HU3010 Final Report

May 13, 2020
Russian Dragon D2020 Group
Benjamin Klaiman, Grant Ferguson, Ryan Astor, Victor Mercola

Introduction

Over the past term, we have made several critical improvements to the Russian Dragon circuit started by the A19 team. This included implementing a full-wave voltage rectifier to make processing the input audio easier, a peak detector to output a high voltage signal when a note was played, and comparing a low-pass T-filter and a voltage divider to better determine when the difference between notes occurs.

In addition to the Analog Front End, our team created a new component of this project: a software simulation of the Russian Dragon device through Python. We go much more into depth of how it works in our ReadMe but the quick rundown is that it takes in a .wav audio file as input and also records what the user wants to compare to and what they think the tempo is. This then brings up a graphical interface of how the program reads the file for tempo at each beat and estimates by how much it is rushing or dragging.

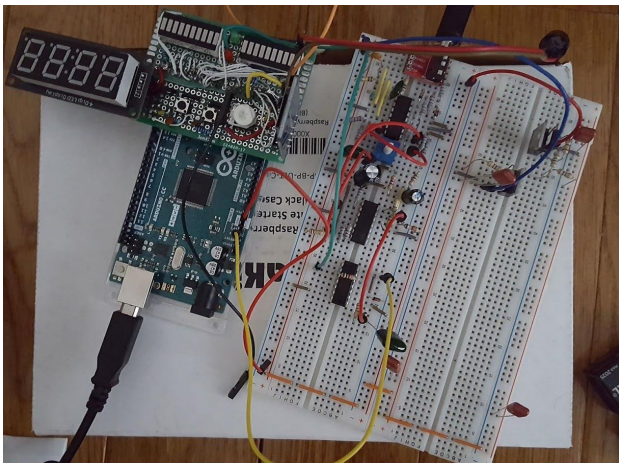

Fig. 1. D20 Russian Dragon breadboard

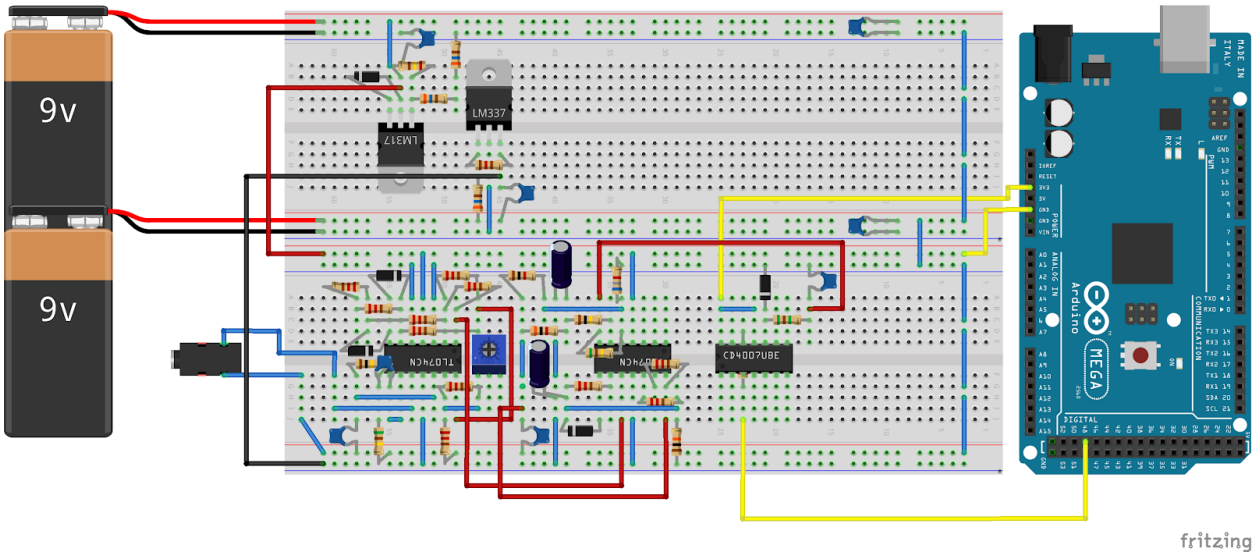

Fig. 2. D20 Russian Dragon breadboard diagram, drawn in Fritzing

Hardware Component

The hardware component of the Russian Dragon takes an audio jack input and outputs a 3.3V-low pulse when a note is played. In our implementation, an input audio signal first goes through a voltage buffer and a bandpass filter so that notes outside a subset of the MIDI note frequencies are attenuated. This signal then goes through a non-inverting amplifier controlled by a potentiometer: this acts both as a “gain” and “sensitivity” knob for the circuit. This signal then is sent through a full-wave rectifier, which flips negative parts of the signal so that the