ESE 2150 LAB 10
Panos Dimtsoudis & Damodar Pai
Demo Date: December 5th, 2025
Submission Date: December 5th, 2025

Part A

1. Summarize your design and measurement procedure and your observations in one or two paragraphs and include a picture of your breadboarded design. Include your linear element component design choices for the LPF, HPF and virtual ground amplifier.

The point of this lab is to take a heartbeat (ECG signal) and amplify/clean it. We learned that an ECG works by taking a particular wave with a small amplitude and then applying a gain to it so that it could be read by an oscilloscope or your local EKG. Our experiment had 4 main components. First, we had the first op-amp connected to our virtual ground. The point of this op-amp was to detect our signal by taking the difference between the positive IN and negative IN. We then apply an initial amplification of around 32-24 dB which is exactly what we expect from our diagram.

The virtual ground here is also very important. Since our heart signal is an AC, we need an offset so that we can actually see the negative half of our wave. We want to maximize R9 and R11 so that we allow for the maximum voltage swing. This value is essentially anything that satisfies R9 = R11, so we choose 10 kOhm which seemed to work well for us.

The second op-amp is our low-pass filter. To get an upper cutoff frequency of 160 Hz, we use a R1 = 1 kOhm, C6 = 1 uF. This is so that we only get frequencies under 160 Hz.

The third op-amp is our high-pass filter with a lower cutoff frequency of 0.5 Hz. We use R4 = 330 kOhm and C4 = 10 uF.

The last important part of our circuit are the capacitors on the side of our circuit. These are used to remove noise and make the signal as clear as possible.

LAB 10:

Low-pass cutoff frequency:

$f_c = 160\,Hz \Rightarrow \frac{1}{2\pi RC} = 160 \Rightarrow$

$RC = \frac{1}{320\pi} \xrightarrow{C=10nF} R = 10k\Omega$

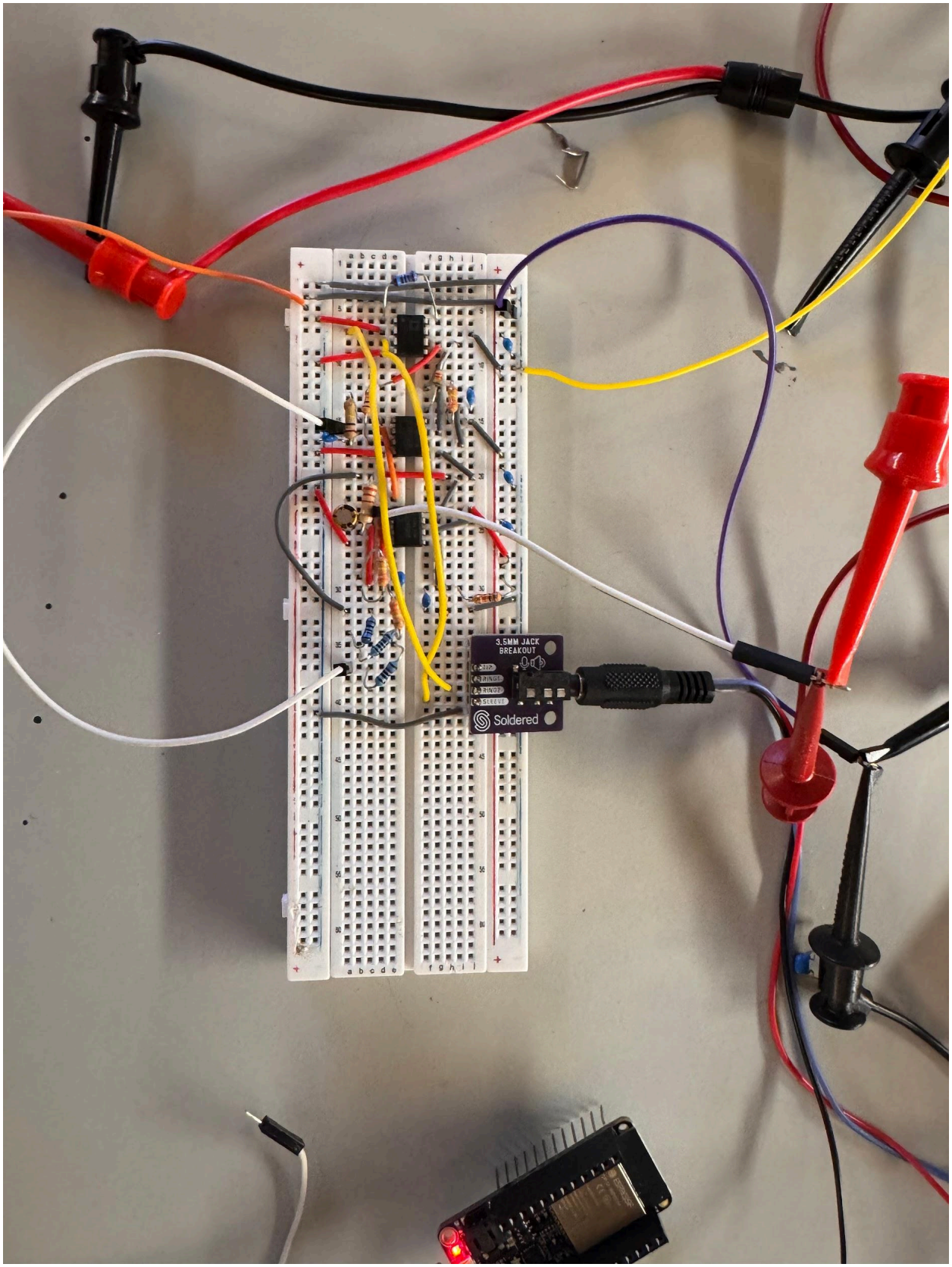
High-pass cutoff frequency:

$f_c = \frac{1}{2}\,Hz \Rightarrow \frac{V}{2\pi RC} = \frac{1}{\sqrt{2}} \Rightarrow$

