Event Based Acoustic Localization

Haochen Zhao
Tianyuan Nan

Motivation and Objectives

Motivation:

Making smart surface out of everyday flat surface with 3 acoustic sensors

Objective:

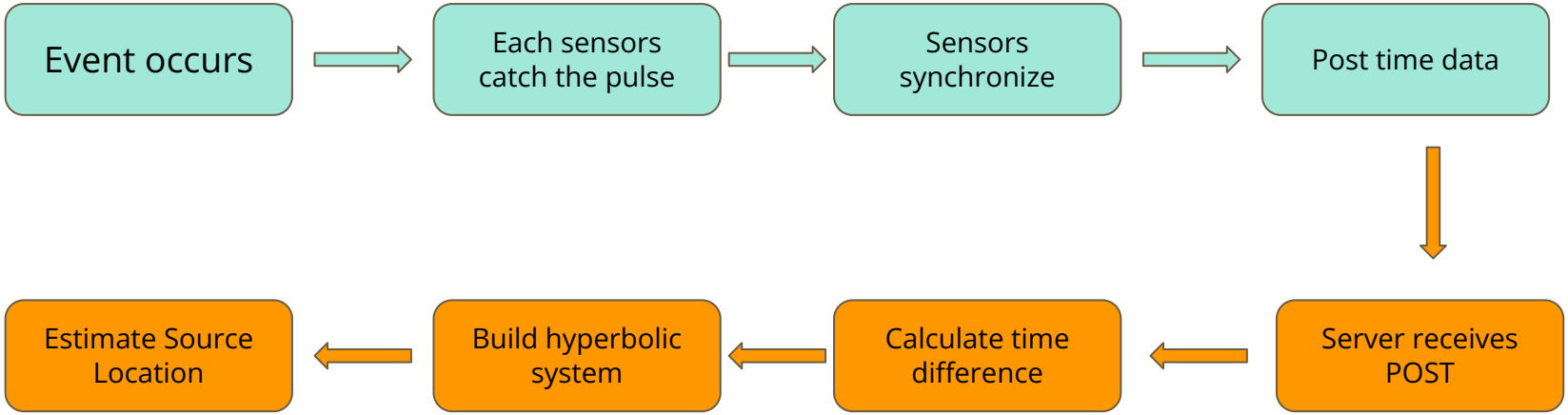
Firmware: 3 **synchronized** sensors deliver timestamps to host server

Software: Receive the data payload, calculate the time difference, and estimate the **source location**

Goals

- Familiarize ourselves with Time Difference of Arrival(TDOA) method
- Practise writing time critical code on ESP32
- Do synchronization across acoustic sensors
- Realize communication between sensors and host server
- Estimate a reasonable source location

Localization Procedure



Current Approach

Hardware:

- Camera detection: Using camera to detect the location of event happened
- Linear microphone array: microphone are arranged on the same line

Software:

- Angle of Arrival (AoA) estimation
- Controllable Beamforming Localization Method (maximizing signal power)

Novelty

- Use a bunch of independent devices instead of one device
- Potential to use large amount of sensors for accuracy gain
- Detect and synchronize all acoustically

- Time Difference of Arrival (TDOA) method
- Set up a simple host to receive post data via Wifi network
- Utilize the Jacobian matrix and scipy’s least-squares solver to ensure estimation accuracy