signal is now between 0V and 5V. This rectified signal is then sent through a peak detector, which determines the maximum output of the now-rectified signal over a (short) period of time. From here, the signal splits: the low-pass filter smooths out the peak detector’s input and approximates its middle to determine when a note should stop being played, and the voltage divider downscales the voltage so that detecting a noted impact is easier. The comparator outputs a 5V high pulse whenever a note is played, and -5V when a note is not played. The 5.1MΩ resistor and the diode here bring the negative voltage up to about 0 V, and the CD4007UBE chip acts as a CMOS inverter to output a voltage-low when a note is played.

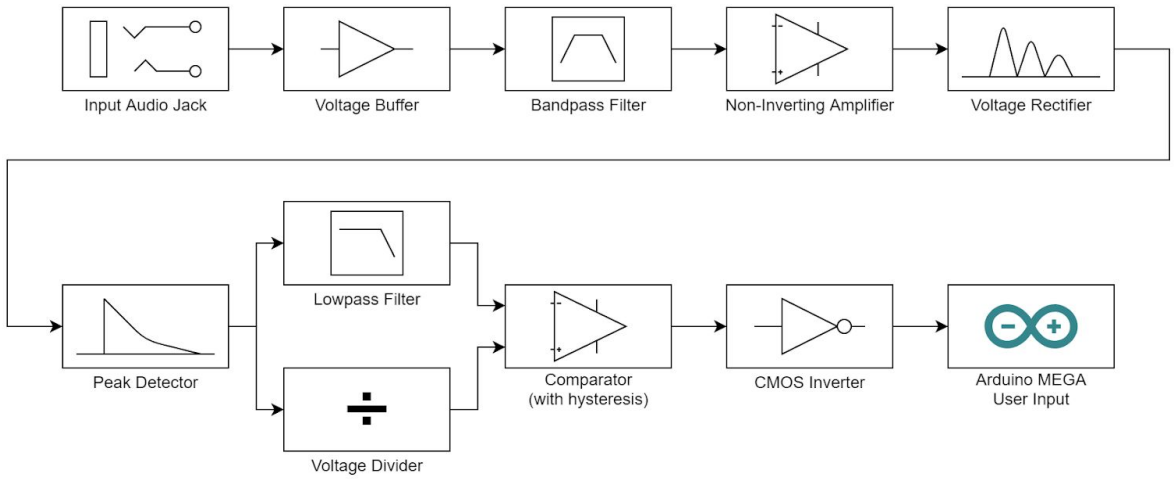

Fig. 3. D20 Russian Dragon block diagram

Process

There were three main steps to prototyping the hardware component: simulation, computer audio, then instrument audio. At the beginning of the project, LTSpice and Falstad Circuit Simulator (a.k.a. circuitjs) were used to simulate the circuit's outputs given an input file - this allowed for much safer rapid prototyping instead of hot-swapping resistors or constantly plugging and unplugging voltage sources. After the circuit was successfully built according to the simulation, the 3.5mm audio jack was plugged into a computer and audio files were tested with the circuit. An audio jack splitter connected to a speaker is recommended for listening to the music while watching the data get processed by the circuit in real-time. After the circuit was built, the circuit could then be plugged into an instrument. In our case, we chose the Yamaha P115 electronic 88-key piano. The AUX mono output of the piano was connected to a hum destroyer, which was then fed to the circuit, as suggested by project sponsor Ryan McKenna.

