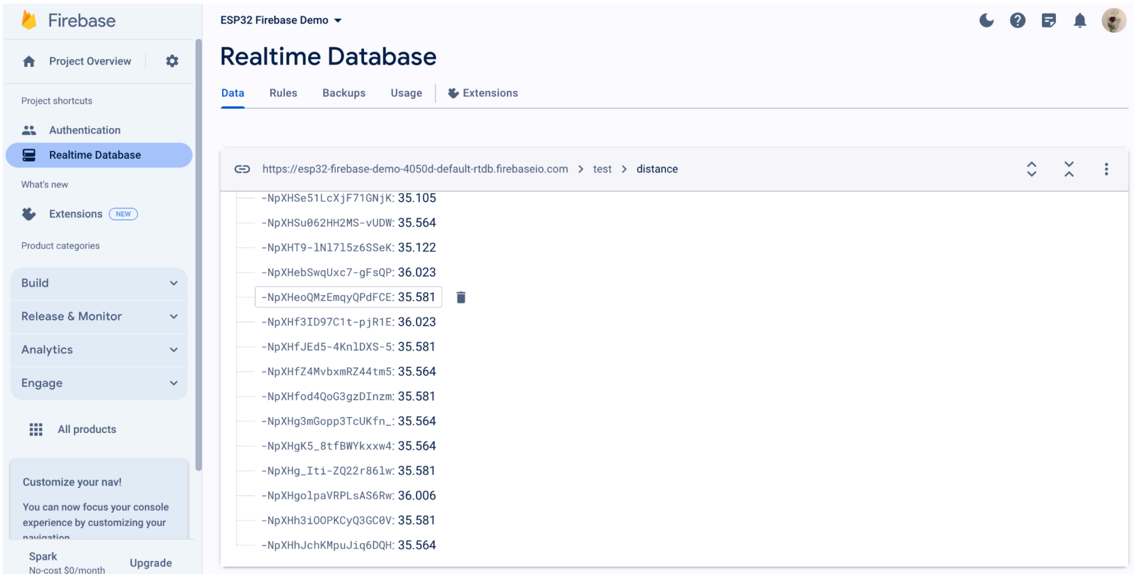
- Screenshot of your firebase RTDB receiving the ultrasonic sensor readouts.
    - It is acceptable that the data showing up on Firebase are all 0s, since this is not the objective of this lab.

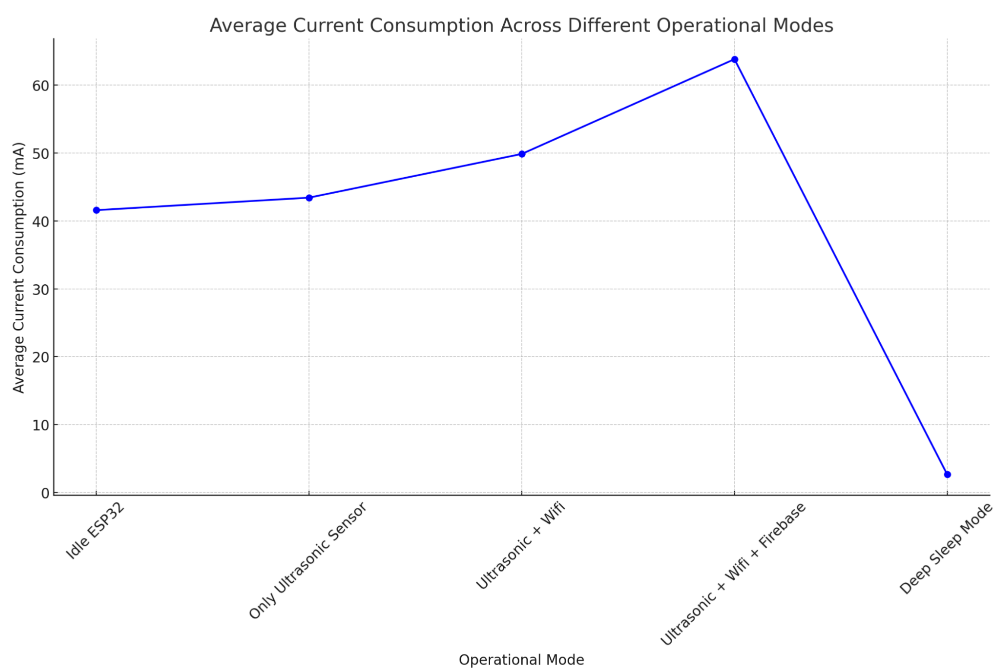


< 6. Connect your ESP32 circuit to your Power Profiler Kit and view power consumption through the Power Profiler App>

- Annotated screenshot of the plot on your power profiler app to show the power consumption in 5 different stages of ESP32S3's usage.
    - You need to have a single screenshot showing all five stages altogether, with each stage labeled/annotated.

