

Data Pull Automation Proposal

Automation

David Healy, Jeff Underhill, Kathleen Errington, Marcus Chase and Zachary Thoroughgood | NHDES PA Monitoring Project | 07/12/2022

# Introduction

We are tasked as a team to tap a new resource that is up and coming for data pulling not only in New Hampshire; but also, other states and all over the world. This technology being a monitor that is so small it can fit in your hand but is seemingly inexpensive when put up against a full-fledged data collecting field station. If we can correctly pull data and collect ten-minute averages; we can expand the data collection across the state at a cheaper cost and, it is a great way to interact with the public in a collaboration of resources for the greater good. So how far have we come so far?

* With the help from our good friends at Keene State College (JC Woodward and Dr. Traviss); we took in their current model for pulling data from the Purple Air Monitors through their two python scripts of (Fetch.py and MapMaker.py) and formulated a plan for our own model.
* Currently, we have a program called Pull.py that can pull instantaneously the data given back from a call to the Purple Air API servers for every unit that is listed in the configuration file (an .ini file) and reports all the data back through a .csv file that is stored. The program also logs everything during execution; every execution with a time stamp to help keep track of any errors. This program can also recognize whether we have a .csv file created for the day or not to either append to the existing or create a new one for the beginning of the day.

## Where we’re at?

We are at the final stages of our first deliverable, which is a working automation program that will be set up to run for our first testing period inside of our contained testing environment. This program would be tested to run for an overnight at first and if all goes accordingly, it will go for a week straight of pulling to analyze the effectiveness of the program to debug and improve it to continue our steps in pulling the data from the Purple Air Monitors effectively/efficiently.

## What we need Assistance With?

Well, we have realized that the model provided by JC and Dr. Traviss was continuously running their pulling code until a breakage occurred inside a *while True* loop; which was verified by JC via email:

“*You are correct on every point.  That is the main loop in the program that makes a map then sleeps for ten minutes until it's time to make another one.  There is an obsolete PC not usable for much else that is dedicated to running this one program,”* (Email from JC Woodward on 7/11/2022 at 5:37 P.M.)

Given this verified information there are a few routes we could take as alternative matrices for automating our Pull.py.

# Automation Details

Now we will investigate a few ideas/alternatives to automating the Pull.py to pull our much-needed data from the Purple Air Monitors.

## Alternative Matrices

* Purchase a raspberry pi which is inexpensive at $50 or less from Amazon.com or [Buy a Raspberry Pi – Raspberry Pi](https://www.raspberrypi.com/products/).

Pros:

* Cheap device/affordable
* Continuously run the program without fear of damaging device
* It is small and can fit in any environment

Cons:

* Does not have a lot of storage
* Linux environment which means we would have a while True loop as well
* If it goes down; it would need a manual restart of the program
* Task Scheduler on the Modeling Computer

Pros:

* We already have both
* Task Scheduler can wake up the computer to run automation of a program
* The computer can also be put back to sleep using Task Scheduler
* Modeling computer has lots of storage
* We will have an abundance of control as to when the program runs and how frequently
* Eliminates a lot of human error
* Minimal Servicing

Cons:

* We will need to consider unexpected events such as power outages
* The computer may never sleep unless an unexpected event occurs
* Continuous While Loop on the Modeling Computer

Pros:

* Eliminates a lot of human error
* Continuously grabs our data

Cons:

* Life expectancy of the computer severely decreases
* Puts it at risk for unexpected events
* Would need a manual restart in unexpected events
* Possible loss of data when no one is here to service it
* Hybrid of Task Scheduler on the Modeling Computer

Pros:

* Gives the modeling computer a business days rest
* Allows for a seemingly errorless conversion of pulling
* Still eliminates a lot of human error
* Allows for a hybrid of both full automation and manual conversion
* Minimal service from overlapping and taking over the ongoing shift
* Already have the resources

Cons:

* Human error (forgetting)
* Holidays/Weekends (However the program will run automatically still with breaks through going down to sleep mode until being awoken again)

## Best Choice of Solution

Arguably, the best choice would be the hybrid approach which would consist of allowing the modeling computer to run automatically overnight every ten minutes by being awoken from sleep, and then being put back to sleep until the next morning when a team member can time the overlap of execution. Meaning once it runs, they have a ten-minute window to toggle disabling the task scheduler on the modeling computer and shutting it down to give it some rest and enabling the preloaded task scheduled automations (which would be exact duplicates as the modeling machine) to run for the workday on their own desktop in the background. Thus, giving the modeling computer a chance to power down and reboot itself as well as putting the whole computer to rest in order to get the most out of its life expectancy. This also means that the automation continues while the team member continues to work an ordinary day.