$RC = \frac{1}{\pi} \xrightarrow{C=10\mu F} R \approx 33k\Omega$

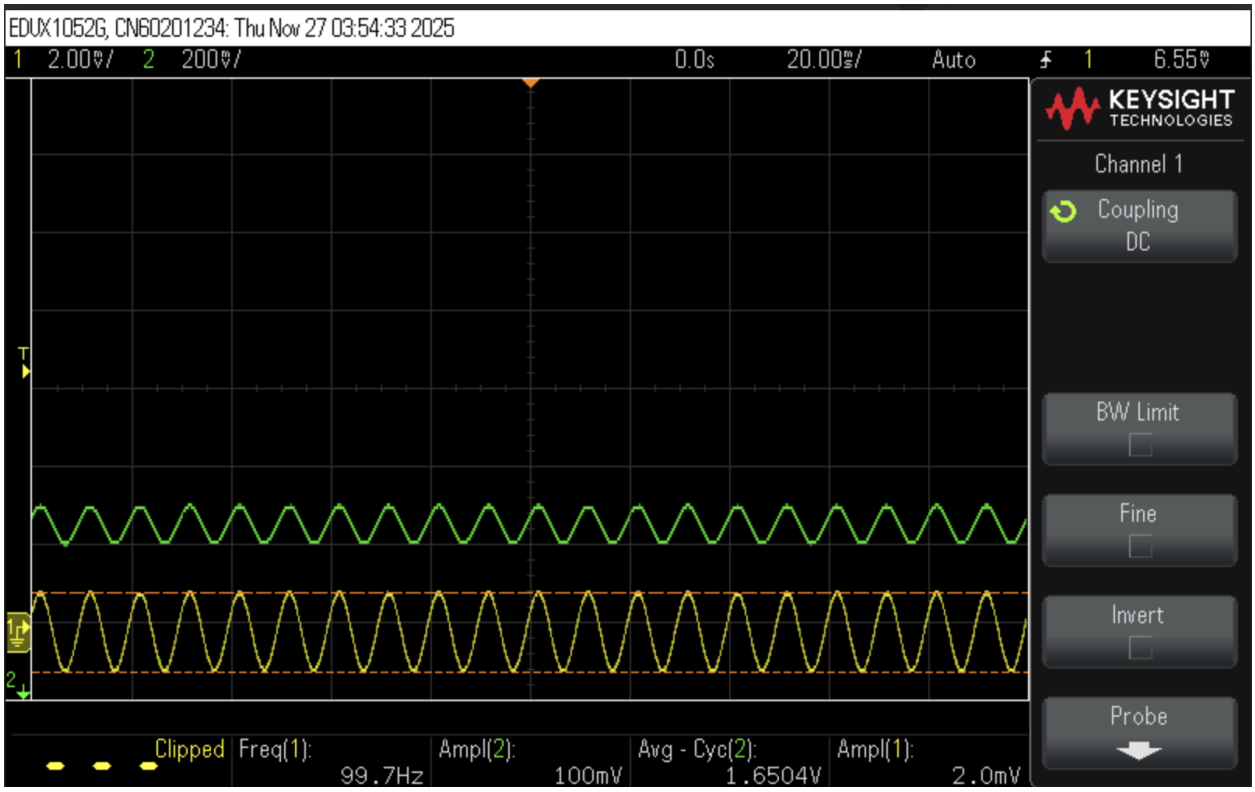
Virtual Ground:

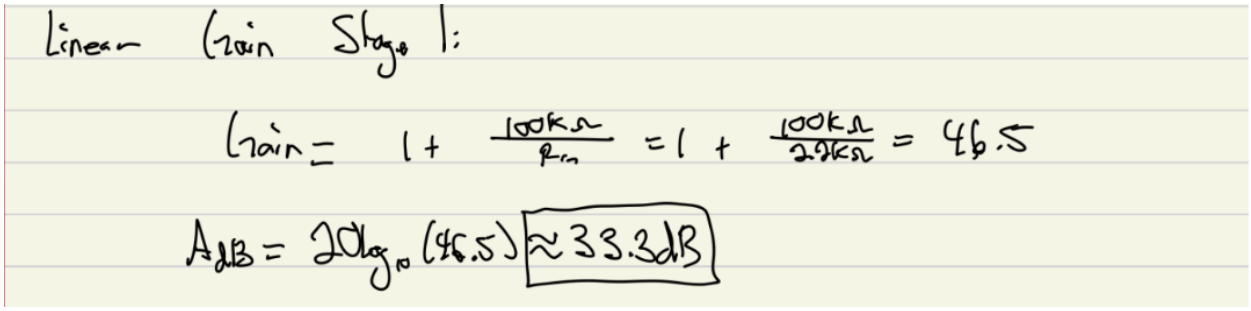
Max power transfer when $R_g = R_{11}$ → chose $10k\Omega$


3.5MM JACK
BREAKOUT
TIP
RING2
RING1
SLEEVE
Soldered

2. For stage 1 include:

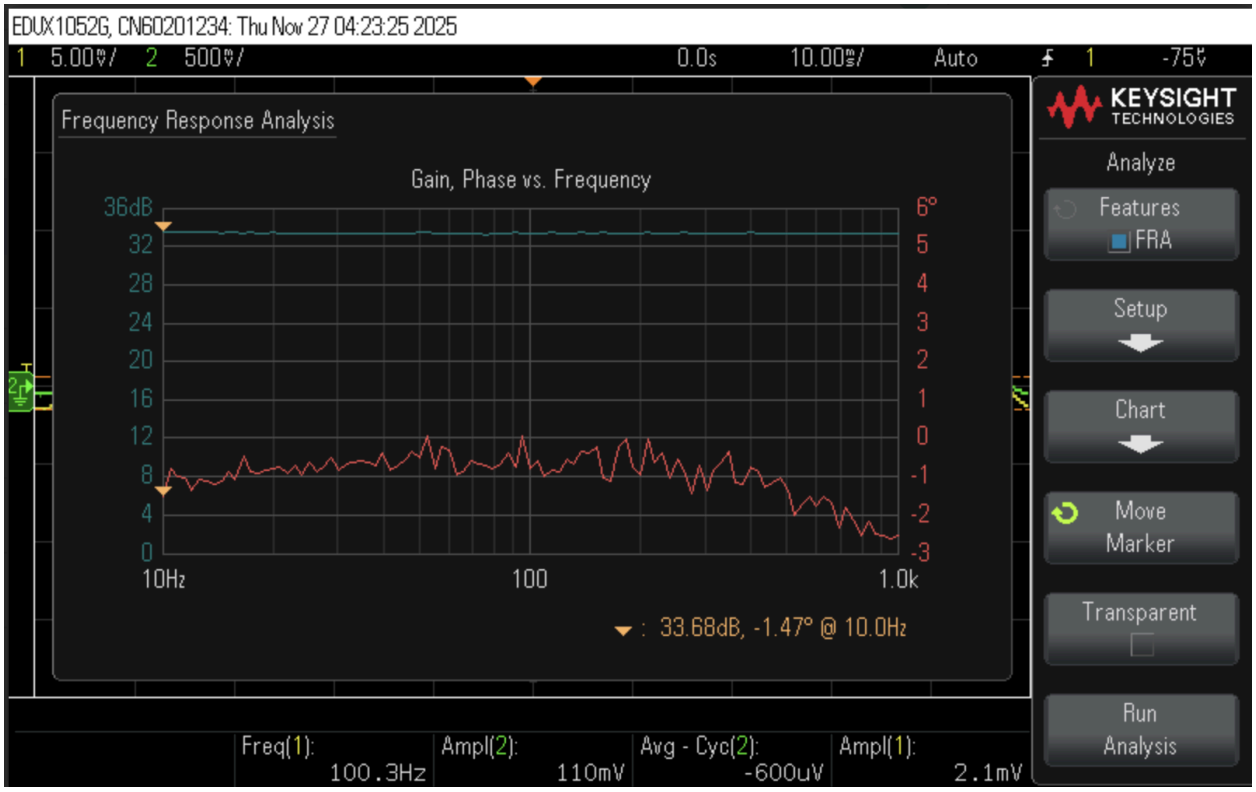
a. The screen grab of the sinusoidal input and output with the calculated gain from amplitude. Also report the gain in units of dB.





