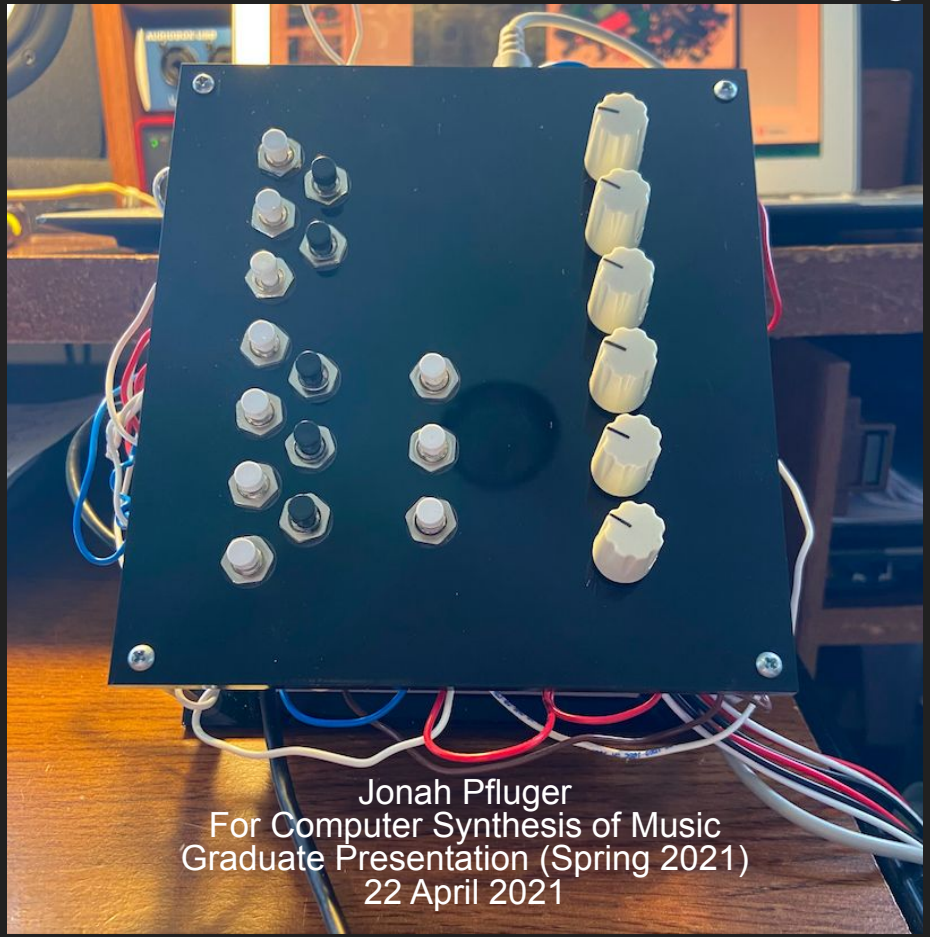
Building an Electronic Wind Synthesizer



Jonah Pfluger
For Computer Synthesis of Music
Graduate Presentation (Spring 2021)
22 April 2021

What have I done?

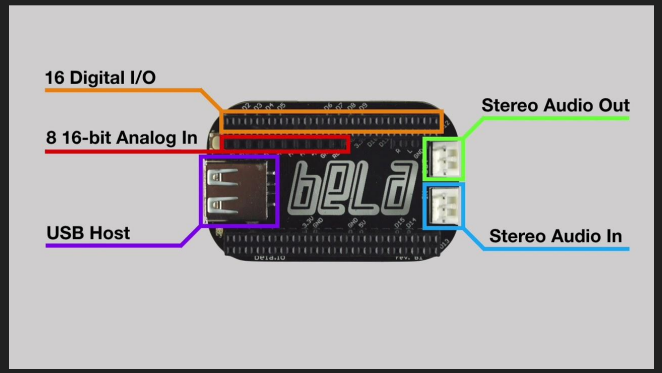
- Built an electronic wind (breath) - controlled synthesizer by
  - Designing and building the circuit
  - Laser cutting the acrylic (for hardware enclosure)
  - Developing the code (sometimes called “firmware” in this context) to run on the microprocessors/microcontrollers with Csound (and a small bit of C)
- Soon: finish composing a piece for this instrument (May 4)

