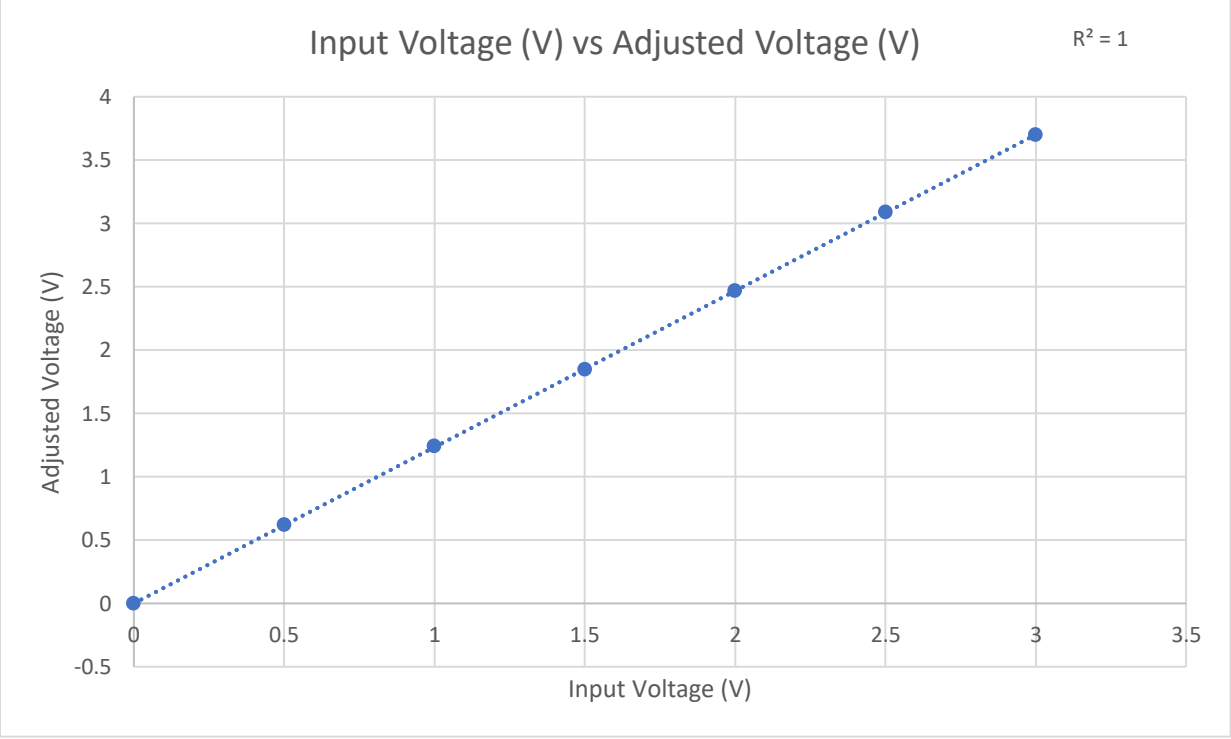
Final Project: Testing & Analysis

Verify the accuracy of your battery level measurement for 0-3.7 V.

A normalized battery level was sent to the Bluetooth Battery Level GATT. The table below shows the actual battery level sent to the development kit and the logged battery level. However, the board can only take in voltages between 0-3 V. So, the software has a function that calculates the actual battery voltage with a ratio such that 3 V is a 3.7 V battery voltage, and 0 V is a 0 V battery voltage. Since this is a ratio, the adjusted voltage will not be the same as the input voltage. But there is a linear relationship between the input voltage and adjusted voltage. This is shown with the graph below the table. The table below shows the results of the adjusted voltage and the outputted battery level to the nRF connect app.

Input Voltage (V)	Adjusted Voltage (V)	Outputted Battery Level
0	-0.002	0
0.5	0.617	16
1	1.238	33
1.5	1.847	49
2	2.467	66
2.5	3.085	83
3	3.698	99

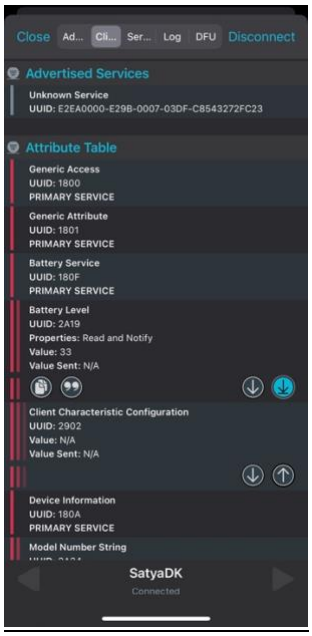


The adjusted voltage and the input voltage have a strong linear relationship. Using this linear relationship, we can accurately set a battery level measurement between 0 and 3.7 V.