Our measured gain was (100mV/2mV) = 50. Our gain in dB is around 34 dB. This is very close to our expected gain which should've been 46.5 and our gain in dB which is 33.3 dB.

b. The Bode plot over the range 10Hz to 1kHz.



i. Report the maximum gain of your plot.

The maximum gain of our plot was approximately 33.68 dB which matches up with our expected of around 34 dB.

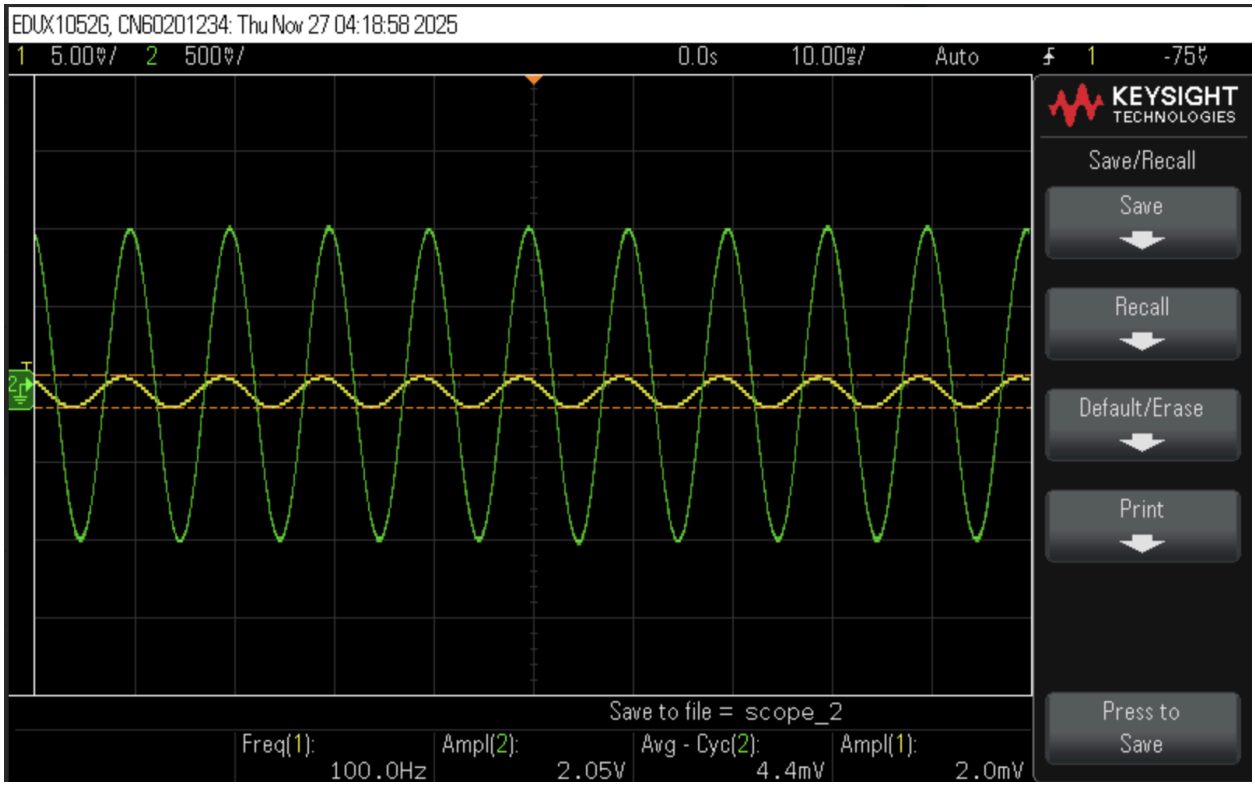
3. For stage 2 include:

a. The screen grab of the sinusoidal input and output with the calculated gain from amplitude. Also report the gain in units of dB.

Linear Gain Stage 2:

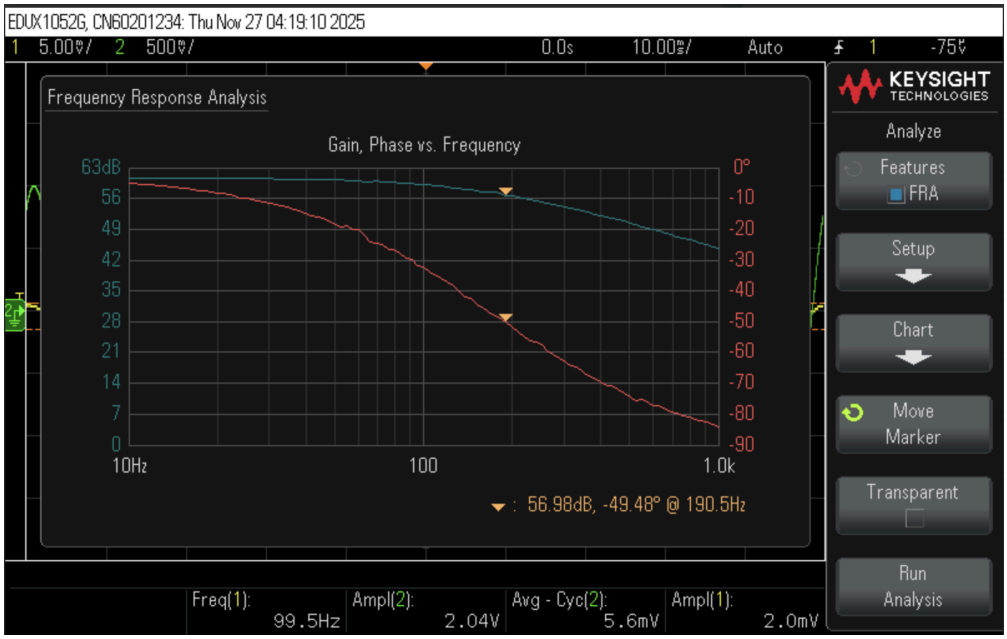
$$\text{Gain} = 1 + \frac{270\text{k}\Omega}{12\text{k}\Omega} = 23.5$$

$$A_{dB} = 20\log_{10}(23.5) \approx 27.5\text{dB}$$



The gain here is (2.05V/2.0mV) = 1025 which in dB is 60.2 dB. This checks out with our calculation as our stage 2 should add another 27.5 dB of gain to our stage 1 to get a total expected gain of 60.7 dB which is very close to what we got.

b. The Bode plot over the range 10Hz to 1kHz.



i. **Report the maximum gain of your plot.**

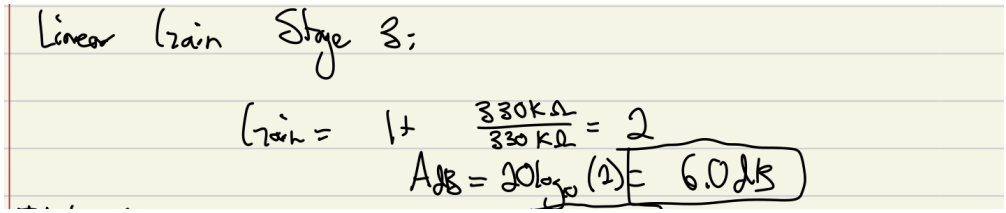
Our maximum gain here is 61 dB which is exactly what we expected from our calculation.