Work Split

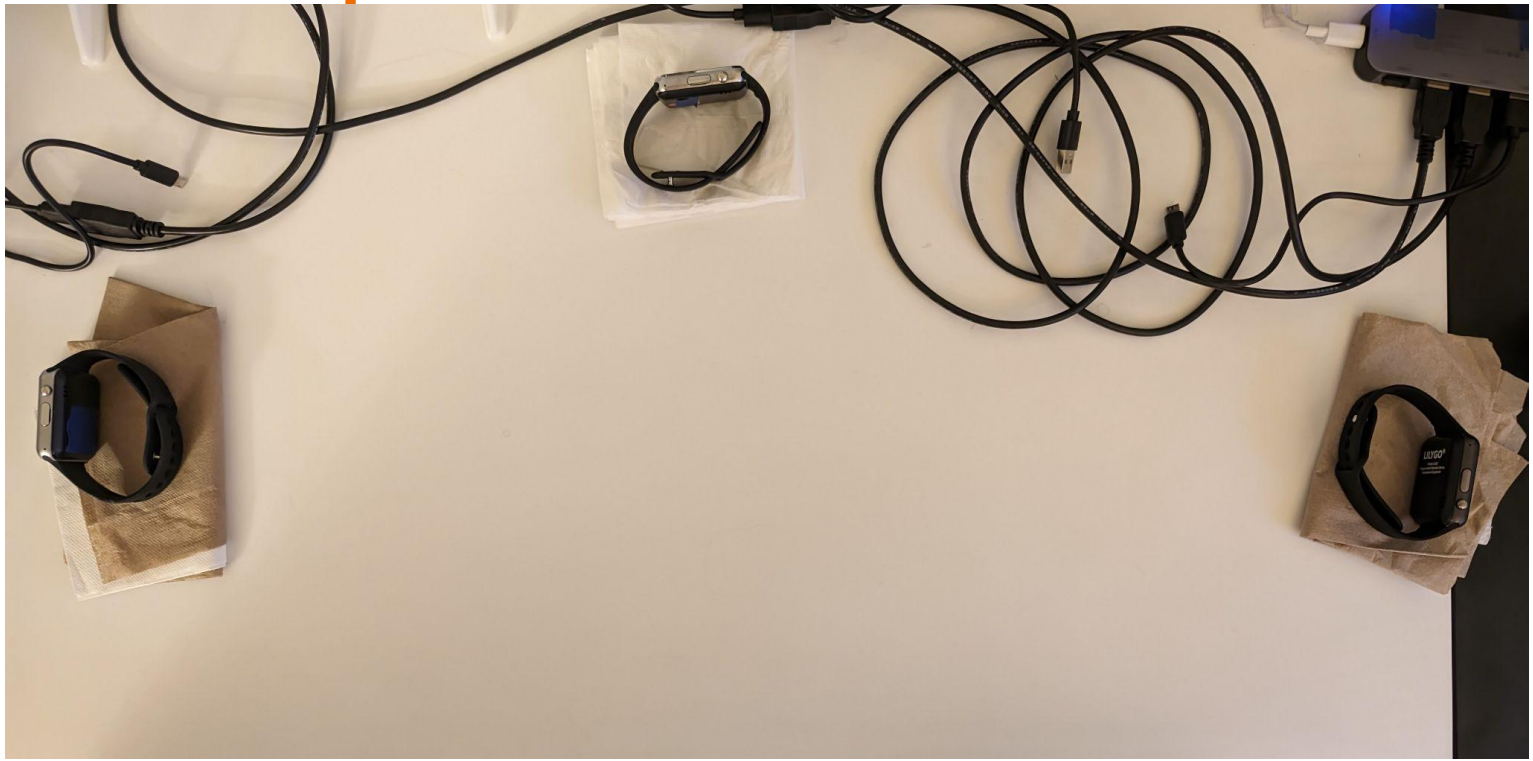
Haochen Zhao:

- audio capture routine
- audio playback routine (for sync using sound)
- acoustic event timestamping
- protocol design

Tianyuan Nan:

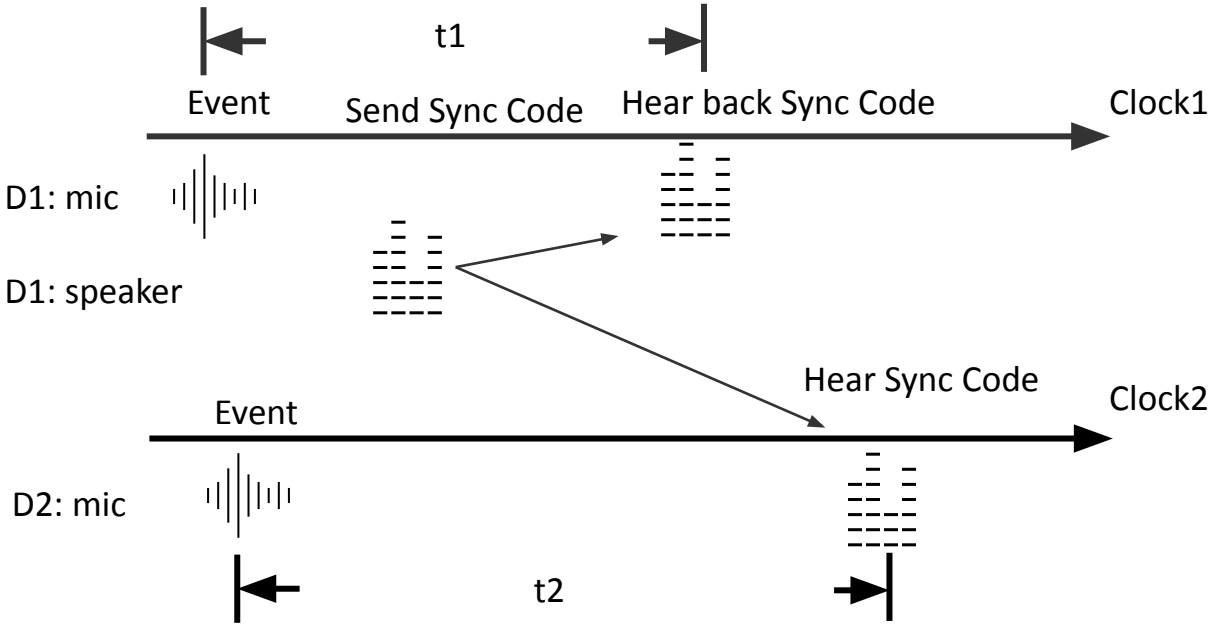
- time difference calculation and correction
- TDOA algorithm program and testing
- host server setup for data receiving
- github report and project website maintenance

Hardware Setup



3 devices at three sides
Each device has mic & speaker

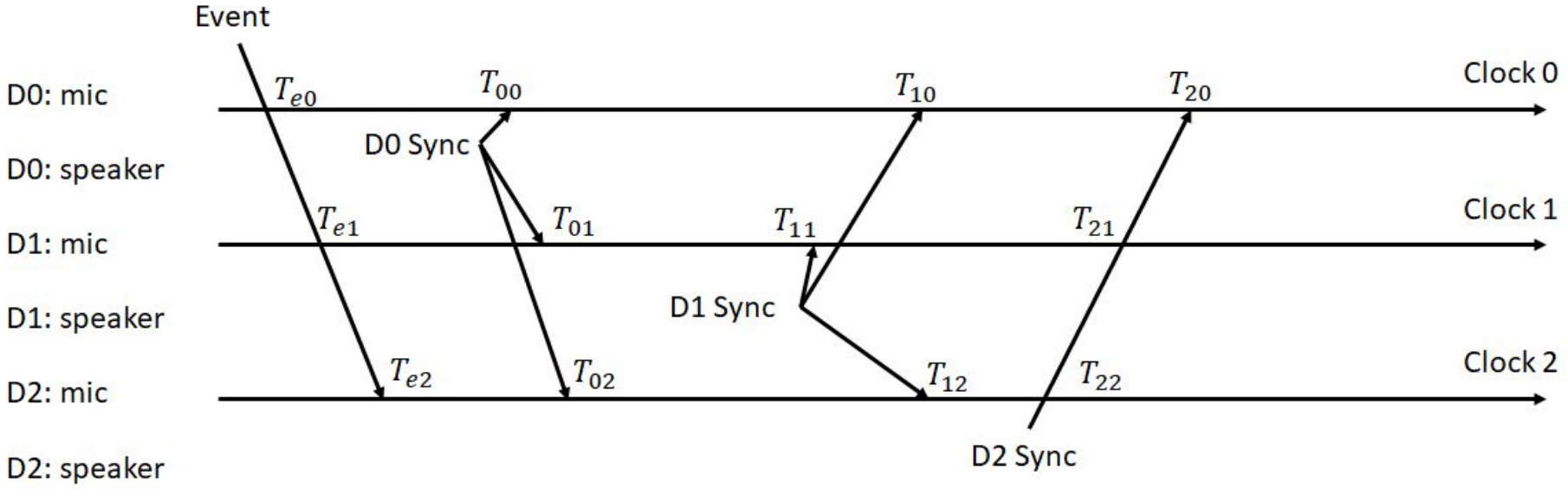
Acoustic Synchronization- Pair of Device



