Overall system architechture

Sensors -> filter -> OPAMP -> ADC -> MCU -> PC

- Sensors
    - RTD:
        - good accuracy, linearity, rugged, electrical noise immunity
        - bad response time
        - susceptible to vibration
        - Low operational range
        - expensive
    - Thermocouple (TC)
        - higher temperature range
        - long term stability not too good -> need recalibration more often
        - does not require power/excitation
            - vibration resistant
        - cheap
    - Thermistor
        - narrow ranges (as long as it's within the operational range issa gucci)
        - fast response
        - good accuracy
- Signal conditioning (analog)
    - filters for noises
    - OPAMP for span matching
- ADC
    - external
        - on chip adc are susceptible to noise due to being on the same die with other components
        - flexibility of sensor placement (we want analog signal lines to be as short as possible due to signal degradation over long distance)
            - Allow better physical isolation for mcu system
        - potential for simultaneous samplings
        - Offer differential inputs (more accurate)
        - bit depth is more guaranteed on external ones
        - *include the maths here*
- MCU
    - Teensy

        - fastest clock
        - mainstream and easy to use (similar to arduino software wise)
        - 3 SPI instances
        - 100mb ethernet
        - ready to go development board package
        - usb is actually will be faster but signal degradation is bad > 2m
    - STM32 (H7)
        - dual cores = separation of concerns
        - more optimized
        - more efficient
        - fast
        - versatile when it comes to peripherals
        - good resume builder/skills/learning experience as industries use them extensively
        - more customizable when you wanna do your own custom pcb from scratch
        - ethernet
        - steep learning curve
            - configure the clock your self
            - set up registers yourself
            - etc.
        - development is going to take time
        - official documents is good
    - It's worth mentioning that there are profession DAQ devices/packages out there already. However, they are very expensive and proprietary. They are super convenient and just work but no learning experience and costly.
    - A balance between performance/development time/cost/learning experience
    - Everything else are cheaper but slower or lack peripheral supports and not that significantly cheaper than stm32h7 or teensy.
- Front end
    - data shift out through rj45 with a-udp to display on PC
    - Data visualization with Flutter
    - Store to Redis (nosql)
- Concerns:
    - preparing data packets before shifting out (memory, dma, ??)
    - Data synchronization? (already helpful with a fast sampling adc and chip as the time delay between samples are super small)
    - Programming structures (do we wanna do OOP? what need to be included? etc.)
    - Devising experiments

- what to test?
- how to test?
- equipments needed?