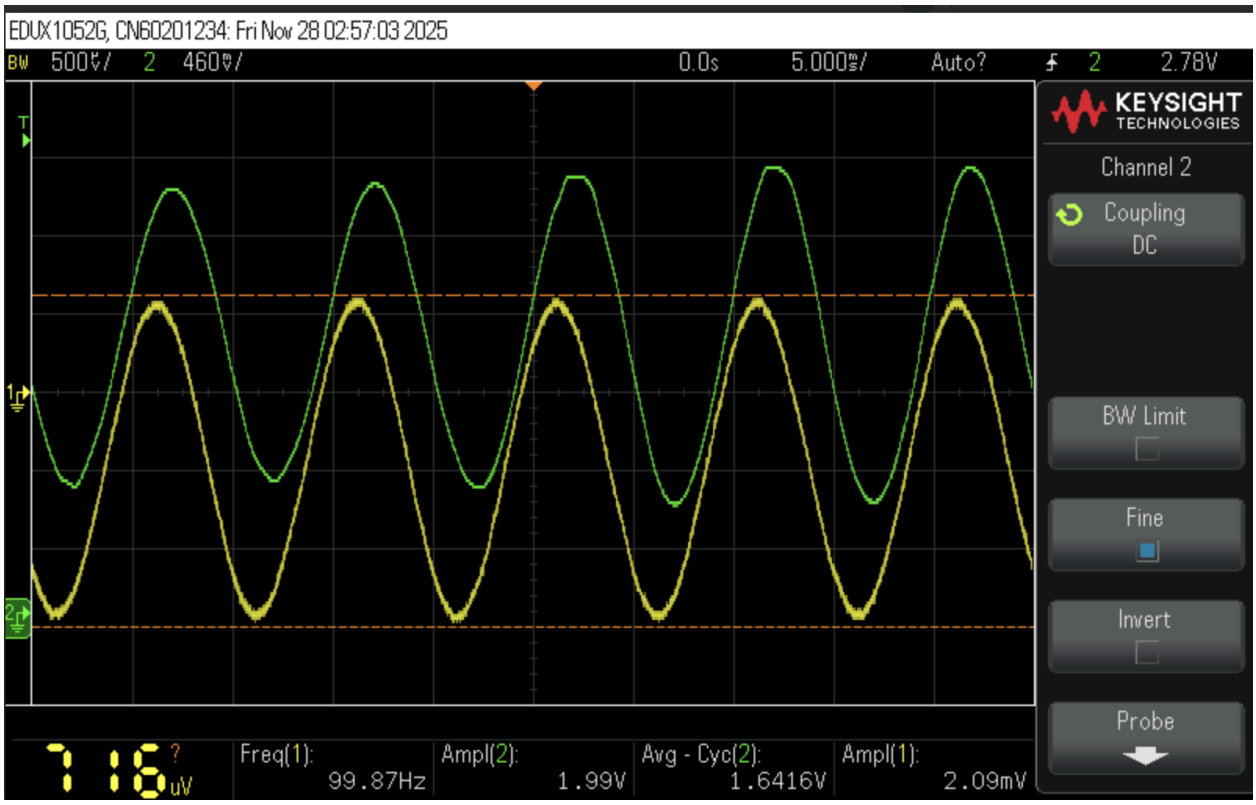
ii. **Report the 3dB drop-off gain and frequency.**

The 3dB drop-off gain is 56.98 dB at a frequency of 190.5 Hz. This represents the drop as a result of crossing the threshold for cutoff frequency. At first 190 Hz seems kind of high but it falls within the range of acceptable frequency as a result of error from our resistor and capacitor. Thus, this is exactly what we want to see.

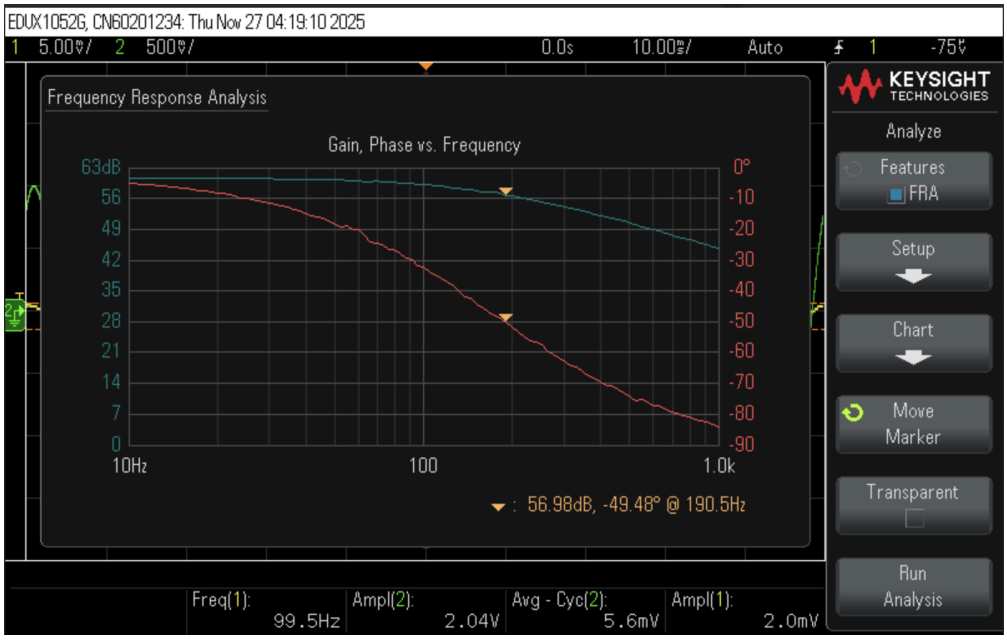
4. For stage 3 include:

a. **The screen grab of the sinusoidal input and output with the calculated gain from amplitude. Also report the gain in units of dB.**

