Time Syncronization via Sensing

- Motivation
    - Time synchronization is an essential part of networked embedded devices. It enables various distributed applications like smart homes, automated agriculture and autonomous vehicles. However, many devices at the edge are resource constrained and existing timing services are a burden on their resources. On the other hand most of these devices are equipped with sensors and send periodic sensing data to gateway servers. We want to leverage this time-stamped sensor data received at the gateway to synchronize the sensing devices.
    - We wanted to do this project because we are both interested in autonomous networked systems. As this is a fairly essential section of networked autonomous robotics, we believe that this would be both an interesting project and useful in future projects.
- Design Goals
    - Main Project Goal: Develop a time sync protocol using timestamped sensor data from two embedded devices. The data will be received by a raspberry pi device, and the synchronization protocol will run on the raspberry pi device
    - Accurately syncronize the two embedded sensors and the raspberry pi
    - Create an efficient protocol that can be used in an embedded system
- Deliverables
    - Characterize the network delay between raspberry pi and the edge device
    - Estimate the relative clock drift between the participating devices
- Hardware Requirements
    - ESP32 Things
- Project Timeline (As we are unsure of the deadline for this project, it is difficult to meter out time for individual sections)
    - Do pertinent external research (1 week)
    - Prototype syncronization protocol using a single sensor (3-5 weeks, depending on final demo)
    - Introduce the second sensor into the system, reworking the protocol to accept the data (1-2 weeks)
    - Prepare for demonstration and generate draft report (1-2 weeks)
    - Final Demonstration and Report TBD
- Team Member Responsibilities
    - Ian: Networking between Raspberry Pi and Sensors; Calculate network delay between devices
    - John: Determine relative clock drift of each edge device; Interpret incoming data and determine correct outputs from the Rasp Pi
- References
    - Automated Synchronization of Driving Data Using Vibration and Steering Events
    - FLIGHT: clock calibration using fluorescent lighting
    - Exploiting Smartphone Peripherals for Precise Time Synchronization