**Microphone Array**

**USER MANUAL**

CSE 453 SPRING 2016

Thomas Schouten

Korben Grover

Arunan Bala Krishnan



**Table of Contents**

**Getting Started**……………………………………………………………3

Hardware needed

Software Involved

**Operations**……………………………………………………………………4

Setup

Walkthrough

**Extra Information** …………………………………………………………5

Data formats

Troubleshooting

**Exit System** …………………………………………………….…………….5

# **Getting Started**

1. In order to run this Project, the microphone array must be assembled first:

* Ensure that all microphones are properly in place. For now, there are only three in the lower corners.
* The microcontroller should be wired into the array appropriately.
  + Refer to the designl document for wiring into the microcontroller.
* A laptop with a free USB port and power to last through data collection time is needed as well for the array to operate, make sure this is attached to the microcontroller by the USB cord.

1. Now that the array is properly set up, the software involved in this is:

* Arduino code
* The microcontroller should be preloaded with this regardless, so if you never overwrite then it should be fine. If needed, one can re-upload the included Arduino file to the microcontroller.
* Four python files are needed for this project. They are:

-readClient.py

-analyzeClient.py

-local.py

-timeAnalysis.py

* The user should only ever have to interact with the two Client codes. The other python files must be in the same local folder, and can otherwise be safely ignored.

It is worth noting that a python environment is needed to run the above files. One library is required beyond the standard, PySerial. Make sure to have everything available, or the microphone array will be unable to be accessed.

With everything described here, operation of the Array can begin.

# **Operation**

* First ensure that the system is set up as in the requirements section above. Once everything is ready, proceed to follow the steps below:

1. Run the program “readClient.py”. This should auto-detect the connected microcontroller, and establish a connection to it.
2. Reset the microcontroller. Failure to reset the controller after first running readClient tends to result in a read failure later on. Doing this step properly syncs the readClient program with the program running on the microcontroller
3. Enter the amount of time you want the microphone array to run. This value is interpreted in seconds. For now, enter a small value to ensure that everything is in working order, before entering a larger value. This should function for times up to about an hour, after which function of the array is undefined due to a timer overflow.
4. Wait while the program reads. This is when the array is performing its operation. Data read during this time is stored into a text file in the root folder, called “dataLog.txt”. This contains the microphone data for each microphone, comma separated and each cycle on a new line. When the read is finished, a final line is added stating the average sample rate of the system during this read.

* From here, direct microphone data is available in the dataLog.txt file created in the program. One can do as they wish with the data, and just remember that the array will erase the previous file on subsequent runs if left in the same place with the same name. To start, the analyzeClient.py program properly shows the current functionality of this array. To operate:

1. Make sure you have a dataLog file previously generated, and run the “analyzeClient.py” program.
2. Select the log file generated before in the pop up window.
3. The program will start to analyze the data within the file. First, a window pops up showing a visual representation of the sound data, as a function of time for all microphones.
4. Next, a window will pop up showing the locations of detected sounds, relevant to the microphone array. The center rectangle is the array, and any red dots contain a detected sound. Clicking on them will also print the time that dot was detected.

Finally, the program will give you the option to save the position data after this analysis in another file, posTimeData.txt. Enter y or n to agree or decline to save this analysis data for any future use.

# **Extra Information**

* Two data formats are used in the auto-generated log files. The first, dataLog.txt, is just a comma separated list of values read. Left to right, is data for microphone 1-2-3-etc. For now, only three microphones are included so there are only three columns.
* As for the PosTimeData file, the goal here is to log a position with a time of when a sound was detected coming from there. Since this project currently works with a 2D grid, the first two columns account for x and y positioning, with the third term being the average time that each microphone heard that data, and thus an effective global time.
* Most errors encountered while operating this system were resolved by resetting the microcontroller and relaunching any program involved. If this fails to work, check to make sure everything is connected properly.

**Exit System**

User can exit or close by turning off the microcontroller. This can be done by unplugging the microcontroller USB cable from the laptop.