Music Therapy

Laurynas Zavistanavicius, Anna Martelli, Patrick Condon, Ryan Schenck

Design Review 1

Meeting: Thursday, February 22nd 9:15 AM

Requirements:

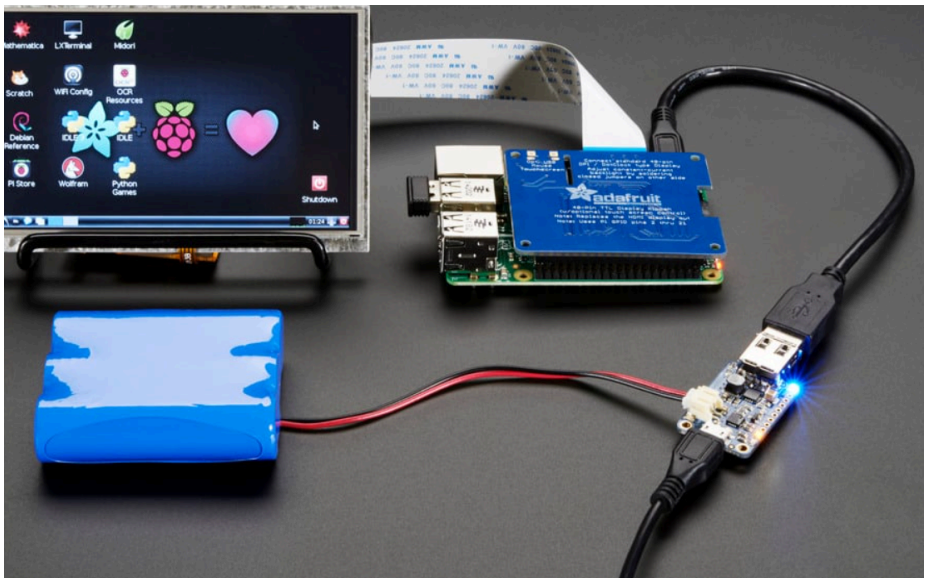
- Provide a detailed design of each major subsystem.
- Give a detailed description of all major components and describe the function they serve in your design and how the devices realize the required function.
- Specify essential connections on all major components. These would include all power and ground connections (with appropriate values for device voltage and expected current requirement), decoupling, and other essential support connections such as clocking, programming, etc.
- There are likely to be a set of problems that you are not clear on how you will solve them. Give a list of these items and an action plan to reduce them to solved problems.

Subsystems:

- Power
- Gait Tracking
- Gait Analysis
- Music Storage
- Music Processing
- Music Output
- User Interface

Subsystem: Power and Regulation

For our system, we will use the 5V unregulated VIN pin on the voltage regulator on our PCB, which will be regulated to 3.3V suitable for the ESP32–S3 microcontroller. The 5V input, when we are programming, will come from the USB port of the programming computer. When operating in its functional state, the 5V will come from a boosted power circuit that we are designing.

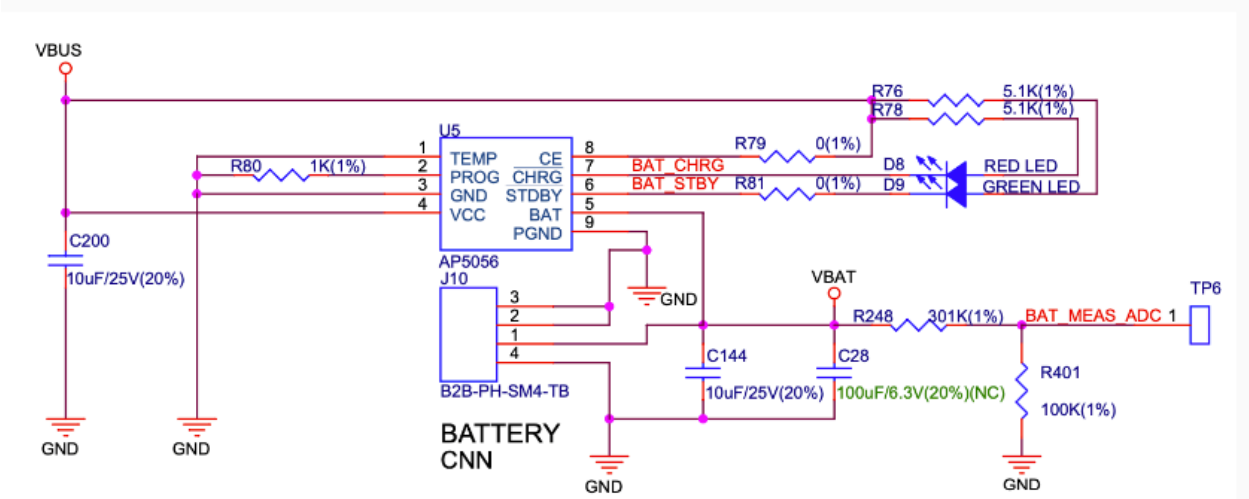


In the above photo, the battery pack (which we have purchased) is a 3.7V LiPo battery. It will be fed via a 2-pin JST to the Adafruit Powerboost 1000 Charger circuit, linked below.

https://www.adafruit.com/product/2465

The Powerboost will boost the 3.7V battery to 5V, which will be fed to our touchscreen user interface (which requires 5V), the audio power amplifier, and the VIN pin on our voltage regulator. An additional MicroB connection on the Powerboost allows the battery pack to be recharged at will.