Final Project: Testing & Analysis

Verify that you can send those battery levels to the nRF Connect app using the Bluetooth Battery Level GATT.

From the experiment above, I have verified that I can send these battery levels to the app. This can be tested by others by downloading the firmware. A screenshot is provided below when 1 V is supplied to the board.



Using the function generator and the oscilloscope, make measurements varying the 100 and 500 Hz input signals linearly over their V_{p-p} ranges, and measure the corresponding LED brightness output for each input.

Duty Cycle (%) was used to quantify the LED brightness output for each input. The tables with the values are shown below.

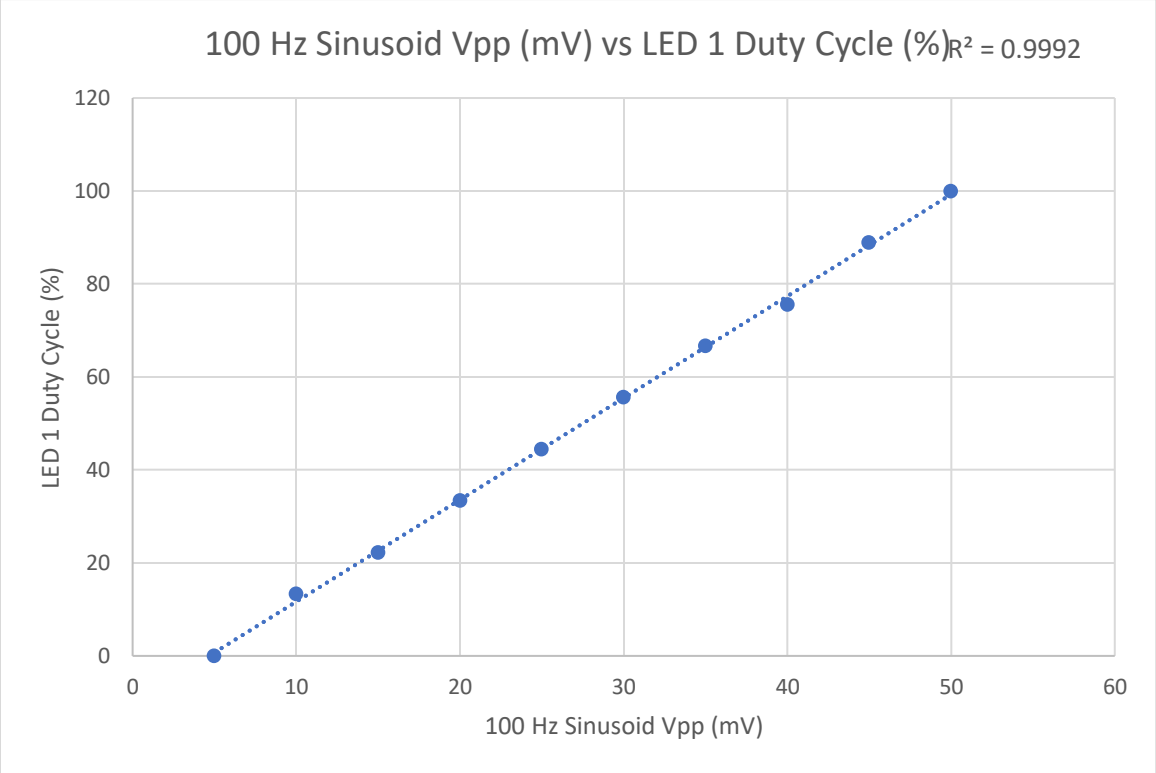
100 Hz sinusoid (mV)	Duty Cycle (%)
5	0
10	13.33
15	22.22
20	33.33
25	44.44
30	55.55
35	66.67
40	75.56
45	88.89
50	100

Final Project: Testing & Analysis

500 Hz sinusoid (mV)	Duty Cycle (%)
10	0
20	6.42
30	13.58
40	20.71
50	27.85
60	35
70	42.14
80	49.28
90	56.42
100	63.57
110	70
120	77.85
130	85.71
140	91.42
150	99.29