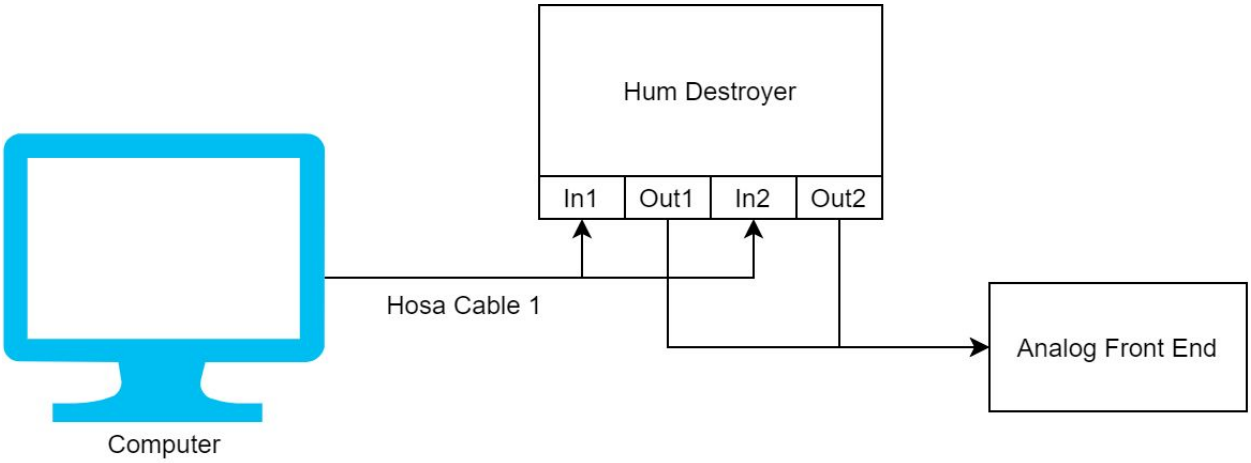

Fig. 4. D20 Russian Dragon hum destroyer-to-circuit connection

Issues

Unfortunately, our implementation of the Russian Dragon isn't perfect and still needs fixing. For starters, the bandpass filter attenuates hi-hat frequencies, which may be problematic to some musicians. Using two D-batteries for a positive and negative rail is unwieldy. To fix this, we suggest putting an active full-wave voltage rectifier first (see the unused concepts page in our schematic) and adjust all other circuit components to use the ground instead of the -5V source. We didn't have time to implement it because of simulation problems and time constraints. Another problem is that we didn't include a full-blown sample-and-hold with a decay subcircuit for the frontend. The lowpass T-filter and voltage divider have a similar function, but do not operate on the same level as a MOSFET-powered sample-and-hold subcircuit. Our current version has trouble determining guitar slides since the action of sliding does not create a noted impact. On the ease-of-use side, an ON/OFF switch and an LED (as suggested by sponsor Robert Palmer) would be easy additions to this project.

Software Component

For the software component, we started developing the system about 3-4 weeks into the project, as we discovered that we needed a way to simulate output of the device. We first started by making a basic Python script that took in a predetermined file, and outputting a graph of the beats of the song and the tempo. For this step, we used a library called Librosa, that gives information on sound files, and is what we used to calculate the tempo and beats of the music inputted. The next phase was the development of a UI system, for which we used Python's built-in UI framework, Tkinter. For the rushing and dragging icons, we used Google's Material Design icon library in order to make the icons as simple and easy to understand as possible.

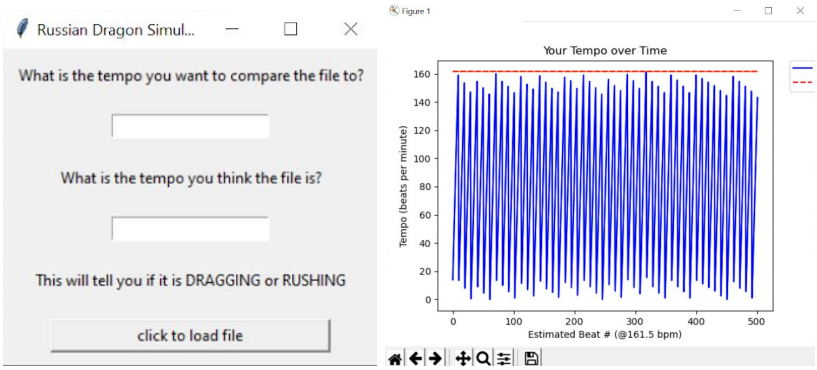

Fig. 5. & 6. D20 Russian Dragon Simulator stating screen and example graph

Resources Used

For physical resources, the Russian Dragon circuit and accompanying Arduino Mega are most of the resources you will need to get started. However, more materials will be required for making various improvements such as capacitors, resistors, op-amps, and more. All accompanying code for the circuit is on the GitLab repository. When you have access to that, cloning it to your computer should be fairly straightforward and then run the program with the circuit attached to your computer.

