149 Policy Stuff
Project overview:
https://docs.google.com/document/d/1VWgchZwb4YSkIeHLotjCA2AXnBVhZNE7uqNkVcOfhOA/edit?usp=sharing

Suggested topics:
https://docs.google.com/document/d/1RoVewBl50l4VtFgiusN8wlzIvoe46Q8aNZJjaZzmbjQ/edit?usp=sharing

Project PPT:
https://docs.google.com/presentation/d/16AebhlckoujpUPbl00nyk3TAmIgagr2u3vcp0YdT-YQ/edit?usp=sharing

Purchase form:
https://docs.google.com/forms/d/e/1FAIpQLScT6ZTXOtYE07OEw_1ZUuINhYrhzrluE4i6Y3jqlB95NIoXrA/viewform

Office hours:
https://docs.google.com/spreadsheets/d/1nBrlQSqXqiQu6S8UHKVsjnQo_O1q-a2VHX3x_QaUykE/edit?usp=sharing

Haptic Feedback
http://www.ti.com/lit/ml/slyt418/slyt418.pdf

http://www.ti.com/motor-drivers/actuator-drivers/products.html?DCMP=HAPTICDRIVER+Other&HQS=Other+OT+touch

http://www.ti.com/motor-drivers/actuator-drivers/actuators.html

TI sells drivers for haptic feedback actuators and has a list of actuator manufacturers they recommend. Based on the descriptions I think we would want to use some kind of disk-shaped piezo or motor actuator.

http://www.ti.com/motor-drivers/actuator-drivers/products.html?DCMP=HAPTICDRIVER+Other&HQS=Other+OT+touch

http://www.vibration-motor.com/ ← these guys are the stuff

https://www.digikey.com/ordering/shipping/shipcostestimator

https://github.com/NordicPlayground/nRF52-teensy-sgtl5000-audio

nRF52 Datasheet
http://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.4.pdf

nRF52DK Guide
http://infocenter.nordicsemi.com/pdf/nRF52_DK_User_Guide_v1.2.pdf

VSCode:
http://shadetail.com/blog/using-visual-studio-code-for-arm-development-introduction/

I2S:
http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.nrf52832.ps.v1.1%2Fi2s.html (Link from Josh) (Hardware layer with registers)
http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.nrf52832.ps.v1.1%2Fi2s.html (Software driver layer)
https://learn.adafruit.com/adafruit-max98357-i2s-class-d-mono-amp (tutorials)

Looks like we may want to find .WAV files for the instruments we're interested in, then read the values out of those .WAV files using python's wave library (cause of course python has a library to read .WAV files.) Then we somehow stream those values to the DAC.

WAV visualizer: https://twistedwave.com/online

This is the protocol our DAC board uses.

https://en.wikipedia.org/wiki/Piano_key_frequencies
http://www.szynalski.com/tone-generator/

Limited number of analog inputs (8)
Multiplexer (need to have sampled each one by end of cycle)
Dac + speaker
Use sine wave instead of triangle/square

nRF5 SDK (as in how to use all the Nordic libraries):
http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk%2Fdita%2Fsdk%2Fnrf5_sdk.html
For reference, we have SDK 15.0.0.
"nrfx" basically just means newer libraries than ones that just say "nrf."

If you want to add new libraries to your app:
Add the relevant headers to main.c with #includes.
If you get “symbol not found” linker errors, it’s probably because the libraries didn’t build, or if they did build, their bodies were #ifdefined out. To fix this:

- Edit “software/boards/buckler_revB/Board.mk” and add the relevant C files to the big BOARD_SOURCES += block. This will build the libraries
- Edit “software/boards/buckler_revB/app_config.h” and add the relevant ENABLE #defines. This will fix #ifdefined checks that disable entire libraries even if they’re built.

If you’re getting “no rule to make target <some_library>.o”, it might mean that <some_library> is header-only, and you actually shouldn’t have added <some_library>.c to Board.mk, because no such file exists.

