ISE REU Project Report

IoT Air Quality Monitoring for Health

Elizabeth T. Ramos

Faculty Advisor: Professor Richard Sowers

Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign™, Urbana, IL 61801

Table of Contents

I: Introduction

The objective of this project is to integrate a Raspberry Pi with several indoor air quality sensors and implement an automatic cloud logging system by drawing on existing Internet of Things infrastructure. Through the integration of the sensors and their respective software with the Raspberry Pi, it is possible for various brands of sensor data to be visualized on one platform.

Two sensors, a Lascar EL-USB-CO300 Carbon Monoxide Sensor and an Onset HOBO Temperature/Relative Humidity/2 External Channel Data Logger, were chosen for this project. The Raspberry Pi 3 – Model B was the model of Raspberry Pi used.

Figure 1: Lascar EL-USB-CO300

Figure 2: HOBO Temperature/Relative Humidity Sensor

Figure 3: Raspberry Pi 3 – Model B

The carbon monoxide sensor uses EasyLog™ software for data retrieval while the temperature and relative humidity probe uses HOBOware™ software. The first step of the project, which involved installing the sensors' software on the Raspberry Pi to export the air quality data, faced many roadblocks. Once that problem was overcome, the process of exporting data from each software was automated.

Over this project I created a fair amount of code, which all can be found at the link below.

https://gitlab.engr.illinois.edu/iot-air/sensing

II: A Series of Road Blocks

Initially, the Raspberry Pi was chosen as the hub for sensor data exportation because it was small, inexpensive, and had the qualities of a computer which allowed for ease of Python program implementation. However, it was not understood at the time that the Raspberry Pi's software infrastructure does not support EXE files, the executable files needed to launch a program on a standard Windows PC. This was problematic because the EasyLog™ and

HOBOware™ software both come in an EXE format. Several attempts were made at trying to work around the fact that the Raspberry Pi does not support this file type.

II.1: Raspbian Operating System

The first problem was that the software which allowed access to the data within the sensor was not compatible with the Raspbian operating system on the Raspberry Pi. The Raspberry Pi is an Advanced RISC (Reduced Instruction Set Computing) Machine (ARM) computer. Because it is an ARM computer, the Raspberry Pi doesn't have the software capabilities necessary to run the HOBOware™ or EasyLog™ software. The manufactures, unfortunately, do not support EasyLog™ or HOBOware™ software in a format compatible with the Raspberry Pi.

II.2: "Hacking" the Sensors

After realizing that the Raspberry Pi does not support the sensors' software, an attempt was made to circumvent the use of the manufactures' software using the PyUSB, LibUSB, and LibHID libraries within Python. However, with limited resources and uncertainty of commands, this route was quickly abandoned as it was deemed too time intensive of an endeavor.

II.3: Windows Emulators

With the knowledge that the sensors' software was designed to run on a Windows operating system, the next route was searching for a suitable Windows Emulator. Wine, a common emulator-like program, is not directly compatible with the Raspberry Pi. However, a program called ExaGear Desktop allowed for Wine, and consequently, the EasyLog™ software to be installed. However, because of limitations associated with the Raspberry Pi USB ports, the EasyLog™ software was unable to detect that the sensor was plugged in. Additionally, the

HOBOware™ software was unable to be used with Wine because it requires Java 8, which is not directly compatible with the Raspbian operating system. Because of the nonfunctional USB ports, unreliability, and inability to easily install the HOBOware™ software, the route of using an emulator to solve the problem was abandoned.

III: Tunneling Through Raspberry Pi

The final idea to extract the data, which proved to be fruitful, was tunneling through the Raspberry Pi to the sensors. This involved turning the Raspberry Pi into a wireless USB hub and exporting the sensor data through a Windows machine with the associated sensor software.