Event time different $\Delta t = t1 + \frac{d_{d1,d2}}{c} - t2$

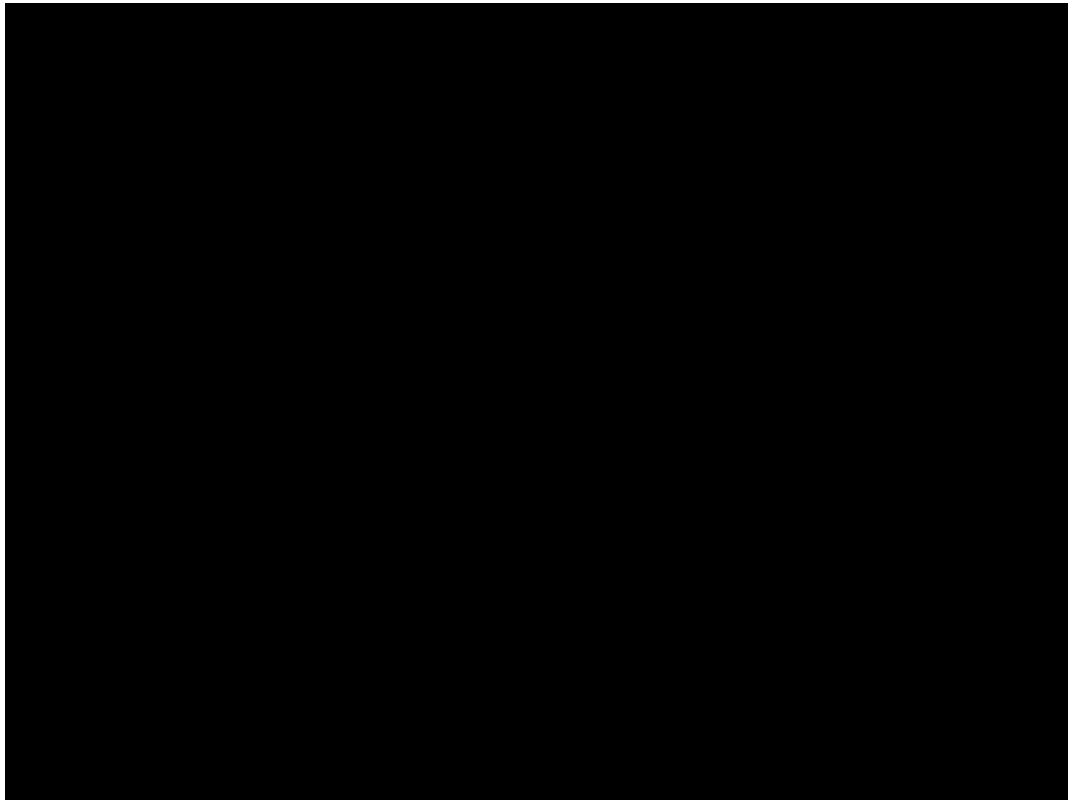
Similar to PTP
Key Differences:
No latency estimation
Process on third party

