of sensors that detect a specified pollutant within our list of common qualities.

When the device turns on, the input ports for the sensors will be excited with a specific amount of voltage to activate the sensor once plugged in by the user. From our research so far, we have encountered various sensors that require slightly different amounts of input voltage. While there are some sensors that will run on 5VDC, there is a good portion that also takes in 3VDC. Therefore, using the 2 DC-DC converters from the power block, we will connect wires to the input ports within the interface.

V. WEB AND WIRELESS CONNECTIONS

Wireless acquisition of the data is not technically necessary, but it adds an undeniably large boost to the "quality of life" when using the device. Having to physically retrieve each and every device that was deployed would be very tedious as opposed to retrieving them only when the user is ready to move them. The components of the "Web" block include: server, storage, user interface (UI), encryption, and Transmission Control Protocol (TCP).

A. SERVER

The server has two possible solutions: a website hosted by a third party service or a locally hosted server machine. Hosting the server from a third party would be easier and allow for quick setup. Doing this solution trades the cost of buying and maintaining a local machine for rent fees. Writing and building a server from scratch would allow for total control of the server's protocols, be cheaper in the long run, and secure the future of the project as it would always be available whereas a third party could stop providing the service. This option would cost more upfront and potentially require maintenance from time to time. For the purposes of this project, writing a simple server from scratch as a proof of concept seems to be the better option.

The first job of the server will be to wait for a signal from the device. The server will need to verify the connection to the device and ensure a secure line of contact. This can be done by implementing TCP as the method of communication. TCP is a common Internet Protocol for transmitting data across small to medium distances [6]. For the purposes of this project, it will be used to create its tunnel and help ensure a secure connection. Second, the server will commence a "handshake" with the device to set up the tunnel for TCP. Third, the connection has been established and the server will now ask for the data. The data should come encrypted. What this means is that the true value and meaning of the data is being masked or scrambled for integrity and security protection of the data itself. With the nature of the Internet, there is always a security risk in that a hacker could discover the server and attempt to get in [7]. The server will wait for the data to be fully sent. Once the data is fully sent, the server will decrypt the data to find the true value of the data. Fourth, the server will format the data both for storage and for the user to read from later. Fifth, the server will kill the connection and deconstruct the tunnel. The server is finally done with the data retrieval and will await for another device to submit data. Data will only be submitted by one device at a time. Multiple data transfer requests will be handled by a queuing system.

The server itself will be written in C++ and utilize Windows 10 WinSock to transfer packets (data). Microsoft Windows is the most popular operating system and provides its own library for data transfer through the Internet [8][9]. Given that most personal computers used at home are running Windows, using a native library will allow for a large target audience. Users of Linux, MacOS, or another operating system will not be able to host their own server. So, personally running the server for each user is not feasible, but Windows users may opt to do so. An ideal end result would be that the server is running 24 hours a day and is available to all users as a default option for those who do not or cannot host a private server. Linux, MacOS, and other operating systems may still communicate with the default server as normal. The code may be available alongside the software and documentation on GitHub.

B. STORAGE

The data will need to be stored on the server host machine. This means that it will be a requirement for the server host machine to possess adequate storage space for the server code to store all of the collected data. In terms of hardware, hard disk drives and solid state drives would work optimally. Floppy disks and flash drives could work as well but would be a rare case to be handled by the native operating system of Windows. Once the data has been gathered, the server will read in and format the data before storing it in memory waiting for the user to collect when needed. The format has yet to be determined but will be simple to leave room for potential customization or improvements later.

C. USER INTERFACE

The UI will be what the user sees when attempting to read the data that was collected from the device. It will be bare bones. The purpose of the UI on the server is simply to get the data into the hands of the user in an easy manner. The data should already be formatted in a way that makes sense, so the user will only need to see what is being presented and do whatever is wished with it. The format will likely be a similar setup to Google Sheets or Microsoft Excel with generic names for subjects. It will be up to the user to interpret the data. The UI itself will be a portion of code within the server that provides a palatable visual for the user to overview the data.

REFERENCES

[1] Kim Rutledge, Melissa McDaniel, Santani Teng, Hilary Hall, Tara Ramroop, Erin Sprout, Jeff Hunt,

Diane Boudreau, Hilary Costa, National Geographic, “Air Pollution”, https://education.nationalgeographic.org/resource/air-pollution. (Accessed on 9-21-22).