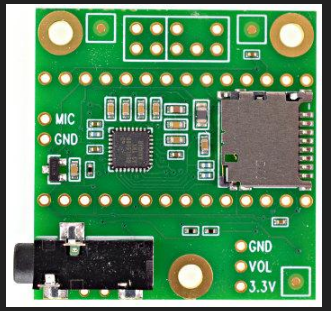
Functionality

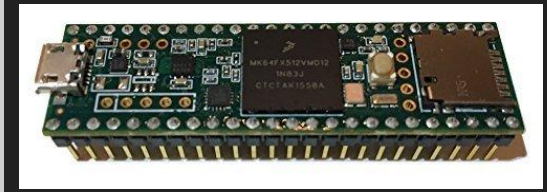
- Polyphonic
- “Microtonal”
- Utilizes amplitude data from breath control to control loudness of a synthesis algorithm
- Uses buttons to act as valves of the instrument
  - We’ll look into the code a bit more later

Hardware (Bill of Materials)

- Bela Mini
- Teensy 3.5
- Teensy Audio Shield
- 15 SPST Push Buttons
- 6 B10K Potentiometers
  - 6 Knobs
- 1 Electret Mic
- Lots o’ wire
- Acrylic for enclosure
  - + 4 Standoffs, 8 Screws
- (+ Soldering Materials)