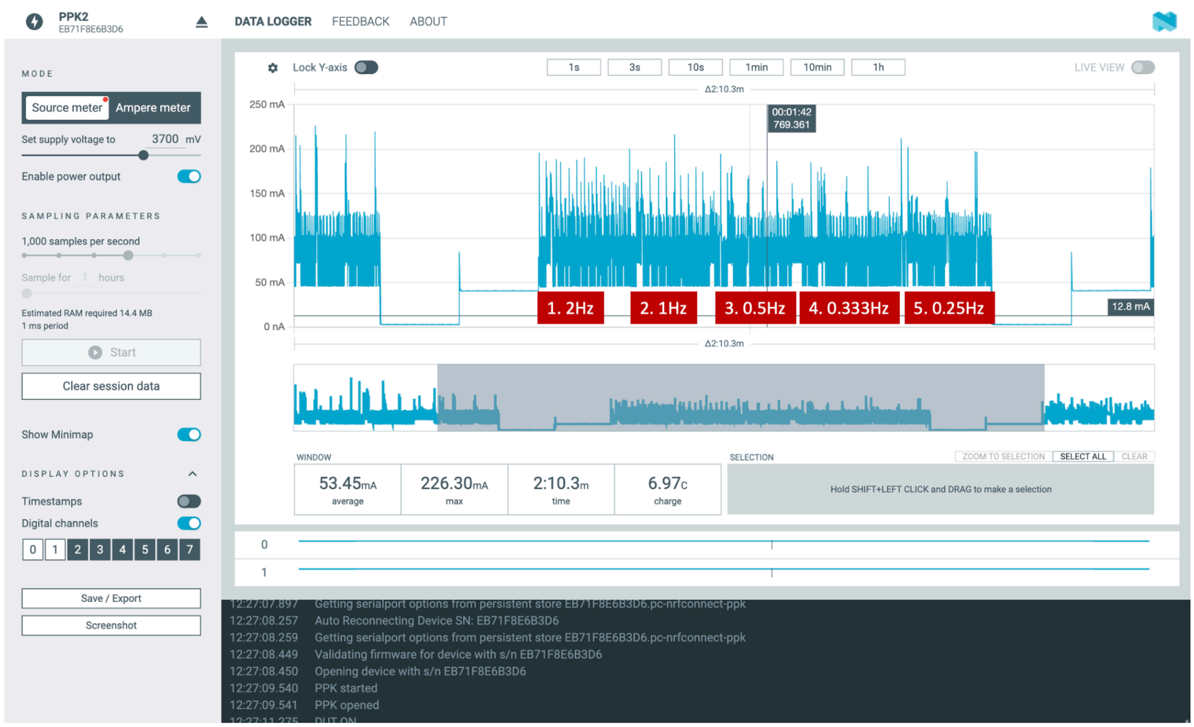
- Calculations of each state's power consumption and estimated battery-lasting time, respectively.
  - You should zoom in the window to make sure it shows the power consumption within only ONE STAGE, and then read the averaged current.
  - 1. Idle ESP32 (not running WiFi or ultrasonic sensor)
    - 500(mAh) / 41.59(mA) = 12.02 hrs
  - 2. Only ultrasonic sensor working
    - 500(mAh) / 43.43(mA) = 11.51 hrs
  - 3. Ultrasonic + Wifi working
    - 500(mAh) / 49.87(mA) = 10.03 hrs
  - 4. Ultrasonic + Wifi + Sending data to Firebase
    - 500(mAh) / 63.83(mA) = 7.83 hrs
  - 5. Deep Sleep mode
    - 500(mAh) / 2.65(mA) = 188.68 hrs

- Demonstrate the power consumption (W) during the data transmitting stage with different data transmitting frequencies. Plot the figure demonstrating the correlation between data transmitting frequency (Hz) and power (W).
  - Same as above, you do NOT need to take a screenshot with data transmitting frequency, but you need to read the averaged current with the same frequency being used throughout the plot on your window.