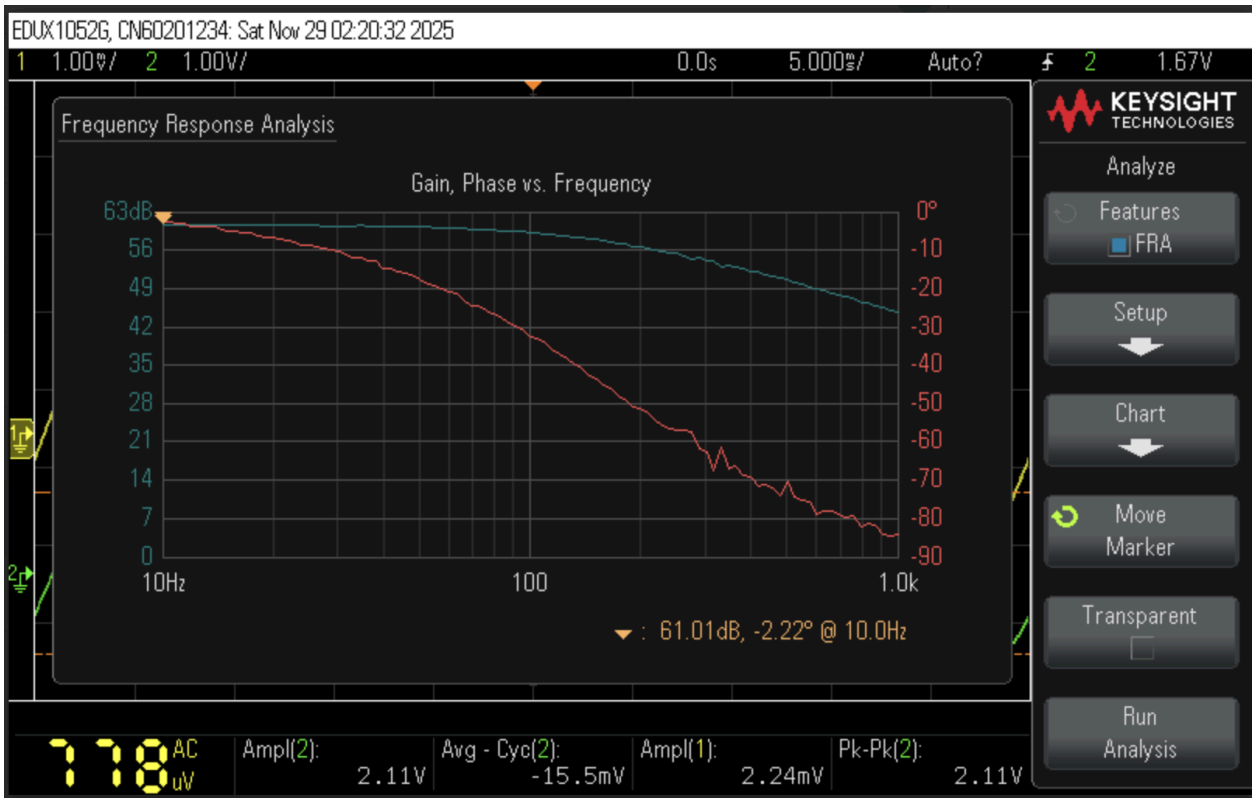




b. The screen grab of the sinusoidal input and output showing the 3dB drop-off gain for the low cut-off frequency. Show the pk-pk amplitudes and frequency on the screen grab.

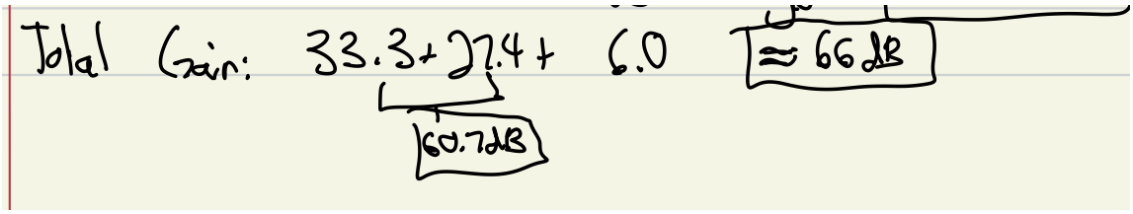


c. The Bode plot over the range 10Hz to 1kHz.



i. **Report the maximum gain of your plot.**

The maximum gain of our plot is around 61.01 dB very similar to our stage 2 since there is minimal gain after we run the signal through the high pass filter. We can attribute the error here to the

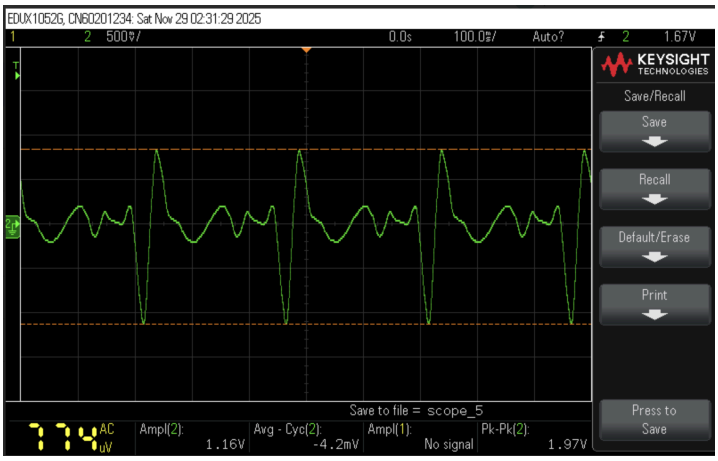
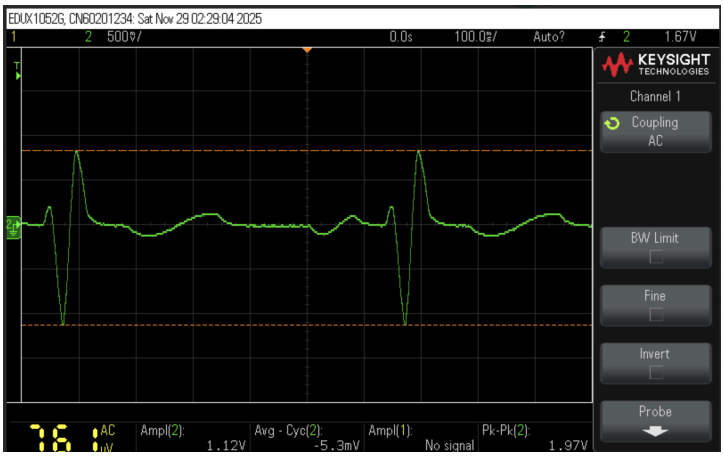
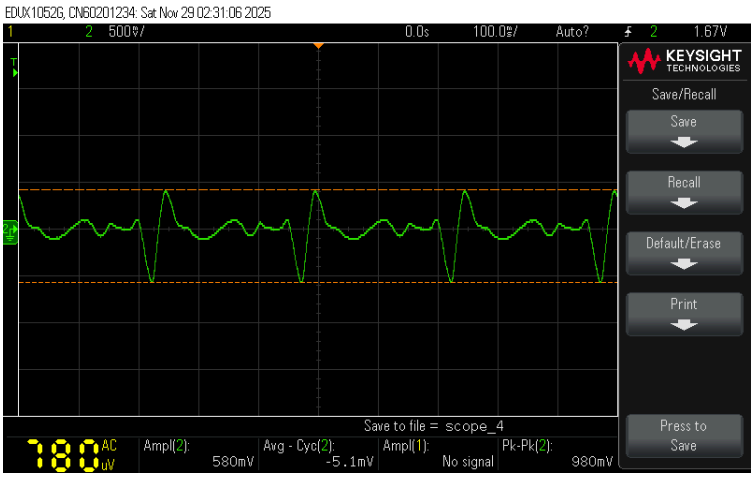
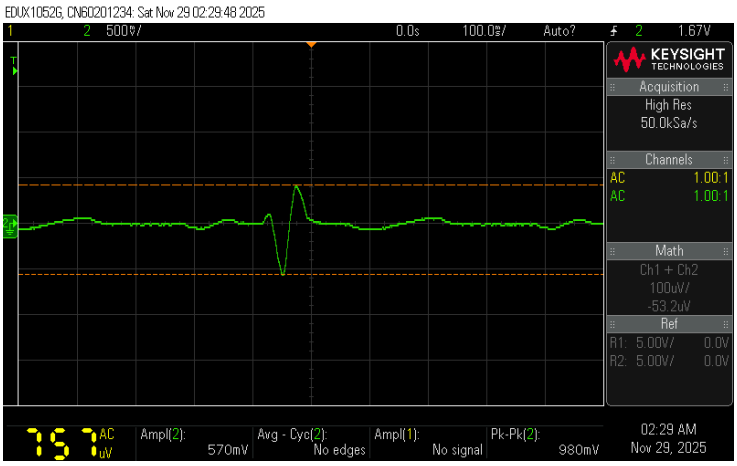