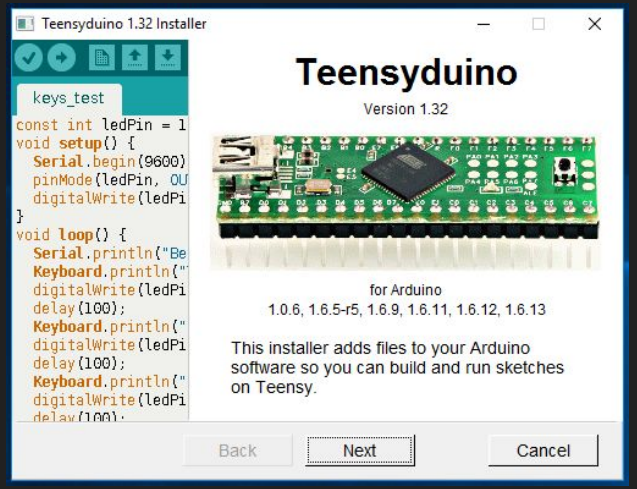


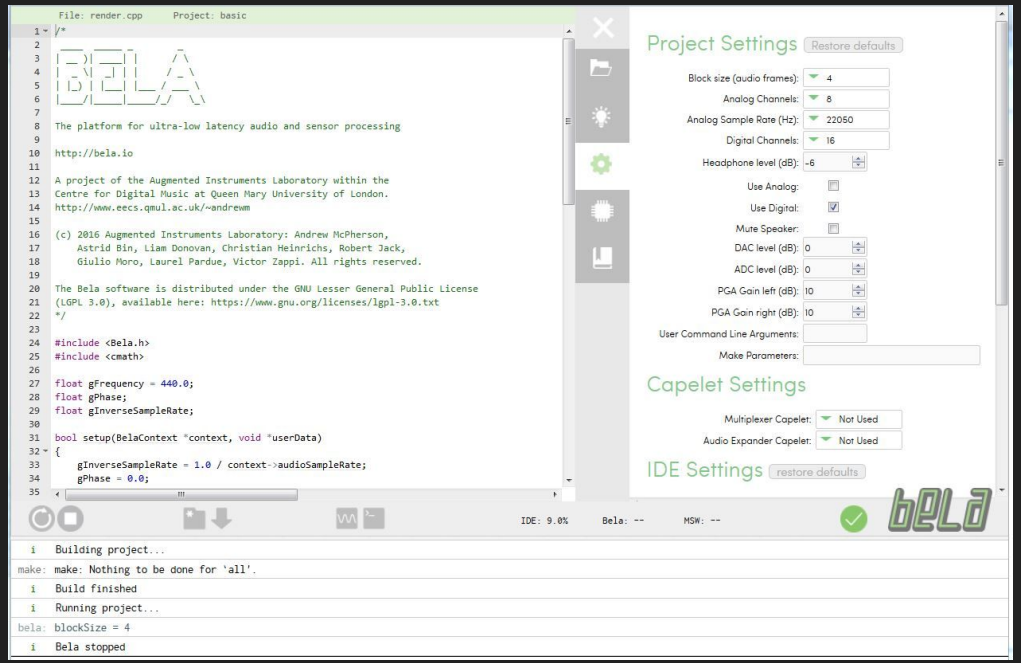




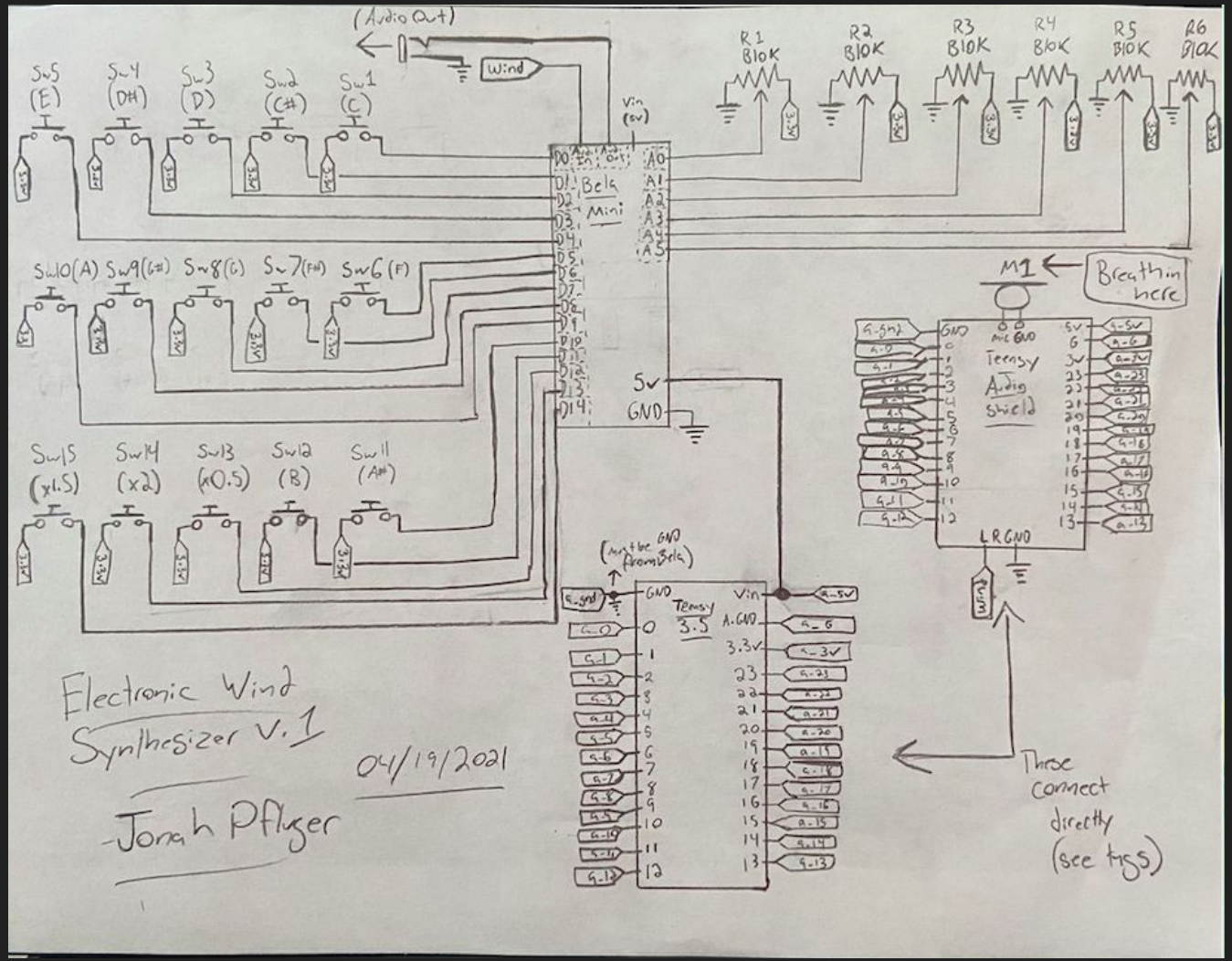
Software Used

- Bela IDE
  - Csound code written in this environment
  - Browser based IDE 😲
- Arduino/Teensyduino
  - C code written in this program