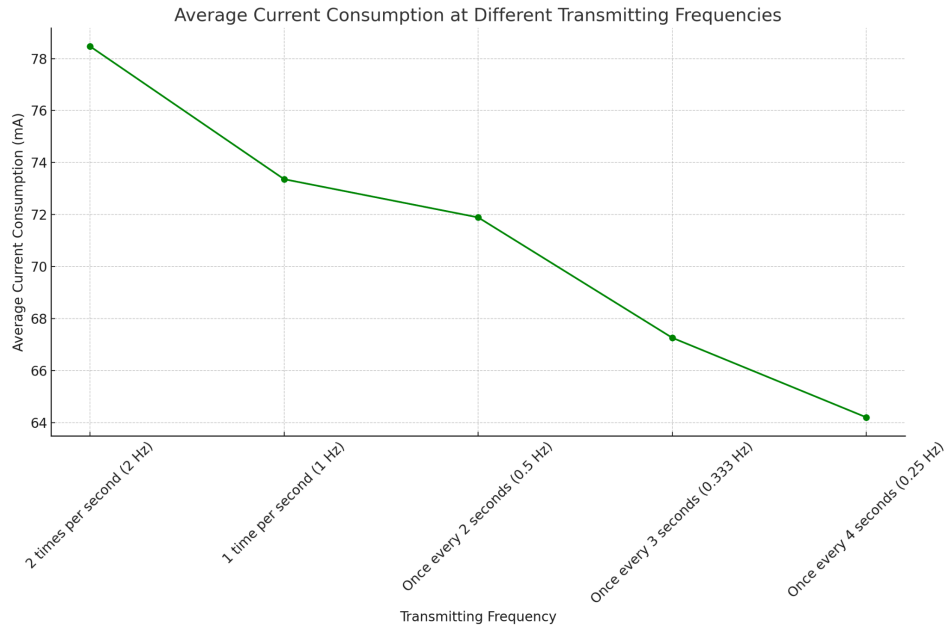
< 7. Estimate the battery life for different use cases >

- Annotated screenshot of the plot on your power profiler app to show the power consumption in 5 different stages of ESP32S3's usage.
    - You need to have a single screenshot showing all five stages altogether, with each stage labeled/annotated.



- Calculations of each state's power consumption and estimated battery-lasting time, respectively.
    - You should zoom in the window to make sure it shows the power consumption within only ONE STAGE, and then read the averaged current.
    - 1. 2 times per second (2 Hz)
        - 500(mAh) / 78.47(mA) = 6.37 hrs
    - 2. 1 time per second (1 Hz)
        - 500(mAh) / 73.36(mA) = 6.81 hrs
    - 3. Once every 2 seconds (0.5 Hz)
        - 500(mAh) / 71.89(mA) = 6.95 hrs
    - 4. Once every 3 seconds (0.333 Hz)
        - 500(mAh) / 67.26(mA) = 7.43 hrs
    - 5. Once every 4 seconds (0.25 Hz)
        - 500(mAh) / 64.20(mA) = 7.78 hrs