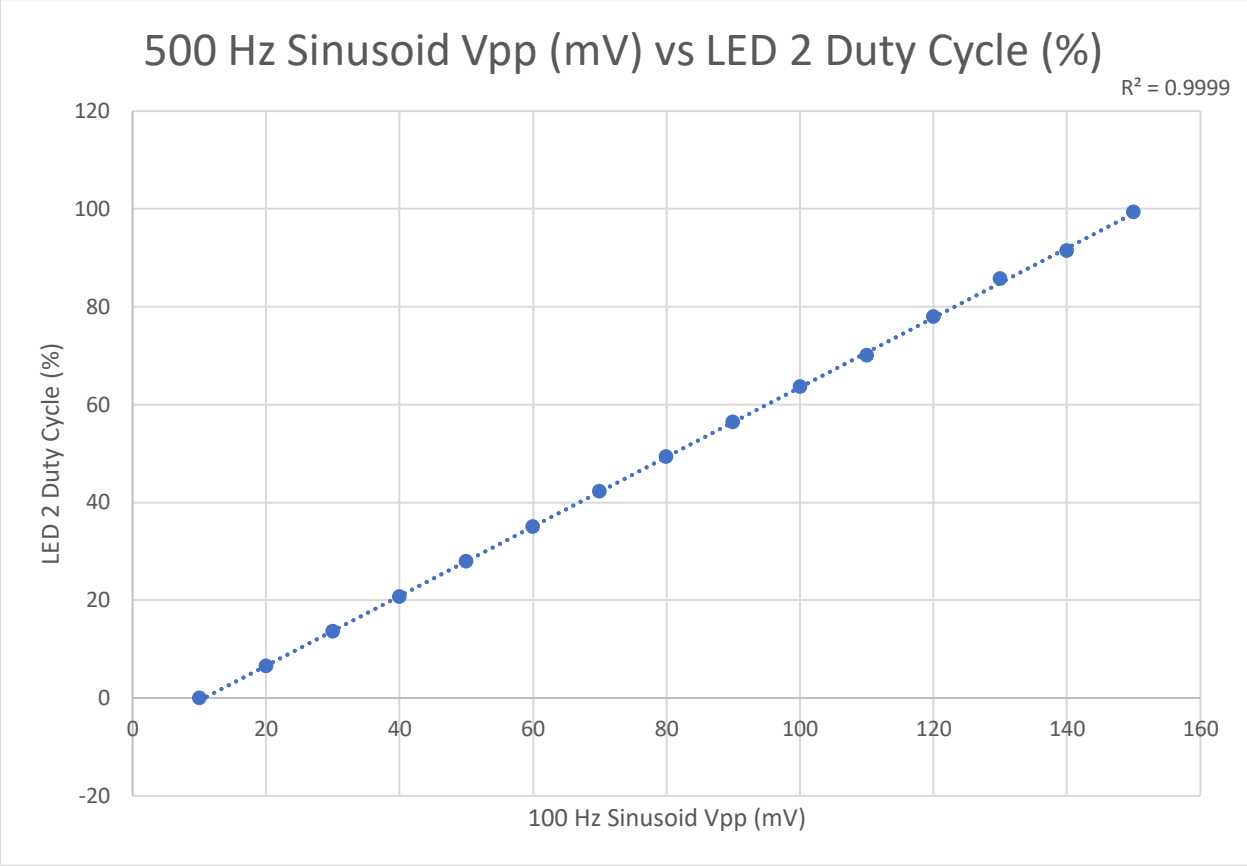
How linear is the relationship between V_{p-p} and PWM output? Quantify this relationship with linear regressions and associated R2 values.

The graphs for each input are shown below.



Final Project: Testing & Analysis

In the graph above, the R^2 value is 0.9992 which means that there is a strong linear correlation between the Vpp values of the 100 Hz sinusoid and the LED 1 Duty Cycle (%).



In the graph above, the R^2 value is 0.9999 which means that there is a strong linear correlation between the Vpp values of the 500 Hz sinusoid and the LED 2 Duty Cycle (%).

Demonstrate that your 5 second data saving algorithm works by amplitude modulating your input sinusoids over those 5 second windows in a known manner to compare with your saved arrays.