Schematic



Excuse the hand drawn nature of this

Pseudo-Code Explanation and Considerations

- Bring in data from sensors to Csound
  - First: breath data
    - Wind v.s. Sound… both pressure waves in air! = Microphone
    - Bela Mini does not have a preamp...
  - This is why we need help from the Teensy! (+ it's audio shield)

```
void setup() {
  AudioMemory(600);
  sgtl5000_1.enable();
  sgtl5000_1.volume(0.5);
  sgtl5000_1.inputSelect(AUDIO_INPUT_MIC);
  sgtl5000_1.micGain(2.4);
  sgtl5000_1.lineOutLevel(25);
  mixer1.gain(0, 0.707);
}
```

```
void loop() {
float rmsRead = rms1.read();
        if (rmsRead == 0.0){
                mixer1.gain(0, 0);
        }
        else{
                mixer1.gain(0, rmsRead);
        }
delay(100);
}
```

- In Bela:

```
aL, aR ins ; brings in audio from bela stereo input
aTrk  follow2  aL, 0.04, 0.5 ; amplitude tracking… for the curious: (ares follow2 asig, katt, krel)
aTrk tone aTrk, 10
aTrk = aTrk * aTrk
```

Pseudo-Code Explanation and Considerations

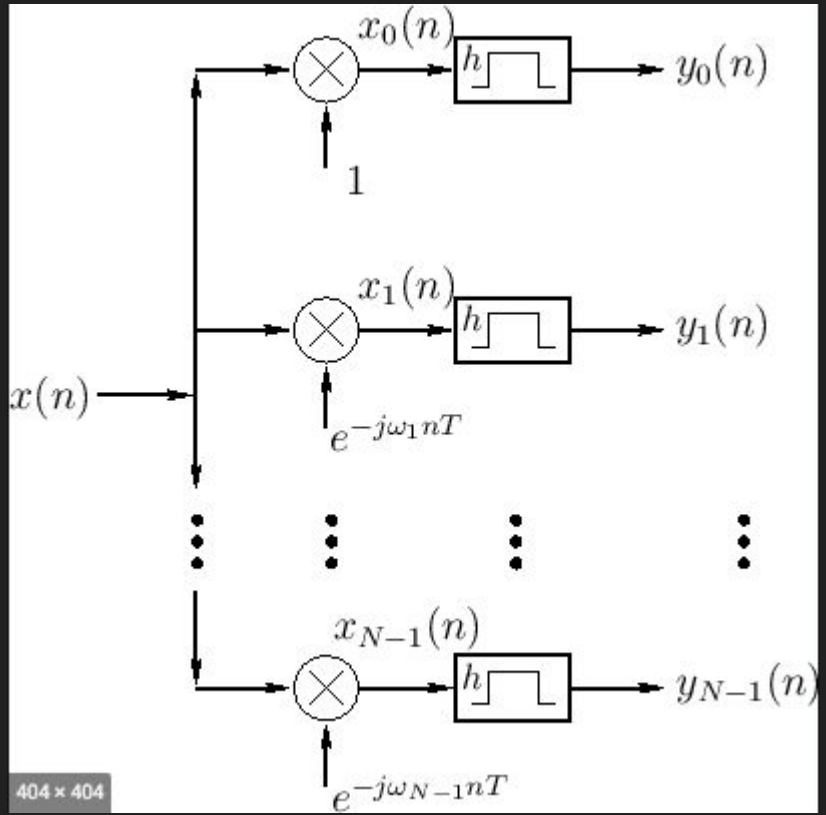
- Next: let’s bring in the analog data from potentiometers
  - aKnob1 chnget "analogIn0"
- Knob data will be 0-1 by default, and will need to be scaled
  - Yes - this is an “strange” range if you’re accustomed to other embedded computing platforms like Arduino that treat this data with a 0-1023 range
  - Don’t use a map function! Too hungry
    - Use this: scaledReading = ((rawReading / rangeHigh) * (scaleTop - scaleBottom)) + scaleBottom
    - Or: scaled = (raw * scaleTop) + scaleBottom
- Third: bring in the digital data from the SPST push buttons
  - iButtPin[1 - 15] = 0
  - kSwitch1 digiInBela iButtPin1
  - 😢 The instantaneous changes from 0-1 and 1-0 as a result of button pushes will cause “clicks”
    - aSwitch tone a(kSwitch), 15
      - Simple IIR Lowpass filter will “slew”/average the data changes

Pseudo-Code Explanation and Considerations (Fixed Waveform)