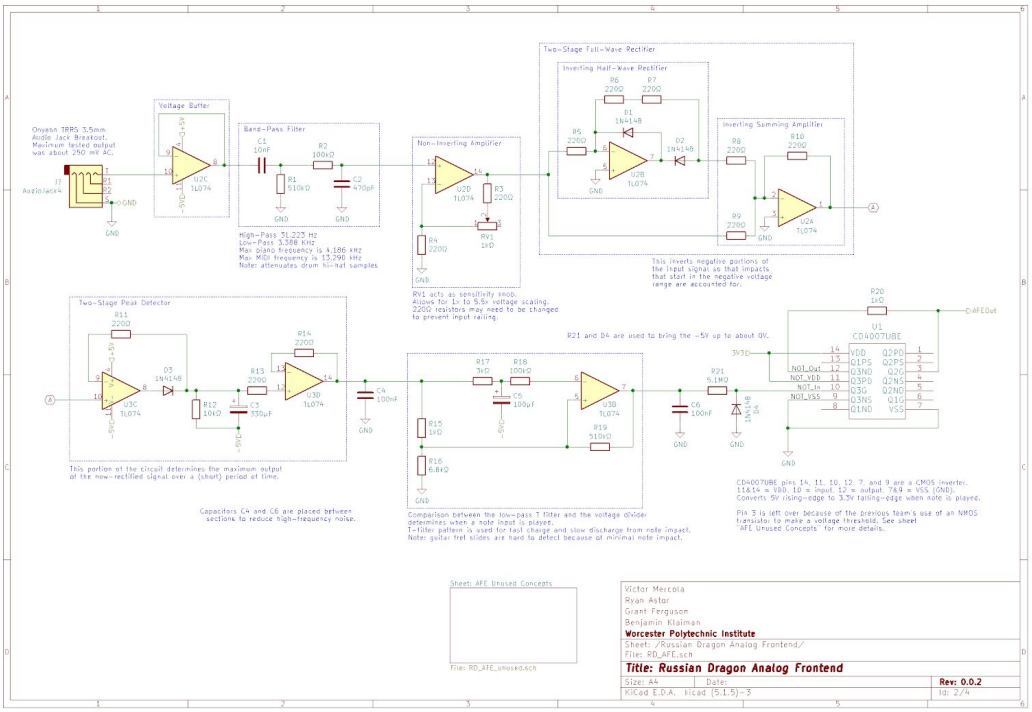

Fig. 7. D20 Russian Dragon circuit documentation (Available in GitLab at higher res)

The software used to model & simulate were KiCAD, Falstad Circuit Simulator (circuitjs), LTSpice, and Fritzing. Despite being constantly updated with new features, LTSpice feels like it was made by someone who has hyper-nostalgia for the late 1990s UI and a deep hatred of conventional shortcuts. Fritzing, on the other hand, is stuck in development limbo and is in dire need of donations.

For the Russian Dragon Simulator, it is also found on the GitLab repository so you can clone it. Assuming you have Python installed on your computer, it should run fine. We used and would recommend PyCharm to run and make any edits to said program.

All information that is necessary to understand the project is on the GitLab circuit diagram or other documentation.

Improvements

Some improvements that can be made to the hardware and software are as follows:
Hardware (AFE) & Arduino improvements:

- Fix band-pass filter values
- Remove -5V source by rearranging individual subcircuit blocks
- Determine note changes from guitar fret slides
- Add a dedicated ON/OFF switch for safety and ease-of-use
- Add an LED to show when the Russian Dragon is on (suggested by sponsor Robert Palmer)
- Double-check note input Arduino code (if needed)
- Improve Arduino Shield documentation (if needed)

Simulation Improvements:

- Multi-File Input
- Improved graphing and filtering
    - Adding the graph to the program window instead of displaying in PyCharm, invisible to the end user
- Adding more icons to spruce up the UI, the decision to add icons came fairly close to the deadline and we were unable to add more in time