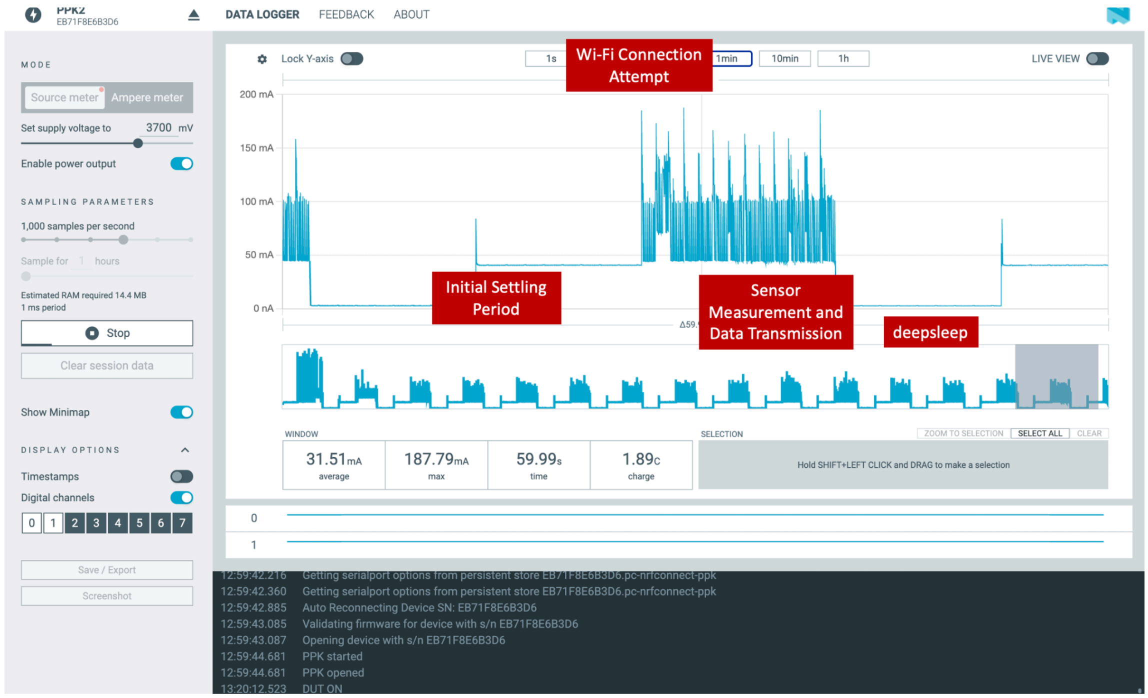
- Demonstrate the power consumption (W) during the data transmitting stage with different data transmitting frequencies. Plot the figure demonstrating the correlation between data transmitting frequency (Hz) and power (W).
    - Same as above, you do NOT need to take a screenshot with data transmitting frequency, but you need to read the averaged current with the same frequency being used throughout the plot on your window.



- Description of your own power-saving strategies/policies.

  The power-saving strategy for the ESP32-S3 device focuses on optimizing power consumption by using deep sleep modes and minimizing unnecessary data transmissions. The device uses an ultrasonic sensor to detect objects; if no object is detected within 50cm for 30 seconds, it enters deep sleep for 30 seconds to conserve energy. Data is transmitted to Firebase only when an object is detected within 50cm, reducing the frequency of active transmissions and thereby saving power. This approach ensures efficient use of the 500mAh battery, allowing the device to operate for 24 hours without recharging.

- Annotated screenshots of the power consumption plot of your power-saving strategy's use case. The screenshot should at least include the deep-sleep stage and the working stage of your device.



- Estimate your strategy's electric consumption (mAh) in 24 hours under your simulated scenario. Prove that your strategy can make your device work for at least 24 hours with a 500mAh battery.

  The device's activities, including WiFi operation, sensor readings, and data transmission to Firebase, significantly affect its power consumption. Initially, the device operates actively for 48 seconds per cycle, involving sensor measurements and WiFi communications, before entering a 12-second deep sleep mode to conserve energy.

  Our calculations reveal that the total daily power consumption is approximately 1344.05mAh, exceeding the 500mAh battery capacity. This indicates that, without modifications, the device cannot sustain 24 hours of operation on a single charge. The formula for calculating the total daily consumption combines the active and sleep mode consumptions:

$$\text{Total Daily Consumption (mAh)} = \left(\text{Active Current (mA)} \times \frac{\text{Active Duration (s)}}{3600} \times \text{Cycles Per Day}\right) + \left(\text{Deep Sleep Current (mA)} \times \frac{\text{Deep Sleep Duration (s)}}{3600} \times \text{Cycles Per Day}\right)$$

To align with the 500mAh capavcity for 24-hour operation, consider:

- Extending deep sleep duration beyond 12 seconds.
- Reducing the frequency of sensor readings and WiFi transmissions.
- Optimizing power consumption during active phases, possibly by adjusting the WiFi transmission power or batching data to minimize active communication time.

Implementing such optimizations can significantly reduce power consumption, potentially allowing the device to operate within the constraints of a 500mAh battery for the desired duration. Adjusting operational parameters to minimize active consumption while maximizing sleep periods is key to achieving efficient power management.

- The link to your edited code with your power-saving strategy on GitHub. You can either create a new repo or push the updates to your existing repo for all labs (recommended).
  https://github.com/chelsey0527/TECHIN514_Lab5/tree/main/Lab5-power_saving