The below figure shows a possible configuration for a battery charging system using a 3.7V battery and 5V from USB. In order to modify this for our use, we would integrate the Powerboost to step up from 3.7V to the 5V necessary to power components of the board when USB is not connected.



ESP32-S3-Korvo-2 V3.0 - Power Supply from a Battery

Devices:

- 3.7V LiPo battery
    - 1000 mAh
    - Rechargeable
- Powerboost 1000
    - 3.7 V -> 5V, 1 A output

POWER - NEW INFO

.

Datasheet of boost converter circuit using above part - 5V fixed output
https://www.ti.com/lit/ds/symlink/tps61202.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1709567756259&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Ftps61202

11.2.2.1 Programming the Output Voltage

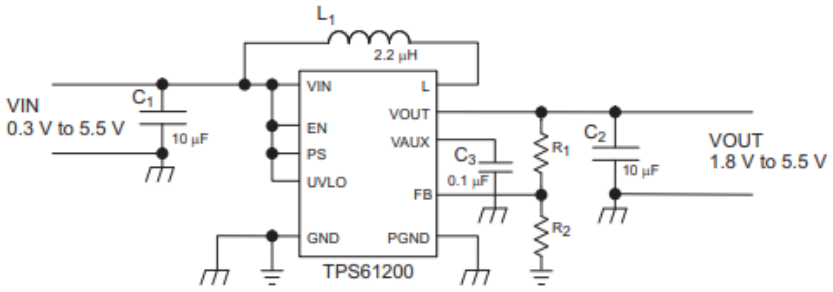
Within the TPS6120X family, there are fixed and adjustable output voltage versions available. To properly configure the fixed output voltage devices, the FB pin is used to sense the output voltage. This means that it must be connected directly to VOUT. For the adjustable output voltage version, an external resistor divider is used to adjust the output voltage. The resistor divider must be connected between VOUT, FB and GND. When the output voltage is regulated properly, the typical value of the voltage at the FB pin is 500 mV. The maximum recommended value for the output voltage is 5.5 V. The current through the resistive divider should be about 100 times greater than the current into the FB pin. The typical current into the FB pin is 0.01 µA, and the voltage

across the resistor between FB and GND, R2, is typically 500 mV. Based on those two values, the recommended value for R2 should be lower than 500 kΩ, in order to set the divider current at 1 µA or higher. It is recommended to keep the value for this resistor in the range of 200 kΩ. The value of the resistor connected between VOUT and FB, R1, depending on the needed output voltage (V_{OUT}), can be calculated using Equation 1:

$$R1 = R2 \times \left(\frac{V_{OUT}}{V_{FB}} - 1 \right) \quad (1)$$

As an example, for an output voltage of 3.3 V, a 1-MΩ resistor should be chosen for R_1 when a 180-kΩ is selected for R_2.

4 Typical Application



Subsystem: Gait Tracking

Add more specifics on connections and I2C *

The BNO055 is a nine axis sensor fusion device. It combines a 16 bit gyroscope with a 14 bit accelerometer and a geomagnetic sensor. We have determined that this part will

accurately track our patient's gait, with the accelerometer and gyroscope functions counterbalancing and error correcting for each other.

Essential Operating Conditions:

OPERATING CONDITIONS BNO055

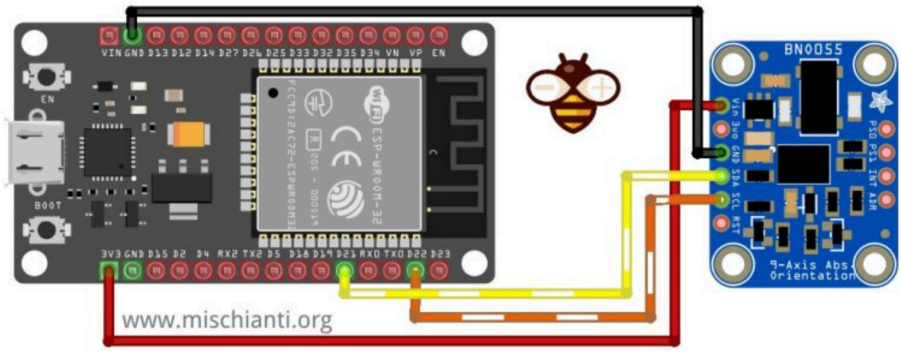
Parameter	Symbol	Condition	Min	Typ	Max	Unit
Supply Voltage (only Sensors)	V_{DD}	--	2.4	--	3.6	V
Supply Voltage (µC and I/O Domain)	V_{DDIO}	--	1.7	--	3.6	V
Voltage Input Low Level (UART, I2C)	V_{DDIO_VIL}	V_{DDIO} = 1.7-2.7V	--	--	0.25	V_{DDIO}
		V_{DDIO} = 2.7-3.6V	--	--	0.3	V_{DDIO}
Voltage Input High Level (UART, I2C)	V_{DDIO_VIH}	V_{DDIO} = 1.7-2.7V	0.7	--	--	V_{DDIO}
		V_{DDIO} = 2.7-3.6V	0.55	--	--	V_{DDIO}
Voltage Output Low Level (UART, I2C)	V_{DDIO_VOL}	V_{DDIO} > 3V , I_{OL} =20mA	--	0.1	0.2	V_{DDIO}
Voltage Output High Level (UART, I2C)	V_{DDIO_VOH}	V_{DDIO} > 3V , I_{OH} =10mA	0.9	0.8	--	V_{DDIO}
POR Voltage threshold on VDDIO-IN rising	V_{DDIO_POT+}	V_{DDIO} falls at 1V/ms or slower	--	1.45	--	V
POR Voltage threshold on VDDIO-IN falling	V_{DDIO_POT-}		--	0.99	--	V
Operating Temperature	T_A	--	-40	--	+85	°C
Total supply current normal mode at T_A (9DOF @100Hz output data rate)	I_{DD} + I_{DDIO}	V_{DD} = 3V, V_{DDIO} = 2.5V	--	--	12.3	mA
Total supply current Low power mode at T_A	I_{DD_LPM}	V_{DD} = 3V, V_{DDIO} = 2.5V	--	--	0.4	mA
Total supply current suspend mode at T_A	I_{DD_SuM}	V_{DD} = 3V, V_{DDIO} = 2.5V	--	--	0.04	mA