ii. **Report the 3dB drop-off gain and high cut-off frequency.**

The high-cut off frequency doesn't happen on the screen as we start at 10 Hz. The 3dB drop off is similar as it is in stage 2, it's around 58 dB.

5. Screen shot of your two different ECG input and output scope screen shots with amplitude measurements.

These are our 2 difference ECG input/output scopes with amplitude measurements. We chose to capture a normal heart rate (around 60-80 resting) and a super-fast heart rate of 240BPMs over an amplitude of 1 and 2 mV.

240 BPM = 4Hz & 2 mV	**80 BPM = 1.33 Hz & 2 mV**
	
240 BPM = 4Hz & 1 mV	**80 BPM = 1.33 Hz & 1 mV**
	

6. Calculate the gain for your two input signals. Also report the gains in units of dB.

The gain for the first one is 580 and the second one is 560. We make this assumption given that the complete signal goes through the circuit and into the oscilloscope. This however isn't the case as we can see in the above 2 examples, we see that only part of the signal gets through which we can attribute to the low-pass filter cutting off some of the total amplitude.

The gain for both should be 1000 as both frequencies are within our frequency bandwidth but we get lower than expected because some of the wave gets cut off by the low-pass filter.

Part B:

1. Summarize what you observed on the serial monitor using V!! = 3.3V and what the min/max adc values were.

With the function generator set to Vpp=3.3 V, the serial monitor showed that the ADC was using essentially its full range: the readings varied from a minimum of about 0 (0 V) to a maximum of about 4.095 (≈4.095 V, i.e., ADC code ≈4095).

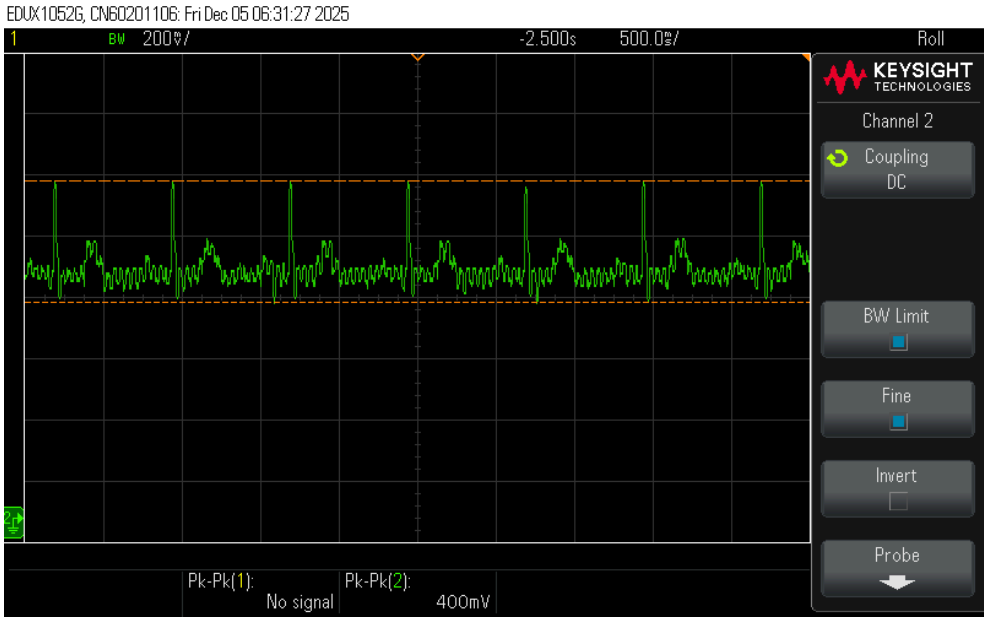
2. Summarize what you saw when you changed the V!! to be larger and smaller. Explain what you think is happening in the ADC.

For smaller Vpp=2.5 V the signal no longer used the full ADC range: on the serial plot the values stayed between about 0.358 V and 3.583 V, which corresponds to roughly 350 and 3580 ADC counts (out of 0–4095).
For larger Vpp=4.5 V the ADC output was stuck at 0 and 4.096 V (codes 0 and 4095) and the tops and bottoms of the waveform were visibly clipped.

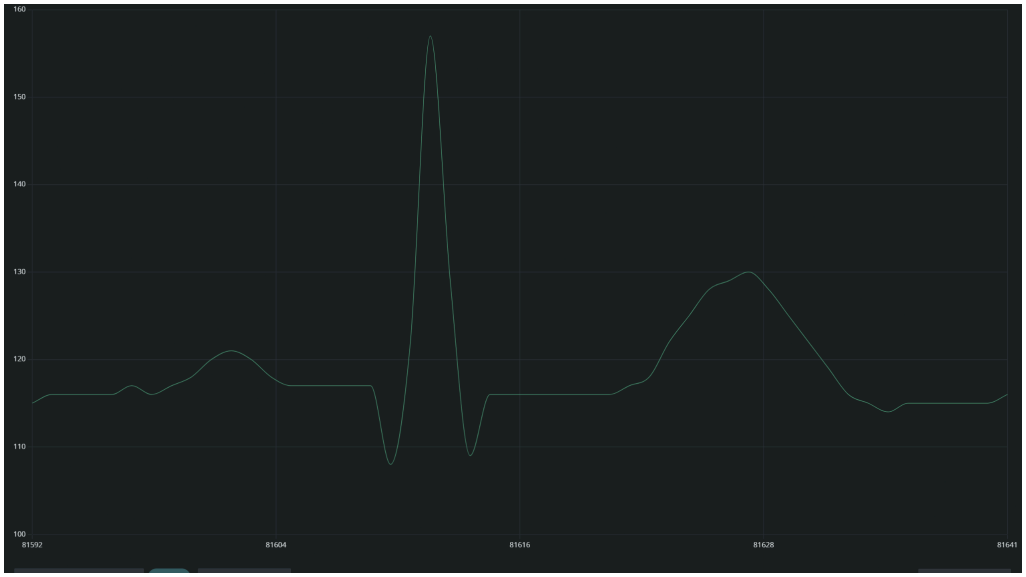
So when the input amplitude is smaller than the ADC full-scale it just occupies a smaller code range, and when the input is larger than full-scale the ADC saturates at its 0-V and 4.096-V limits and cannot represent the extra amplitude.