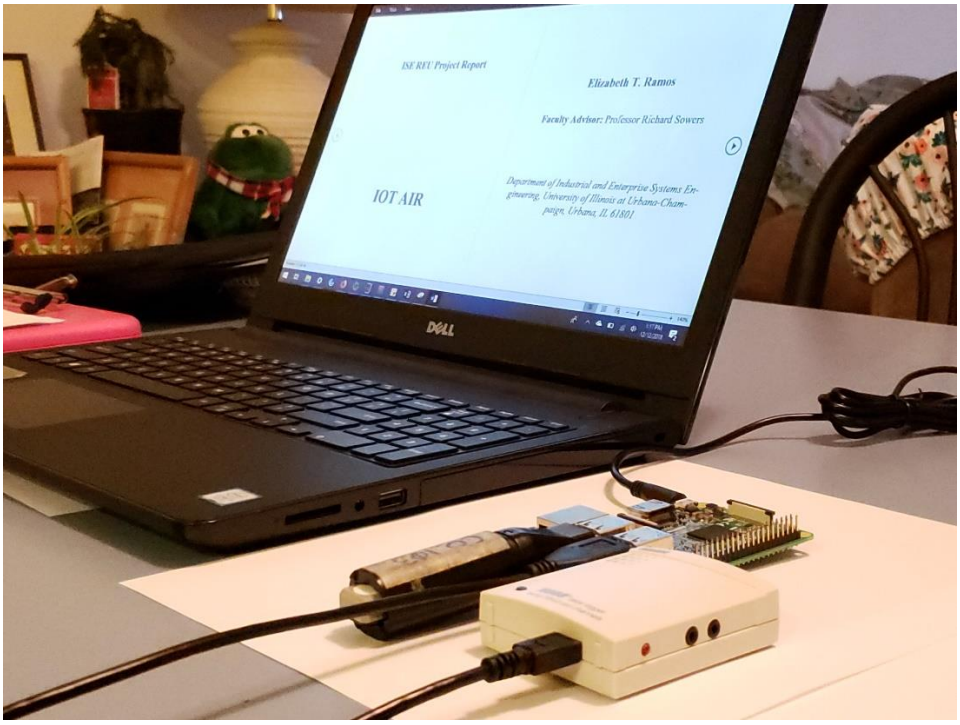

Figure 4: Raspberry Pi and PC Setup

Turning the Raspberry Pi into a Wireless USB hub was achieved through a software called VirtualHere. With access to the internet, the software allows the traditional computer to access the ports on the Raspberry Pi as if they were attached directly to the computer. While this

project has only utilized a traditional computer as the client, in the future we plan to shift over to the use of a cloud client.

Although all components of the VirtualHere software work perfectly on both Windows and the Raspberry Pi, efforts to launch the program on the Raspberry Pi (which requires a series of codes to be entered into the command line) upon startup proved unfruitful. One step toward efficiency, however, was achieved by writing a script which would execute all the commands in in their corresponding order so that they did not need to be typed in by hand.

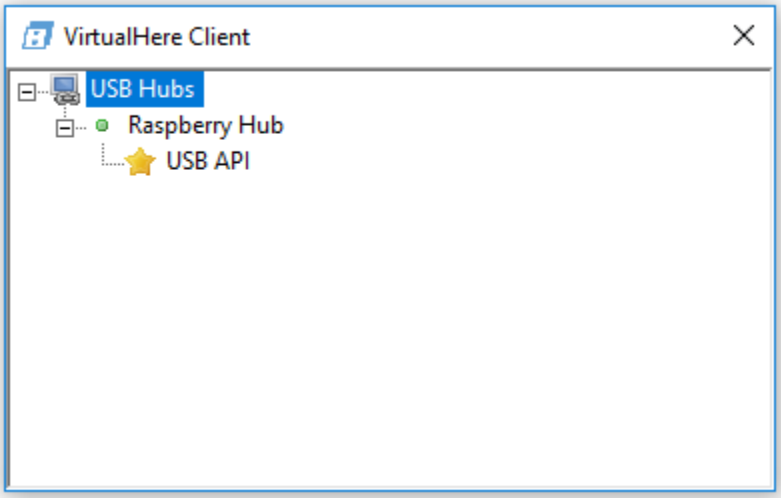

Figure 5: VirtualHere Client Application

Once the commands on the Raspberry Pi's end were executed, a screen would appear on the PC showing the Raspberry Pi's USB ports. This is illustrated in Figure 5.

IV: Graphical User Interface Automation

After it was possible to extract the data from the probes from a remote location, the next step was to automate the process of extracting the data in the form of a CSV file. This would allow for periodic extractions of data, which could then be fed into the cloud platform for real-time data visualization. To automate the process, the PyAutoGUI library within Python was

used. This required making a detailed flowchart of where to click or type within each user interface to export the data. With Python commands, the flowchart was turned into a list of commands the computer could execute by itself. The product were two Python programs which could open each respective software, export the data, and close the software.

V: Conclusion

Despite many roadblocks early on, a great deal of progress was made over the course of the semester. Now that data retrieval is automated, work can commence on building the platform that the data will be visualized on. Once that is complete, tests of the device can begin in areas of interest such as a kitchen, dorm room, or classroom space. With a fully functional design, we hope that a device of this nature will be of use to individuals, such as physicians, to interpret data within their realm of expertise, draw conclusions and make informed decisions.

Acknowledgements

I would like to thank the Department of Industrial and Systems Enterprise Engineering for the funding to conduct this research. I would also like to thank my faculty advisor, Professor Richard Sowers, as well as Paul Francisco for their help, guidance, and encouragement on this project. Finally, I would like to thank my older brother, Evan Ramos, for lending his ear and experience in software problem solving.

References

EasyLog™ USB. (n.d.). Retrieved December 13, 2018, from Lascar Electronics website: https://www.lascarelectronics.com/software/EasyLog-software/EasyLog-usb/

ExaGear desktop. (n.d.). Retrieved December 13, 2018, from eltechs website: https://eltechs.com/product/exagear-desktop/

Freitas, T. (Producer). (2016). *VirtualHere installation for* [Video file]. Retrieved from https://www.youtube.com/watch?v=OPlKNnNq4hQ

HOBOware™. (n.d.). Retrieved December 13, 2018, from Onset website: https://www.onsetcomp.com/products/software/HOBOware

Raspberry Pi. (n.d.). Retrieved December 13, 2018, from https://www.raspberrypi.org/

VirtualHere. (n.d.). Retrieved December 13, 2018, from https://www.virtualhere.com/home

Welcome to PyAutoGUI's documentation! (n.d.). Retrieved December 13, 2018, from PyAutoGUI website: https://pyautogui.readthedocs.io/en/latest/