# Implementation and Evaluation of MPI on STM32

https://coder6583.github.io/15418-project-page/

Soma Narita, Josef Macera
URL:

## Progress Summary

We have been working on implementing the MPI interface. We found out that implementing MPI is harder than it seems because we are on a wired communication protocol. We experimented with UART/SPI and ring/star topology. Star topology worked well, but the problem was that there are only four SPI modules, and the scalability is poor. UART was a better fit for MPI because UART is peer-to-peer but SPI is master-slave. However, SPI is about 4x faster in terms of bandwidth (10Mb/s vs 40Mb/s), so we ended up with the SPI ring topology. We also have implemented a packet protocol that wraps around the SPI syscalls, and acts as a layer below the MPI.

## Revised Goals

The MPI implementation took longer than we expected, so we will be prioritizing working on the digital signal processing part, and push back working with actual microphones.

We *plan* to achieve:

- A working implementation of a lightweight MPI library. We plan to implement the functions below, and add more if needed.
    - MPI_Init
    - MPI_Send
    - MPI_Recv
    - MPI_Barrier
    - MPI_Allgather
- Functional matrix multiplier using multiple STM32s connected with our MPI library
    - Explore how scalable our system is
- Simulate with fake microphone data, and do the needed digital signal processing over multiple STM32s over the MPI interface
- As part of the analysis of the scaling of our project, we hope to optimize and design our system to be able to scale linearly in two ways (with fake simulated data):
    - Fix the sampling frequency of each node, and vary the number of nodes in the system that communicate
    - Fix the number of nodes in the system, and vary the sampling frequency
    - We hope to collect sufficient data and create a sufficient analysis to explain how these limitations impact our ability to scale this system in both directions.

If all goes well, we *hope* to achieve:

- Functional sound source detection under loose conditions (low sampling speed, low number of nodes) with acceptable accuracy on four physical STM32/mic nodes. We believe we can achieve this because the parallel workload is minimally distributed and we believe we will find few challenges to get a basic working physical implementation ready. We plan for this to be as a demo at the poster session, as described above - ideally on a flat table with the microphones on the edge of the table and the display in the center. People can walk around the table, clap their hands, talk, and the heat map will update accordingly and in real time.
- A physical implementation of our system on 8 physical STM32/mic nodes with much, much higher sampling frequency. The specific frequency is not yet determined by us, but we would hope to support linear scaling up to ~100x speedup in frequency, that is, varying f=1kHz to f=100kHz, for example.

## Poster Session

We will be doing a demo with multiple STM32s.

## Preliminary results

We are still working on getting matrix multiplication working on our MPI interface.

## Concerns

We are worried that there is not enough SRAM on the STM32s (only 96kB). There may be too little data on each node, and the arithmetic intensity may end up very low.

## Schedule

Week 1

1. ~~Implement SPI syscalls~~
2. ~~Design MPI API~~

Week 2

1. ~~Implement MPI_Init, MPI_Send, MPI_Recv~~
2. ~~Implement rank and communicator~~
3. ~~Debug~~

Week 3

1. Implement MPI_Barrier, MPI_Allgather
2. Run matrix multiply

(Milestone)

Week 4

1. Implement MPI_Barrier, MPI_Allgather
2. Run matrix multiply, digital signal processing

Week 5

1. Run experiments!
2. Optimize
3. Do report