3. Include the scope capture of your OutputPin (A1) pin that verifies the BPM you chose. Report the BPM and the peak-to-peak voltage of your ECG signal.

BMP Chosen: 60BPM's and Peak-2-Peak = 400mV

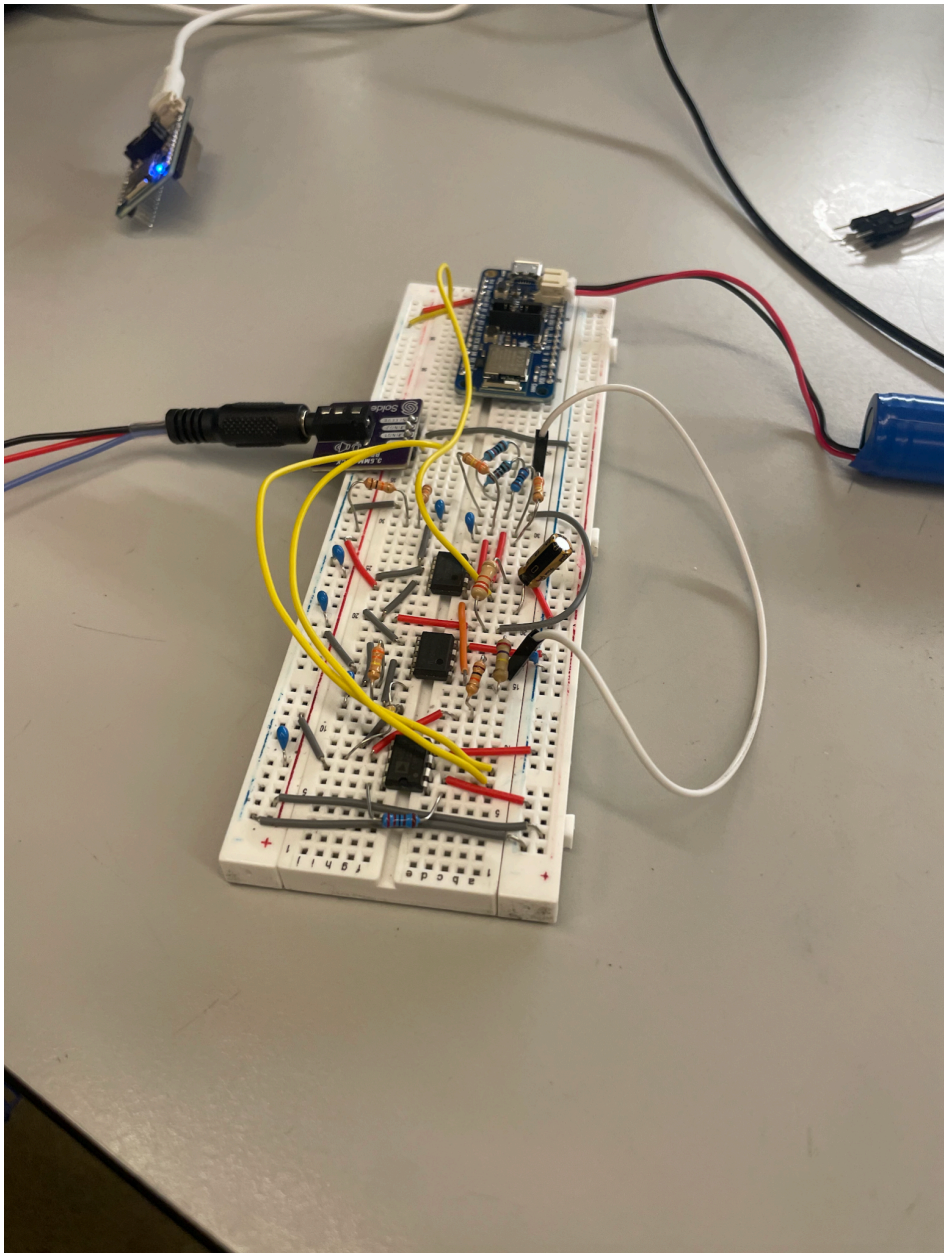


4. Include the serial plotter screen shot showing the ECG signal with at least one full PQRST wave.



Part C:

1. **Include a picture of your board with a central feather and peripheral feather. Summarize operation and communication between the feathers.**

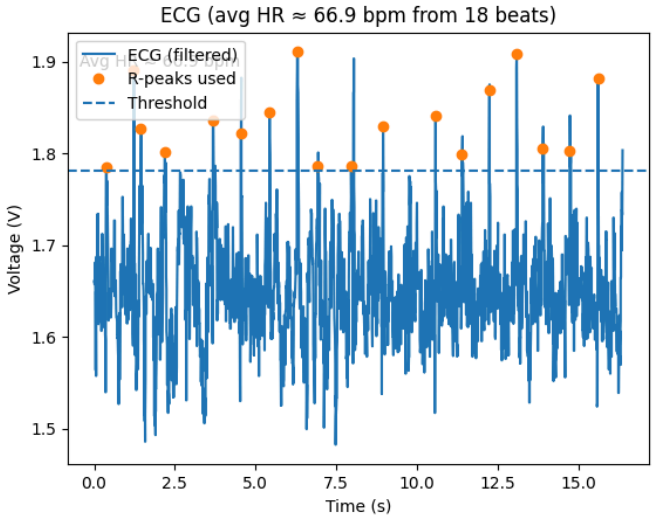


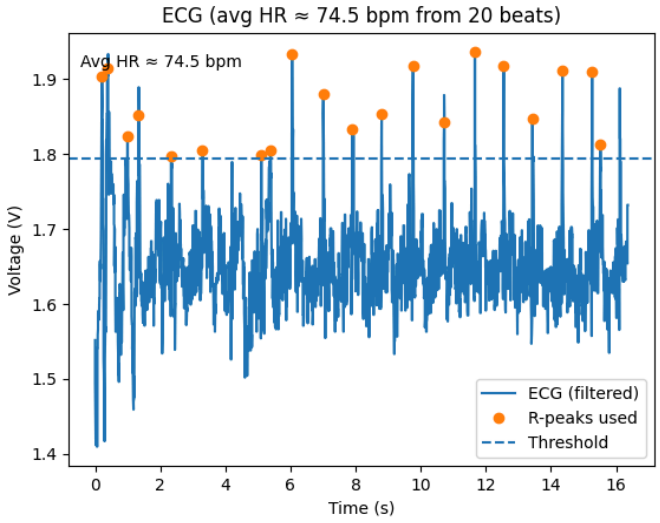
Essentially the way this system worked is we had our circuit on the breadboard that we've aforementioned that has a signal run through it via the heart rate monitor or the electrodes on the body. The output of this goes to the microcontroller which then transmits this data to our other feather via bluetooth which is then analyzed on our computer via the Arduino IDE using python. To be clear the peripheral board is running

circuit python aka code.py and the central feather is running arduino codey. Then we run serial _sample.py which takes the captured data and plots it via matplotlib.