Acoustic Synchronization- All Devices



Device i sends $(T_{ei}, T_{0i}, T_{1i}, T_{2i})$ over Wi-Fi to host

Acoustic Synchronization In Action



My tap sound was followed by three sync sound in sequence

Host Server Build-up

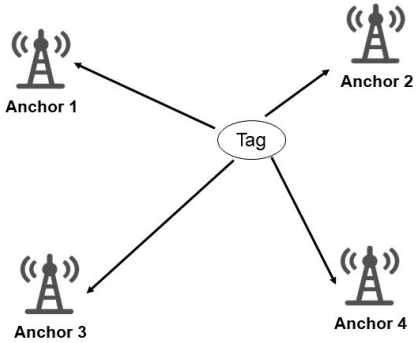
Host server

- Deal with the “POST” request from sensors
- Check each payload with three keys
- Save the payload
- Pass it to the TDOA functions

```
json_data = [
    {
    "id": 0,
    "event_ts": 10828184,
    "sync_ts": [10837460, 10849392, 10865216]
  },
  {
    "id": 1,
    "event_ts": 82892316,
    "sync_ts": [82901653, 82913456, 82929300]
  },
  {
    "id": 2,
    "event_ts": 86157589,
    "sync_ts": [86166957, 86178763, 86194541]
  }
]
```

Payload

TDOA Process



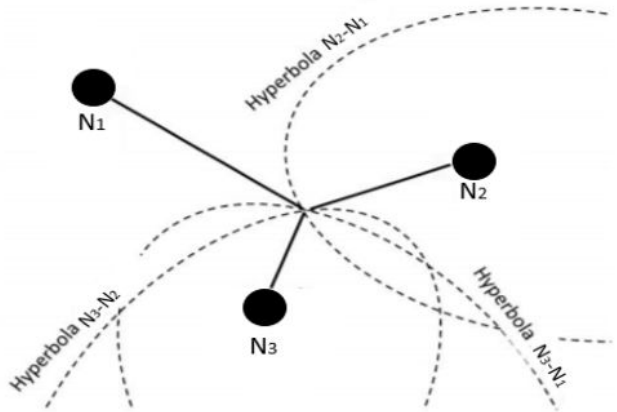
TDoA Solving for Location:

- TDoA: Time difference of arrival
- → Distance difference for the same signal
- → Hyperbolic equation for each pair of sensors
- → Localize the event

The hyperbola we're interested in can be described as

$$\sqrt{(x-x_1)^2+(y-y_1)^2}-\sqrt{(x-x_0)^2+(y-y_0)^2}-v(t_1-t_0)=0$$

where x_0, y_0 is the position of the first observer, t_0 is its observation time, x_1, y_1, t_1 are those parameters of the second observer, and v is the propagation speed of the wave we're interested in.



Solving TDOA System

Solving steps:

- Build up three hyperbolic equations
- Specify the Jacobian matrix of this system (partial derivative of each equation)
- Set the initial position
- Estimate the event location

```
# specify the Jacobian matrix of this system. This is just a collection of the partial derivatives of each function with respect to each independent variable
def jacobian(x0, y0, x1, y1, x2, y2, d01, d02, d12):
    def fn(args):
        x, y = args
        adx = (x - x1) / np.sqrt(np.power(x - x1, 2.) + np.power(y - y1, 2.)) - (x - x0) / np.sqrt(np.power(x - x0, 2.) + np.power(y - y0, 2.))
        bdx = (x - x2) / np.sqrt(np.power(x - x2, 2.) + np.power(y - y2, 2.)) - (x - x0) / np.sqrt(np.power(x - x0, 2.) + np.power(y - y0, 2.))
        cdx = (x - x2) / np.sqrt(np.power(x - x2, 2.) + np.power(y - y2, 2.)) - (x - x1) / np.sqrt(np.power(x - x1, 2.) + np.power(y - y1, 2.))
        ady = (y - y1) / np.sqrt(np.power(x - x1, 2.) + np.power(y - y1, 2.)) - (y - y0) / np.sqrt(np.power(x - x0, 2.) + np.power(y - y0, 2.))
        bdy = (y - y2) / np.sqrt(np.power(x - x2, 2.) + np.power(y - y2, 2.)) - (y - y0) / np.sqrt(np.power(x - x0, 2.) + np.power(y - y0, 2.))
        cdy = (y - y2) / np.sqrt(np.power(x - x2, 2.) + np.power(y - y2, 2.)) - (y - y1) / np.sqrt(np.power(x - x1, 2.) + np.power(y - y1, 2.))

        return [
            [adx, ady],
            [bdx, bdy],
            [cdx, cdy]
        ]
    return fn
```