I2C layout:
https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bno055-ds000.pdf
Datasheet

d

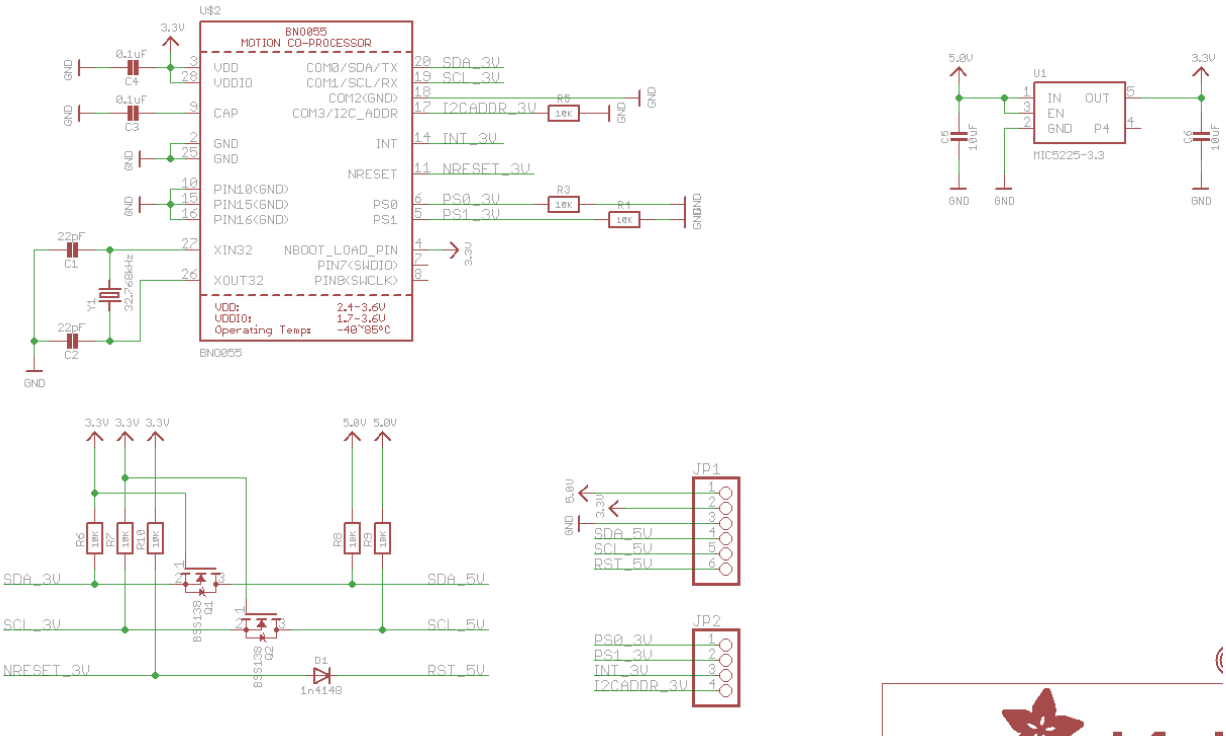
Requirements:

- 3.3 V input, which will be fed from our regulated power circuit. The BNO055 itself will be onboard our PCB, with the associated I2C passives shown above.