2. **Submit a video (or demonstrate on demo day) of you being able to capture an ECG signal (either with electrodes or the simulator) and capture the data using wireless transmission. Upload your videos here: https://upenn.app.box.com/folder/352410464848 (upload your cool videos for us!)**

   Screenshots of captured ECGs during **demo day**:





3. **Describe your code written to measure the HR. How does it perform with different ECG amplitudes and heart rates?**

Our Python script opens the Feather's serial port, reads ECG samples with timestamps into arrays, then converts them to NumPy arrays and median-filters the ECG to remove spikes while preserving sharp R-peaks. It estimates the baseline and maximum ECG values and sets a dynamic threshold halfway between them, then scans the filtered signal for local maxima above this threshold that are separated by at least a minimum number of samples derived from an assumed maximum heart rate. From the times of the first 20 detected R-peaks it computes HR as 60 (Nbeats−1)Δt and plots the ECG with peaks marked while printing the HR. Because the threshold is relative to the baseline and max, the method is fairly insensitive to overall ECG amplitude and still works when we change the simulator amplitude, as long as the signal stays clearly above noise. For normal heart rates (roughly 60–120 bpm) it tracks the programmed value within a few bpm; at very low amplitudes or very high/irregular rates, peaks can be missed or merged, so the HR estimate becomes less accurate.

Code used:

```
import serial
import time
import matplotlib.pyplot as plt
import numpy as np

# Define the serial port the Feather is connected to
port = "COM9"

# Define the number of samples you want to collect
N = 4000

# Initialize index at 0
i = 0

# Define arrays for data
ecg = []
time = []

# Open port
feather = serial.Serial(port, baudrate=115200, timeout=1)

k = N
while k >= 0:
    # Data format: ["ESE 3400", ecg, temp, accel_x, accel_y, accel_z]
```

```
    data = feather.readline().strip()
    # Line below is needed for Python 3; comment out for Python 2
    data = str(data,'utf-8')
    if i == 1:
        ecg.append(float(data))
        i += 1
    elif i == 2:
        time.append(float(data))
        i += 1
    if data == "ESE 2150":
        k -= 1
        i = 1

# Close port
feather.close()

# ---------- Convert to numpy arrays ----------
ecg = np.array(ecg, dtype=float)
t   = np.array(time, dtype=float)

# Make sure they are the same length (just in case)
L = min(len(ecg), len(t))
ecg = ecg[:L]
t   = t[:L]

# Zero the time axis (start at 0 s)
t = t - t[0]

# ---------- Llight median filter ----------
def median_filter(x, window=5):
    half = window // 2
    x_pad = np.pad(x, (half, half), mode="edge")
    y = np.empty_like(x)
    for i in range(len(x)):
        y[i] = np.median(x_pad[i:i+window])
    return y

ecg_filt = median_filter(ecg, window=5)
```

```
# ---------- Simple R-peak detection ----------
baseline = np.mean(ecg_filt)
max_val  = np.max(ecg_filt)
threshold = baseline + 0.5 * (max_val - baseline)   # tweak 0.4-0.6
as needed

print(f"baseline={baseline:.3f}, max={max_val:.3f},
threshold={threshold:.3f}")

# Estimate sampling period and set minimum distance between beats
if len(t) > 1:
    dt = float(np.mean(np.diff(t)))   # seconds per sample
else:
    dt = 0.004                        # fallback if something weird
happens

max_hr = 360.0                        # maximum bpm we want to
support
min_rr = 60.0 / max_hr                # minimum time between beats
(s)
min_samples_between = max(1, int(min_rr / dt))
print(f"dt={dt:.4f} s, min_samples_between={min_samples_between}
samples")

peak_indices = []
last_peak = -10_000_000

for n in range(1, len(ecg_filt) - 1):
    # local maximum above threshold
    if (ecg_filt[n] > threshold and
        ecg_filt[n] > ecg_filt[n - 1] and
        ecg_filt[n] >= ecg_filt[n + 1]):

        # enforce minimum spacing so we don't double-count one spike
        if n - last_peak >= min_samples_between:
            peak_indices.append(n)
            last_peak = n

print("Total peaks detected:", len(peak_indices))
```

```
# ---------- Use first 20 beats to compute HR ----------
beats_to_use = min(20, len(peak_indices))

if beats_to_use >= 2:
    used = peak_indices[:beats_to_use]
    duration = t[used[-1]] - t[used[0]]        # seconds between 1st
and last of the 20
    avg_hr = 60.0 * (beats_to_use - 1) / duration
else:
    avg_hr = float("nan")

print(f"Using first {beats_to_use} beats, avg HR ≈ {avg_hr:.1f} bpm")

# ---------- Plot ECG and mark used peaks ----------
plt.figure()
plt.plot(t, ecg_filt, label="ECG (filtered)")

if beats_to_use > 0:
    used = peak_indices[:beats_to_use]
    plt.plot(t[used], ecg_filt[used], "o", label="R-peaks used")

plt.axhline(threshold, linestyle="--", label="Threshold")
plt.title(f"ECG (avg HR ≈ {avg_hr:.1f} bpm from {beats_to_use}
beats)")
plt.xlabel("Time (s)")
plt.ylabel("Voltage (V)")
plt.legend()
plt.text(0.02, 0.95,
         f"Avg HR ≈ {avg_hr:.1f} bpm",
         transform=plt.gca().transAxes,
         va="top")
plt.show()
```

4. **Discuss and difficulties and how you overcame them to measure and display the heart rate.**

   We had a variety of difficulties when measuring heart rate. Somewhere placing the electrodes on the right part of the body such that we could get a good heart rate that wasn't inhibited by hair or too much skin which adds its own impedance which is

significant for such a small signal. Further, we had to hold our body in a certain position and not move so that the signal could flow through clearly without speaking. This wasn't something we did at the beginning of the lab so it was something we realized later. Moreover, our code wasn't taking as many samples as we intended, as a result our arduino was detecting fewer peaks than the necessary 20. Last but definitely not least, our feather was having a hard time actually connecting to the other feather and transmitting the data. This was just a bluetooth problem but is definitely something engineers need to consider when building their EKG especially in a high stress environment like an ambulance.

5. **Discuss how things changed when using electrodes instead of the ECG simulator. What new considerations might have to be made with capturing a human ECG? What new considerations might have to be made to capture an ECG with arrhythmias?**

   When we used electrodes, it was way harder for us to get a signal and the reason for this was because of the placement of the electrodes but also our heart rate has an amplitude of very close to 1mV which is extremely small. At least the heart rate monitor was directly connected and provided perfect data, as a result we were able to see perfect results, the electrodes however were catching really weak analog data from Dammi's body. When we use a human body, we have to consider safety. AKA, we can't plug our system to a human heart and a wall socket can be dangerous. Further, we have to consider input impedance from putting electrodes on one's skin. This means that we need to add a very large resistor at the beginning so that the impedance of the skin is irrelevant for that first op-amp. Arrhythmia is where the heart beats irregularly. We would simply need to increase our window aka number of samples just so that we can get all of the relevant data related to the heartbeat. Since our data collection is decently continuous there are no worries there.