- Finally: use the data. Ugly but conceptually clear:
  - asig oscil k(aTrk)*k(aSwitch), kpitch*kScaler, 2
  - asig2 oscil k(aTrk)*k(aSwitch2), kpitch*1.066*kScaler, 2
  - asig3 oscil k(aTrk)*k(aSwitch3), kpitch*1.125*kScaler, 2
  - asig4 oscil k(aTrk)*k(aSwitch4), kpitch*1.2*kScaler, 2
  - asig5 oscil k(aTrk)*k(aSwitch5), kpitch*1.25*kScaler, 2
  - asig6 oscil k(aTrk)*k(aSwitch6), kpitch*1.333*kScaler, 2
  - asig7 oscil k(aTrk)*k(aSwitch7), kpitch*1.406*kScaler, 2
  - asig8 oscil k(aTrk)*k(aSwitch8), kpitch*1.5*kScaler, 2
  - asig9 oscil k(aTrk)*k(aSwitch9), kpitch*1.6*kScaler, 2
  - asig10 oscil k(aTrk)*k(aSwitch10), kpitch*1.666*kScaler, 2
  - asig11 oscil k(aTrk)*k(aSwitch11), kpitch*1.8*kScaler, 2
  - asig12 oscil k(aTrk)*k(aSwitch12), kpitch*1.875*kScaler, 2

Pseudo-Code Explanation and Considerations (Filter Bank Decomposition)

- Finally: use the data. Ugly but conceptually clear:
  - aOne butterbp aNoise, kpitch*kScaler, 1
  - aOne = aOne * aSwitch
  - aTwo butterbp aNoise, kpitch*1.066*kScaler, 1
  - aTwo = aTwo * aSwitch2
  - aThree butterbp aNoise, kpitch*1.125*kScaler, 1
  - aThree = aThree * aSwitch3
  - aFour butterbp aNoise, kpitch*1.2*kScaler, 1
  - aFour = aFour * aSwitch4
  - aFive butterbp aNoise, kpitch*1.25*kScaler, 1
  - aFive = aFive * aSwitch5
  - aSix butterbp aNoise, kpitch*1.333*kScaler, 1
  - aSix = aSix * aSwitch6
  - aSeven butterbp aNoise, kpitch*1.406*kScaler, 1
  - aSeven = aSeven * aSwitch7
  - aEight butterbp aNoise, kpitch*1.5*kScaler, 1
  - aEight = aEight * aSwitch8
  - aNine butterbp aNoise, kpitch*1.6*kScaler, 1
  - aNine = aNine * aSwitch9
  - aTen butterbp aNoise, kpitch*1.666*kScaler, 1
  - aTen = aTen * aSwitch10
  - aElev butterbp aNoise, kpitch*1.8*kScaler, 1
  - aElev = aElev * aSwitch11
  - aTwel butterbp aNoise, kpitch*1.875*kScaler, 1
  - aTwel = aTwel * aSwitch12



On Design Philosophy

- Idea -> Proof of Concept -> Prototype [-> revisions -> Prototype… etc] -> “Commercial” Product
- For this project: prototype phase
  - Usable, functional, sounds good (i.e. as intended)
  - Could be prettier in terms of presentation
- Building the instrument and developing the code v.s. packaging it in a “commercially viable” format are two very different things
- “Self-facing” technologies v.s. “Market facing”
  - This is *very* self-facing
- Microprocessors are readily available to be used in another project, or to be “subbed back in” if removed
  - I tend to do a lot of this

Why?

- I find it unfortunate that many wind instruments *cannot* play polyphonically. It's certainly not physically impossible: organs work polyphonically by air…
  - Why can't clarinets or saxophones or flutes?
- Really enjoy using code for music, but sometimes do not like the lack of physical expression involved.
  - Breath is a *very* physical way to play an instrument
  - Weirdly I actually don't play any wind instruments so, learning the breathing for this has been a humorous challenge
- For potential use in improvised music
- Are there not similar pieces of technology out there?
  - Yes (EWIs, EVIs)
  - Existing technology is very referential to the fingering standards of other instruments
  - Not very customizable (MIDI = nice, but also limiting)
- Building instruments is fun and educational!

Future Redesign Plans

- Update enclosure to wood with closed sides
  - And in general: upgrade the appearance
- Develop custom PCB for iterability
- Replace buttons with a continuous touch sensor?
  - No more discrete notes
- Swap location of knobs and octave switchers
- Develop with lower level language?
  - Right now the Bela uses Csound and the Teensy uses C
  - Building with 1 microprocessor is preferable
- Upgrade with higher quality buttons
  - Or swap for FSRs?
- Lights? LEDS?