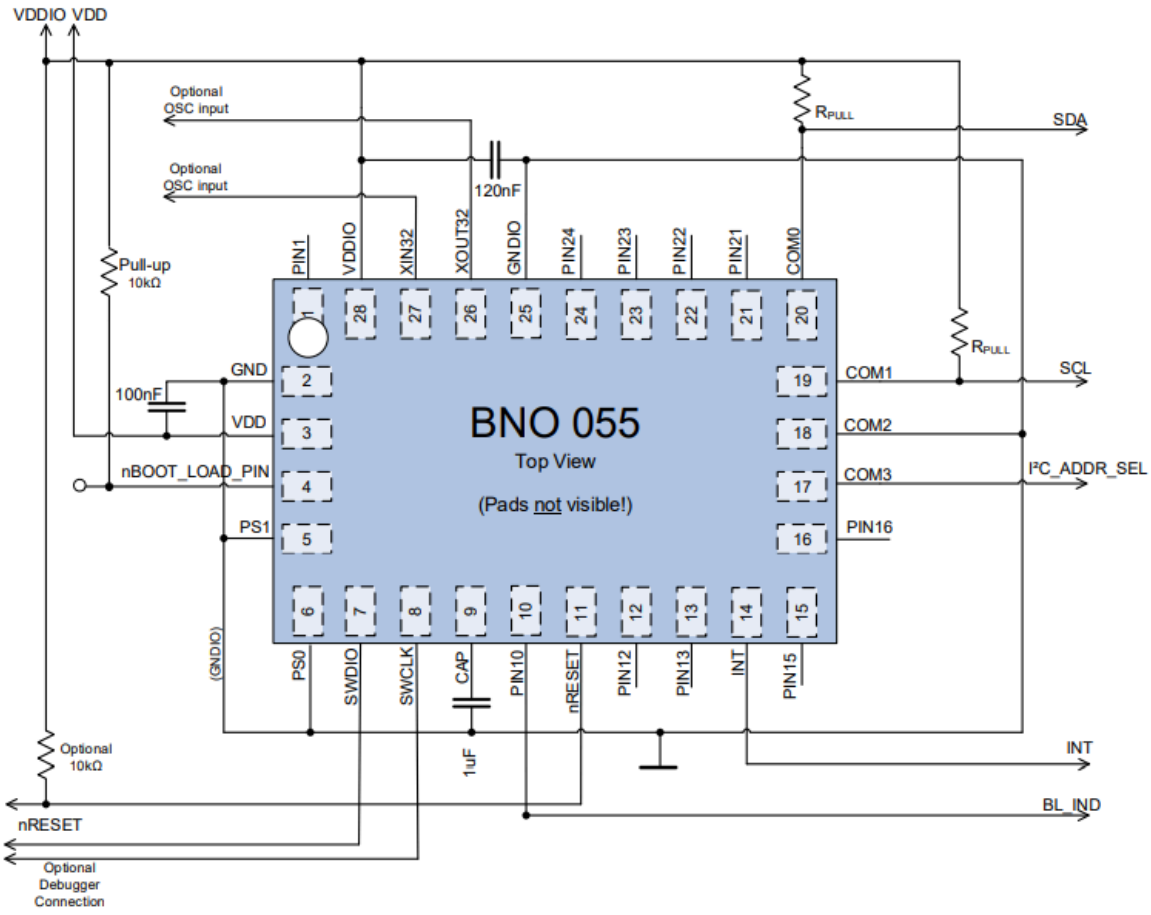


This is an example of the Adafruit BNO055 breakout board being connected to an ESP32-C3. In our pcb, we will be using the S3 microcontroller, which has the following pins available for I2C.

Pin No.	RTC / Analog IO Name	RTC Function			
		0	1	2	3
5	RTC_GPIO0	RTC_GPIO0			sar_i2c_scl_0
6	RTC_GPIO1	RTC_GPIO1			sar_i2c_sda_0
7	RTC_GPIO2	RTC_GPIO2			sar_i2c_scl_1
8	RTC_GPIO3	RTC_GPIO3			sar_i2c_sda_1



Schematic for Adafruit BNO055 breakout board



The breakout board itself has connections as shown in the above schematic. We will model our pcb's connections after the connections used in Adafruit's breakout board, modified to fit the scheme of the ESP32-S3 microcontroller.

Subsystem: Gait Analysis

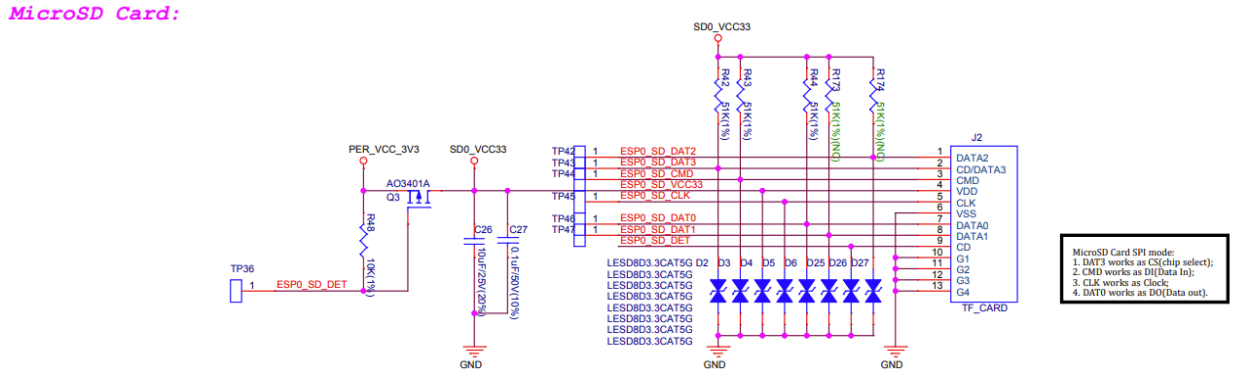
Our gait analysis, the software processing side of gait tracking, will be programmed on our final board. The program will read rotation values in the x,y, and z axes and determine (based on pre-inputted parameters for the patient's height and size) if a step has been taken. In a rolling-average manner, the patient's gait will be determined by how many steps are measured over time. This pace value, compared with desired values, will be sent to the Music Processing subsystem.