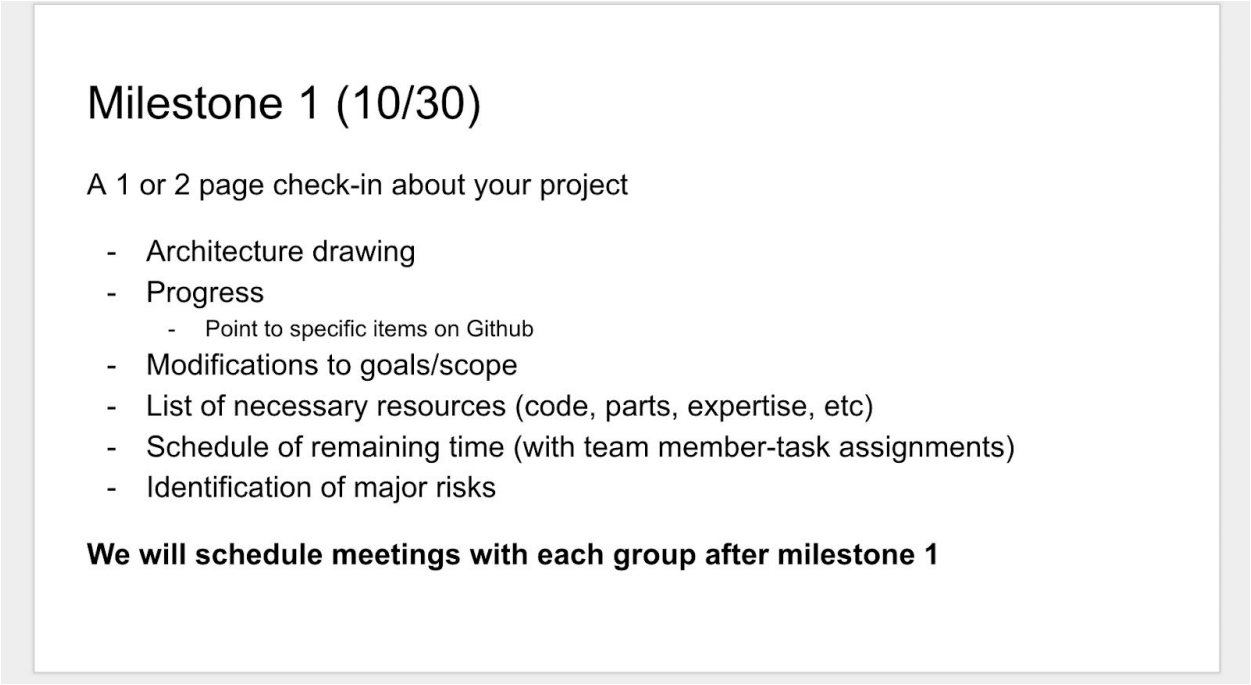
Important stuff we learned

11/2 - we tried to take 10000 samples at once for calibration - that overflows bc is greater than 2^16. Also we forgot to initialize the values to 0

11/12 - ADC cant take more than 2.4v after prescaling - you can give it 3.3v input but you have to prescale it down to less than 2.4v. (can’t use 5v output pin for anything that feeds back into the board)
can’t put more than 3.3v into any of the gpio pins

If you’re switching PWM frequencies rapidly, sound quality degrades with switching speed. Limiting it to one change every ½ second was better for sound, but makes it so you can’t play very fast.

Stuff from milestone 1 doc:



Milestone 1 (10/30)

A 1 or 2 page check-in about your project

- Architecture drawing
- Progress
  - Point to specific items on Github
- Modifications to goals/scope
- List of necessary resources (code, parts, expertise, etc)
- Schedule of remaining time (with team member-task assignments)
- Identification of major risks

We will schedule meetings with each group after milestone 1

Questions for GSI's

- How to attach flex sensors to gloves
  - Sew them
- How to attach to board
  - Sparkfun arduino protoshield
- Should we use buckler????? (or different accelerometer)
  - Analog accelerometer
- Breadboard?????

General notes

- Should we include chords, or just 12 notes is fine?
- HELP US

TODO

1. Figure out .wav stuff

https://learn.adafruit.com/adafruit-max98357-i2s-class-d-mono-amp/pinouts

https://github.com/KristofferKarlAxelEkstrand/AKWF-FREE/tree/master/AKWF

Open source Git repos we need to credit:
https://github.com/andenore/NordicSnippets/blob/master/examples/i2s_master/main.c

POSTER
Usage
Hardware

-

Software

- **Hardware Handling Layer**
- **Instrument Backend Layer**
- **Instrument Soundboard Layer**

Preprocessing

- **Arbitrary Soundboard Generation**

- **Project Purpose/Goals** (i.e. what's the point of the project)
- **Project Approach/Implementation**
    - Including hardware and/or software *architecture diagrams*
    - Be sure to describe algorithms, models, and interfaces that are important to your project
    - Also describe aspects of the project that were particularly challenging
- **Evaluation** (how well is your project performing)
    - You determine what metrics are most important for your project
    - Be sure to include figures describing its performance
- **Connections to Course Topics**
    - Projects should connect to three major course topics
    -

    - Type equivalence - replacing instrument or sound board
        - Instrument that sends signals over i2s and takes in instrument state struct and responds to a play function (you can program how it handles that info), can swap out instrument sounds
    - Interrupts - accelerometer
        - values do not exist on adc unless you send a sample message - so need to poll in order to get interrupts
    - Input/Output - analog input, GPIO
    - Sensors - flex sensors, accelerometer, rotary switch
    - Communication protocols - I2S?

Purpose/Goals

- **Goal:** Create a customizable and full-featured virtual instrument with a human interface
- **Implementation**
    - **Diagrams:**