Evaluation

1. The error-free delivery of data by the firmware.
2. The successful reception of POST requests by the host with accurate payload
3. Correctness of calculated time difference
4. A reasonable and plausible estimated coordinate for the event source

Hardware Setup Result

Timestamping and synchronization:

- ✓ ESP32 development environment
- ✓ Speaker and microphone working together
- ✓ Waveform generator for sync sequence
- ✓ Event timestamping
- ✓ Sync sequence timestamping and control
- ✓ Upload to host over Wifi

Experiment Result

Experiment Setup:

Coordinate of sensors: [0,0], [-0.39, 0.26], [-0.74, 0]

Propagation speed: V = 340 m/s

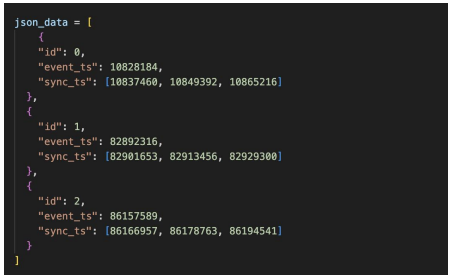
Sample frequency: F = 44100 Hz

```
 * Serving Flask app "server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://192.168.1.162:5000/ (Press CTRL+C to quit)
Received JSON data: {'id': 0, 'event_ts': 10828184, 'sync_ts': [10837460, 10849392,
10865216]}
received: [[10837460, 10849392, 10865216], [], []]
192.168.1.162 - - [13/Dec/2023 15:57:57] "POST /post_json HTTP/1.1" 200 -
Received JSON data: {'id': 1, 'event_ts': 82892316, 'sync_ts': [82901653, 82913456,
82929300]}
received: [[10837460, 10849392, 10865216], [82901653, 82913456, 82929300], []]
192.168.1.162 - - [13/Dec/2023 15:57:57] "POST /post_json HTTP/1.1" 200 -
Received JSON data: {'id': 2, 'event_ts': 86157589, 'sync_ts': [86166957, 86178763,
86194541]}
received: [[10837460, 10849392, 10865216], [82901653, 82913456, 82929300], [86166957
, 86178763, 86194541]]
3 val: [[10837460, 10849392, 10865216], [82901653, 82913456, 82929300], [86166957, 8
6178763, 86194541]]
[-4.6268199123122855e-06, 9.030278778177081e-05, 0.0005113913760513844]
Result: [-0.33101026 -0.0164209 ]
192.168.1.162 - - [13/Dec/2023 15:57:57] "POST /post_json HTTP/1.1" 200 -
```

Experiment Result

Experiment Setup: [0,0], [-0.39, 0.26], and [-0.74, 0]; V = 340m/s; F = 44100Hz

1. Success on delivering data from sensors and receiving the payload on host
2. Accurate calculation on time difference
3. A reasonable estimation on the source location



Program Calculation	Manually Calculation
t01: -4.62682E-06	t01: -4.62682E-06
t02: 9.030279E-05	t02: 9.03028E-05
t12: 0.00051	t12: 0.00051

axis	Source	estimated Source	Standard Deviation	Diff in Distance
x	-0.4	-0.3310	0.0345	0.0690
y	0	-0.0164	0.0082	0.0164

Next Step

- Enhance the host server to facilitate automated location estimation
  - Repeated post requests problem
  - Incomplete payload value
- Replace simple thresholding with DSP for timestamping
  - Combat noise
  - Better sensitivity for sync sequence
  - Better timestamping accuracy
- Improve Estimation algorithm
  - Lower deviation

Thank you!