Current Code design:

An Adafruit_BNO055 library exists on GitHub, and we have downloaded and tested sample code on the Adafruit BNO055 breakout board. We are working on manipulating the code, and using an example main file that reports simple x,y, and z values. Once we discern how to accurately retrieve these values, our program will then have to find the gait “speed” to report to the music processing side of things. This “speed” will be reported in steps per minute. This will be done by timing how long it takes for the user to take 5 steps (we will have to iterate to determine the proper number). Then we will take (60 seconds/x amount of time) and multiply it by 5 (or whatever number of steps we land on) to reach the desired steps per minute output.

Subsystem: Music Storage

The ESP32-S3_WROOM-1 has 8MB of flash and 8MB of PSRAM built into the chip. An external flash will not be connected and an SD card will be used to provide the necessary storage for the audio files. We will use a microSD card in 1 bit mode over SPI, with the schematic shown below.



The microSD card needs 3.3V which will use the same power path as the ESP32.

Subsystem: Music Processing

The following description of the music processing algorithms is based on Jonathan Driedger and Meinard Müller: [TSM Toolbox: MATLAB Implementations of Time-Scale Modification Algorithms].

The tempo of the music will be modified using time-scale modification (TSM) algorithms. The goal of TSM algorithms is to stretch a time-varying signal by a stretching factor *alpha* while minimizing changes to the frequency spectrum content. The original signal is divided into analysis frames on the order of milliseconds, and each analysis frame is processed by the TSM into a synthesis frame such that the length of the synthesis frame divided by the length of the analysis frame equals *alpha*.

TSM algorithms can be based in the time domain or frequency domain. One class of algorithms based in the time domain is Overlap-Add (OLA). In OLA, synthesis frames are created by windowing adjacent analysis frames; this method works well for transient, percussive sounds, but introduces significant distortion to harmonic signals. TSM algorithms based in the frequency domain are typically referred to as phase vocoders (PV). In PV, the FFT of each analysis frame is computed (typically within a STFT), and the extracted frequency and phase information is then used in an IFFT to create a synthesis frame of different length; this method works well for harmonic sounds, but smears out transient, percussive sounds. The optimal solution, proposed by Driedger and Müller, is to separate the source signal into harmonic and percussive components, apply PV and OLA algorithms to each component, respectively, and then recombine.

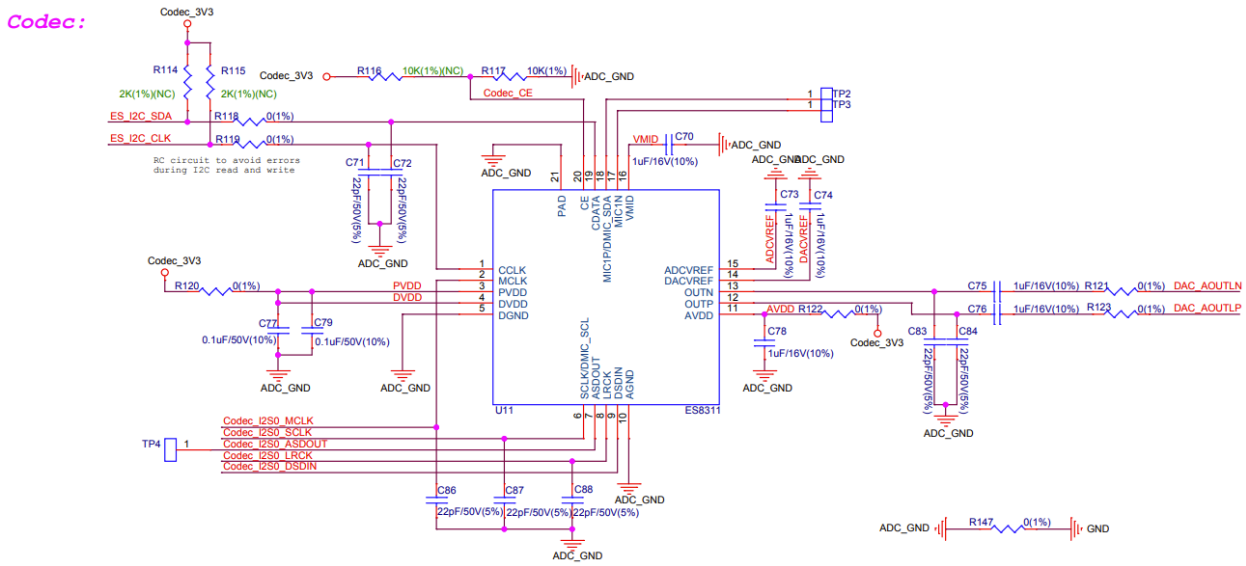
Our anticipated data stream is as follows:

```
sdcard ---> fatfs_stream ---> wav_decoder ---> time_stretch---> i2s_stream ---> codec_chip
```