## Execution of the Chosen Solution

Our focus as a team is to create a testing environment that everyone understands with a guide that is simple and thorough enough that any person with a basic knowledge of a computer can understand and mimic when given the tools (programs). Currently, as of 07/13/2022, team member Zach Thoroughgood has a developed testing environment that is in its later stages of the initial phases on his workplace desktop. The testing environment is composed inside of the local C: drive inside of his user account and has the main components necessary to run the environment. Those being the *.ini* file that tells the program what units to pull, keywords from the retrieved *.json* files, a batch file to archive the *.csv* file from the day prior and a batch file to automate the *Pull.py* program which is the main source for this project. This is all pieced together with a Visual Basic Script that allows the automation to run like a ghost that way the user never knows it is there. The Visual Basic Script is simply called on every ten minutes throughout the day via the Task Scheduler which allows for the user to still do their work while this pulling happens in the back.

# Where We Stand

## What We Know As a Team

We realize that eventually this will be a fully automatic process through the aid of a server, and this is the major go for broke moment for the team. Our plans are to adapt a model that Keene State College has already graciously provided for us along with their help, (we thank you both so much JC Woodward and Dr. Traviss), but with our own NHDES twist. The capabilities that the NHDES has is more than what they had to work with, and this program needed to be based off the NHDES capabilities/workplace environment. Considering, the cost of each machine and the time it takes to get new equipment into this atmosphere; the realization is that there is not much room for error in terms of shaving life expectancy off any piece of equipment. The current model that NHDES stands as, shows that it is a Microsoft Workshop. After careful consideration of the workplace’s Operating System usage, how it operates, and careful research; the testing environment is setup accordingly to accommodate a Microsoft Server eventually. For now, the team is asking for the continuous support of DoIT Helpdesk because there are hoops that we’ve jumped through already and there are plenty more to hurdle to keep this positive motion continuing towards a greater goal of gathering larger pools of data across New Hampshire in hopes that other states can mimic the team’s collective hard work to promote better health and living across the United States of America as well as around the world. This is all contingent upon the support we receive.

## What Do We Need From DoIT Helpdesk?

* Bigger rights in terms of Batch Files

*Why?*

Without this, we cannot further test our created testing environment during the night when no one is here. A lot of our data pulling will be when everyone is away, and the process needs to be reliable or else we lose data. We also need the rights to be able to move files and run the program inside of the S: drive because this will allow for a universal hand to grab the data at any time. Meaning, this process is not reliant on one single team member and can be accessed within our network by any supporting team member.

* Need a folder only accessible by the team and DoIT Helpdesk to store our S: drive testing environment.

*Why?*

If everyone can access our experimental testing environment; accidents can occur. Someone curious may accidentally open a .*json* that is currently being looked at which can block the permission from our program. This means we lose that data for that unit or even worse; if they have the *.csv* file open while the program tries to add all the data, it will lose that ten-minute pull.

* Need to know the mail server information.

*Why?*

There will be eventually an additional piece inside of the Pull.py that will alert certain team members and/or the whole team to when a unit is reading high, missing crucial data, or could even be offline. The team will need to know the Microsoft Exchange Server Information to code in an email alert system that will automatically alert us. This will help in tracking down problems with units and allowing for us to lose minimal amounts of data.

# Closing

## Conclusion

This project means a lot to every single member of this team. For most of us this means a better and efficient way of getting data without wasting a lot of time. For all of us, this means that we can make a major impact in terms of alerting the community of potential bad air days in concentrated areas and allows us to better our relationship with the public. We ask that the DoIT Help Desk continues to be mindful, patient, and supportive of this Purple Air Monitoring Project because together with your assistance; we can reach a milestone moment for Purple Air Monitoring which is a developed System that others can take away from. If any member of DoIT Help Desk wants to see the progress made so far, feel free to reach out regarding a meeting and a demonstration can be arranged to show our efforts thus far as a whole. Otherwise, we would greatly appreciate the careful consideration into the three major hurdles we have thus far. To conclude, there will be more hurdles to come as the whole process goal is to create an automation program that updates an html page for the communities in New Hampshire to view. We appreciate everything you have done for us thus far and are confident that you can clear these hurdles, so our project can continue. Thank you, DoIT Help Desk staff.