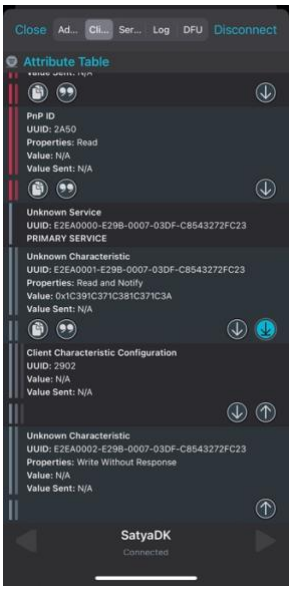
Two experiments were conducted. In the first experiment, the 100 Hz sinusoid's amplitude was modulated from 40 mVpp to 30 mVpp in the 5 second data saving window. In the second experiment, the 500 Hz sinusoid's amplitude was modulated from 80 mVpp to 40 mVpp in the 5 second window. The table below has the results. The array converted back to decimal is also shown in the table.

100 Hz Modulation from 40 mVpp to 30 mVpp	Byte Array: 0x1C371C37193515391538 Decimal Array: {28, 55, 28, 55, 25, 53, 21, 57, 21, 56}
500 Hz Modulation from 80 mVpp to 40 mVpp	Byte Array: 0x1C371C391C2E1C1C1C1C Decimal Array: {28, 55, 28, 57, 28, 46, 28, 28, 28, 28}

Final Project: Testing & Analysis

These numbers are expected as the sinusoids with 30 mVpp, 40 mVpp, and 80 mVpp have RMS values of 21, 29, and 57 respectively. Therefore, this demonstrates that the 5 second data saving algorithm for modulating input sinusoidal waves work.
Demonstrate the ability to receive your two data arrays on the nRF Connect app.

From the experiment above, I have verified that I can receive two data arrays on the app. The output to the app is a 10-byte hex array. The structure of the output can be found in the firmware. All the expected RMS values in the array are displayed. For this example, the 100 Hz sinusoid signal has 40 mVpp and the 500 Hz sinusoid signal has 80 mVpp. The byte array collected is 0x1C391C371C381C371C3A. This converts to a decimal array of {28, 57, 28, 55, 28, 56, 28, 55, 28, 58}. The RMS values for 40 mVpp and 80 mVpp are 29 and respectively. So, it works. A screenshot of the data array is shown below.



Demonstrate the safety feature of your device to cease function and blink LED3 when VBUS is HIGH.

To test the safety feature of the device, the following experiment was conducted. A baseline Bluetooth notification with 30 mVpp from the 100 Hz sinusoid signal and the 80 mVpp from the 500 Hz sinusoid signal. Then, VBUS was connected, and the amplitude of the 100 Hz sinusoid was changed to 40 mVpp while the 5 second RMS collection period was going on. Then, the BLE notification of the arrays were sent while VBUS was on. Then, the two arrays were compared. The arrays are 0x15391538153815391537 baseline and 0x153A153A153A153A153A while VBUS was high. The RMS hex value of 15, 21 in decimal, is the same for both arrays. Therefore, there was no measurement while VBUS was HIGH. The LED does blink at a rate of 1 Hz when VBUS is on. Both functionalities can be tested by others by downloading the firmware.

Final Project: Testing & Analysis

If your device wasn't limited to 100 and 500 Hz inputs, what is the maximum input frequency that your device can support without aliasing? Answer this question from a theoretical perspective and experimentally.

From the product specifications of the nRF52833-DK, the maximum sampling rate of the device is 200 kHz. So theoretically, the maximum input frequency that the device can support without aliasing is 100 kHz. This is because of Nyquist sampling theorem which states the sinusoid must be sampled at twice the maximum frequency. So, you would divide 200 kHz by 2 to get 100 kHz.

Experimentally, I would need to conduct an experiment to confirm the theoretical value. This can be done by inputting sinusoids of different frequencies with the same 40 mVpp value and confirming that the RMS value from the log is accurate. When the RMS value received from the log is not accurate, then it can be concluded that aliasing is occurring. I changed my code accordingly by changing the sampling period of the timer in accordance with the Nyquist sampling theorem. The table below has the input frequency into the development kit and the corresponding RMS value.

Input Frequency	RMS Value
100 Hz	28
500 Hz	28
1 kHz	28
10 kHz	28
100 kHz	29
500 kHz	21

For reference, the expected RMS value for 40 mVpp is 29. So, at 500 kHz the signal is getting aliased as the RMS value is approximately 21 which is incorrect. So, the theoretical value is at about 100 kHz as expected in the theoretical calculations.