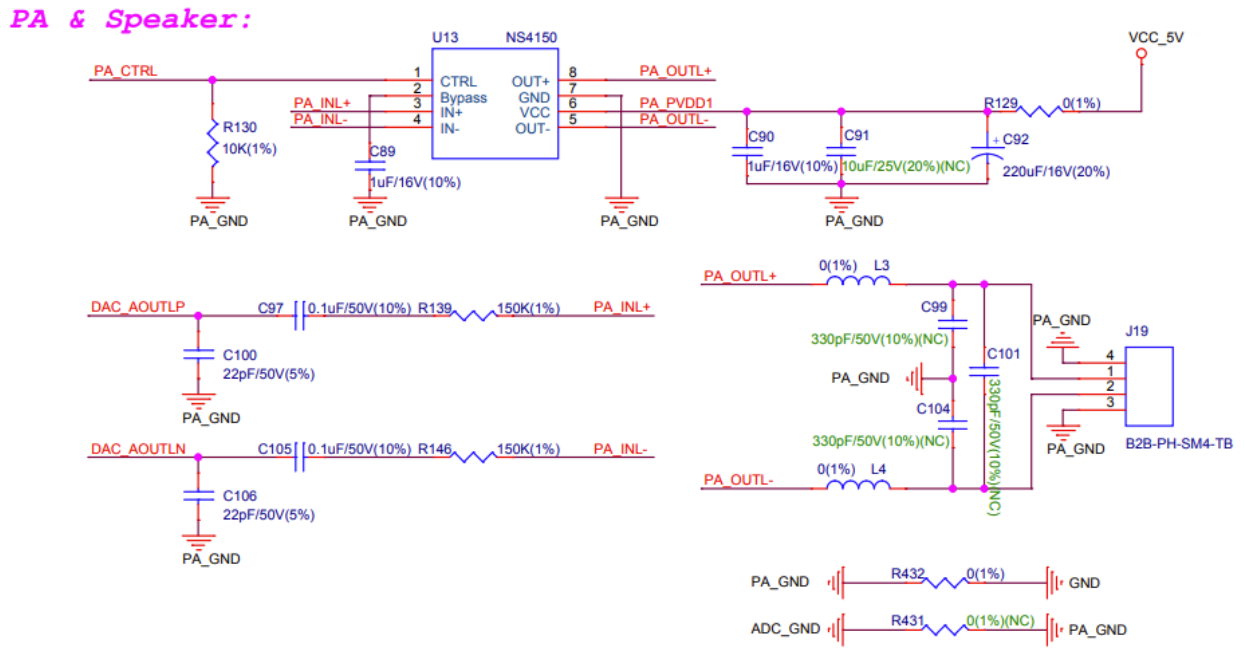
The stretch factor *alpha* will be continuously updated with the resulting values in the gait analysis. *alpha* will be computed as the tempo of the song in bpm divided by the current walking speed in bpm; for example, if the song is at 60 bpm and the walking speed is at 40 bpm, then the stretch factor will be 1.5, so the length of the audio signal will be 1.5 times longer and sound that much slower.

Music Output

The music output will consist of a codec, audio power amplifier, and a speaker connected using a 3.5mm audio jack. The audio codec chip, ES8311, is a low-power mono audio codec. It consists of 1-channel ADC, 1-channel DAC, low noise pre-amplifier, headphone driver, analog mixing, and gain functions. It is interfaced with ESP32-S3-WROOM-1 module over I2S and I2C buses to provide audio processing in hardware independently from the audio application. The audio PA chip, NS4150, is an EMI, 3 W mono Class D audio power amplifier, amplifying audio signals from audio codec chips to drive speakers. The schematic for the codec is shown below.



Schematic for the audio power amplifier and speaker is shown below.

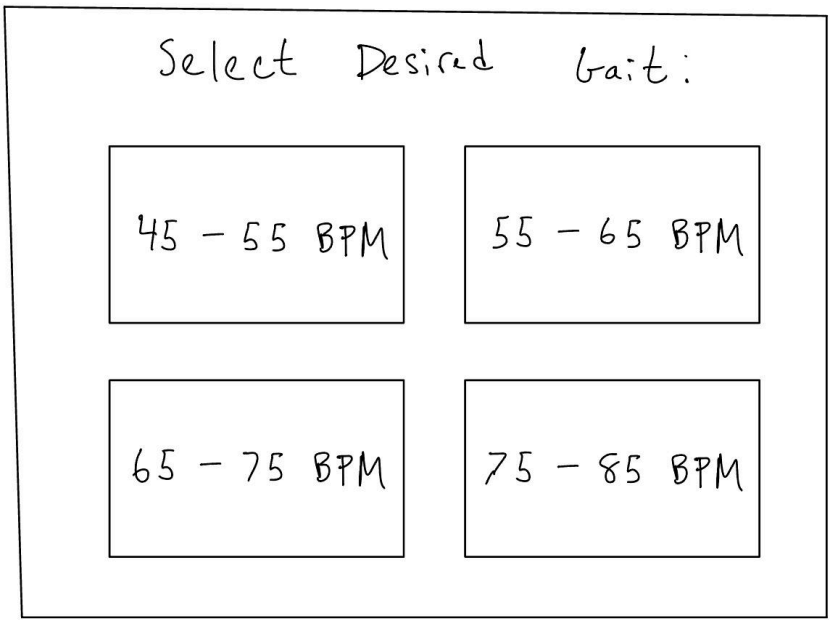


Our aim is to allow for connection of both headphones and speaker through the audio jack, allowing the user to choose their preferred output. The codec and the PA are powered by the regulated 3.3V. The speaker schematic has a jst connector for the speaker which we will swap for a 3.5mm ⅛” audio jack with tip, ring, and sleeve (TRS) connections, corresponding to positive signal to tip, negative signal to ring, and ground to sleeve. We might need to swap for TRRS if using certain headphones with another ring for the microphone.