[2] Minnesota Pollution Control Agency, “Air Quality and Health,” https://www.pca.state.mn.us/air-water-land-climate/air-quality-and-health. (Accessed on 9-21-22).

[3] United States Environmental Protection Agency, “History Of Air Pollution,” https://www.epa.gov/air-research/history-air-pollution. (Accessed on 10-16-22).

[4] Air Quality Life Index, Constructive Inc, “Pollution Facts”, https://aqli.epic.uchicago.edu/pollution-facts/. (Accessed on 9-22-22).

[5] AirNow, Home of the U.S. Air Quality Index, “Air Quality Index (AQI) Basics”, https://www.airnow.gov/aqi/aqi-basics/. (Accessed on 9-21-22).

[6] Defense Advanced Research Projects Agency, “Transmission Control Protocol,” https://datatracker.ietf.org/doc/html/rfc793. (Accessed on 9-21-22).

[7] DeVry University, “A Constant Threat: The Impact of Cyber Hacks,” https://www.devry.edu/blog/threat-cyber-security-hacking-infographic.html. (Accessed on 9-19-22).

[8] Western Governors University, “5 Most Popular Operating Systems,” https://www.wgu.edu/blog/5-most-popular-operating-systems1910.html. (Accessed on 9-21-22).

[9] Uri Raz, Windows Socket Programming Frequently Asked Questions, “Winsock-fag,” http://www.faqs.org/faqs/windows/winsock-faq/. (Accessed on 9-21-22).

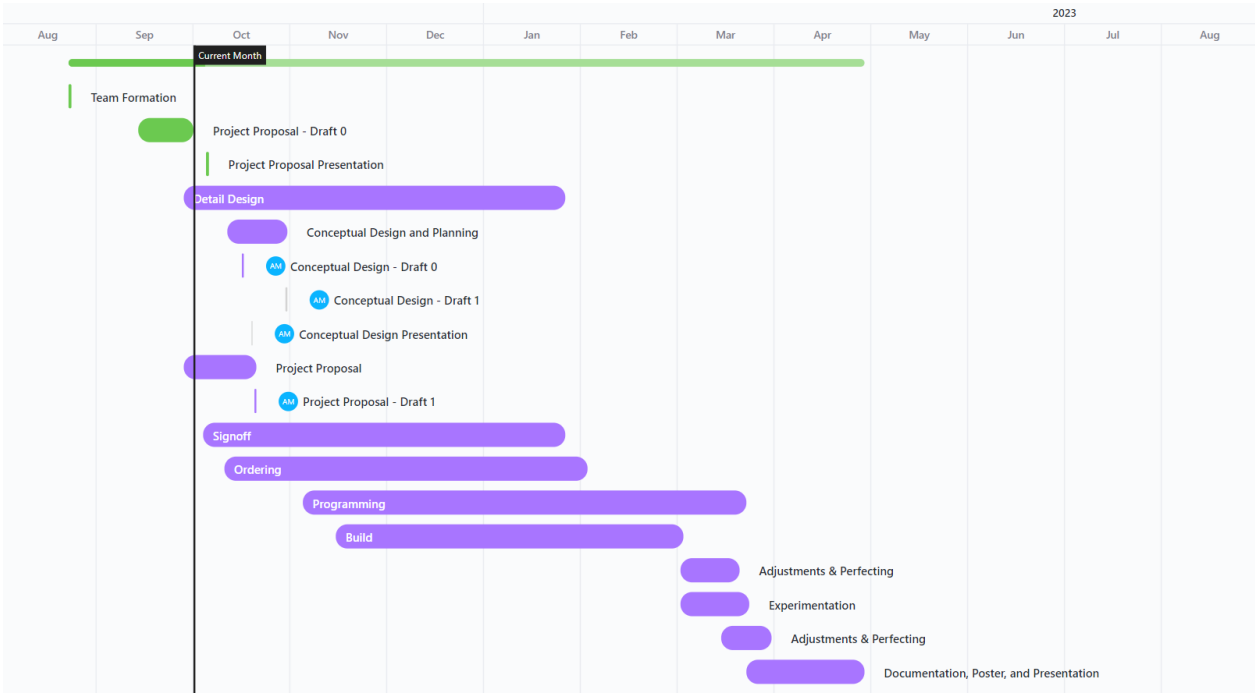

Figure 2. Gantt Chart