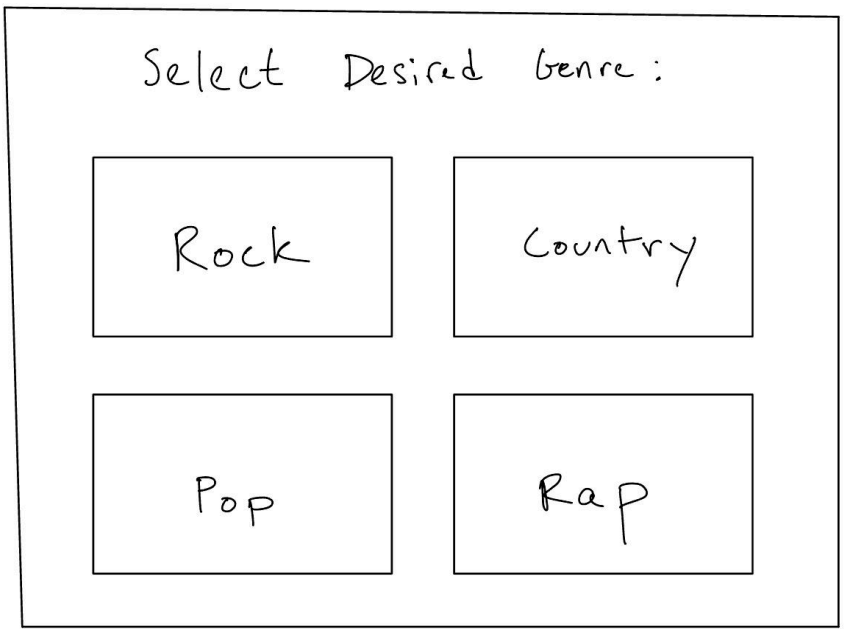
Subsystem: User Interface

Given our target user base of older people with limited mobility, our goal is to make the user interface as simple as possible. It will take only four simple steps to initialize the device after it is turned on.

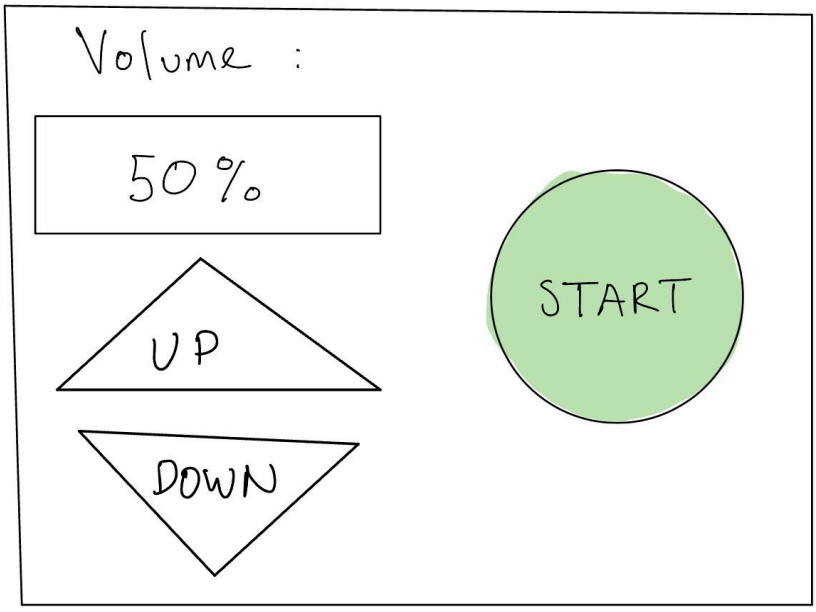
1) *Gait speed selection*: The average gait speed is around 80-100 steps per minute. Given our user base of physically challenged patients, we plan on offering steps/minute ranges that range from significantly slower to about average. We will offer steps per minute ranges of 45-55, 55-65, 65-75, and 75-85. These ranges exist such that the songs aren't limited to exactly 50 bpm, 60 bpm, 70 bpm, or 80 bpm, however the targets are those numbers in the middle of the range. A rough sketch of the interface display follows:



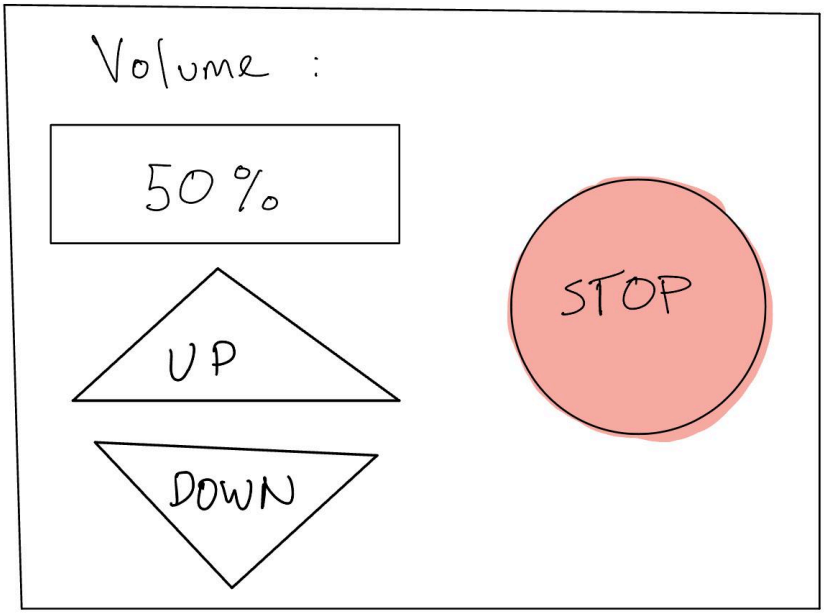
2) *Music selection*: We would intend for the final product to allow the user to upload their own songs, which would then be sorted by BPM. Our version, however, does not know the users preferences, so we plan on having several playlists according to the different genres: rock, pop, country, and rap. For the time being, each playlist will have around 4 songs with the proper genre and corresponding beats per minute. A rough sketch of the interface display follows:



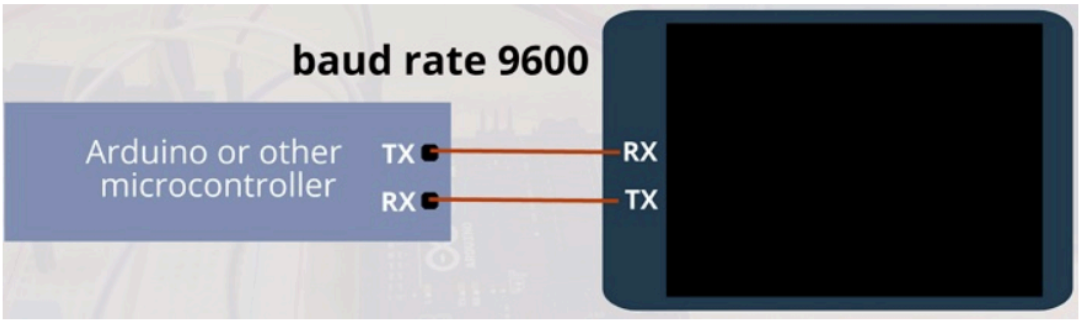
3) *Volume level selection and start*: This is imperative, especially given that our older customers may have diminished hearing levels. To keep it simple, we plan on having ten volume levels. The default will be 50% volume - it will increase and decrease in increments of 10%. The user will be able to adjust the volume with up and down using arrows on the screen. Once they have adjusted their volume to the desired start level, all the user has to do is press start. A rough sketch of the interface display follows:



4) *While walking - volume adjustment and stop*: While walking, the user needs to have control over two things: controlling the volume and being able to stop the device. These options will be on opposite halves of the screen:



Device: We have selected the Nextion 3.2” Basic HMI display. We chose this display because of the simplicity of its integration. Nextion has its own editor. This will make it easier for our team to create the large, simple graphics on the interface. Additionally, the sending and receiving information from the interface becomes more manageable as the only **essential connections** are **GND, RX, TX, and +5V**. This can be done with a basic 4 pin JST connector.



Basic connection between the interface and a microcontroller

	Test Conditions	Min	Typical	Max	Unit
Operating Voltage		4.75	5	7	V
Operating Current	VCC=+5V, Brightness is 100%	–	85	–	mA
	SLEEP Mode	–	15	–	mA
Power supply recommend: **5V, 500mA, DC**					

Electronic Conditions

Identified Problems:

1. Determining what values read from the BNO055 will dictate whether or not a step has been taken. How will the pedometer actually function?
    a. Action steps: research methods used when the device is placed on a patient’s hip, and experiment with the device once we have the BNO055 calibrated and outputting sample data.
2. Determining our method of music processing, i.e. if real-time audio stretching is possible or we need to switch to using pre-processed tracks?
    a. This will be an ongoing process, as we work through and manipulate the software.
3. Compiling our software into one readable program, using RTOS task management.

a. We need to research, learn, and watch tutorials on this process so that once we have each subsystem working properly, we are equipped to combine our individual programs into a working RTOS-based program.

Design Review 2 Subsystem Demonstrations

Gait Tracking & Gait Analysis: The ability to track walking pace will be demonstrated by using the BNO055 on an Adafruit breakout board, wired up on a breadboard. The breadboard will be connected via USB power to a laptop running the gait analysis program.

User Interface: The Nextion display will be wired up to a breadboard & connected to its program (which we will write) via USB power from a laptop. The ability to choose outlined parameters will be demonstrated.

Music processing: We are still answering questions related to how we can best & most feasibly stretch the music in real time (or use pre-processed tracks) but our current progress will be demonstrated via audio files run from a laptop.

Music storage, music output, power: These components are difficult to demonstrate effectively until our pcb is actually made, but our schematic design and physical purchased parts will be outlined.