

ECE 4220 – Guitar Looper Station

Abstract

A guitar looper is a device which allows a musician to record a loop of themselves playing. The looper also passes through sound the entire time so that they can hear what their playing as they record a loop or on top of a loop. Such a device is constructed using a BeagleBone Black mini-computer. The device is controlled using buttons connected to the board's GPIO pins. Audio input and output is facilitated using the JACK API and a USB sound card.

Introduction

The goal of this project for me was to figure out how to get real-time audio recording and playback from my guitar, on my BeagleBone black mini computer. I also wanted some sort of foot-switch mechanism, so at least button controls for proof of concept. I plan to use this as a base for a more complicated DSP guitar effects station. The looper has two buttons, one for recording and one for playback. The musician need only press the recording button once to start recording and again to stop. Immediately the loop starts playing until the playback button is pressed. The playback button pauses a loop until it is pressed again. If no loop is present, no loop is played if the playback button pressed. At the moment, when a new loop is recorded, the old one is erased. Layering would have been implemented but I ran out of time because I had to deal with hardware issues.

Background

The idea of a looper station is not new by any means. Live looping, as it's formally called, has been an integral part of modern music for nearly half a century. Originally these devices were entirely analog and worked by delaying a signal in hardware. Digital signal processing allows for a wide range of programmable effects. The hardware which facilitates basic DSP is now cheap and readily available. Professional live loopers are embedded systems. The hardware is usually designed from the ground up to exactly support the platform. While this makes it cheaper to produce and guaranteed to work to specifications, it incurs a large overhead. As such, looper stations retail for no less than \$100. The BeagleBone device is only \$45 and the USB sound card and other electronics only add up to about \$10 or so. This makes for a much cheaper and potentially versatile looper.

Implementation

The BeagleBone Black is an ARM based mini-PC. It features a 1 GHz processor and runs modified Linux distributions right out of the box. The device also features a USB port. I used a cheap USB sound card which I knew was supported by GNU Linux. The sound card would allow for high quality, low-latency DAC and ADC for input and output of the guitar signal.

A preamp circuit was also needed. A guitar has a very high output impedance (~50M-ohms). A sound card's microphone input has a relatively low input

impedance (600-6.5k-ohms). This means that if the guitar was connected directly to the microphone input then the voltage across the microphone input would be very near ground with respect to the voltage across the guitar's pickups. An amplifier usually has a very high input impedance for measuring voltages accurately, unlike the microphone input. To mimic this and allow for accurate recording, an impedance matching buffer circuit is necessary. The proper circuit would be the one shown in fig. 1. I built a very simple impedance matching circuit with only two AC-blocking ceramic capacitors and an LM471, shown in fig. 2. This circuit is not properly designed and ended up damaging the sound card. It is not current limiting and the floating ground between the batteries adds a lot of noise.

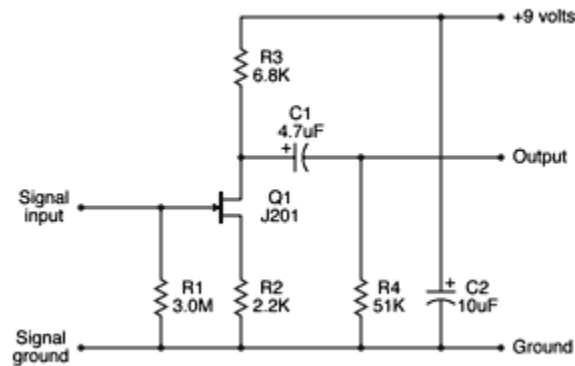


Fig 1. - Improved Discrete JFET Buffer

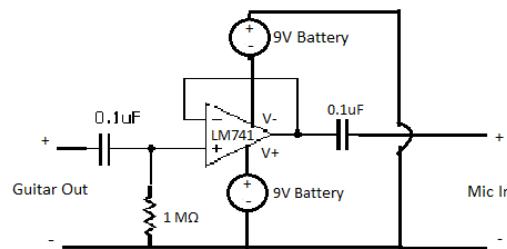


Fig 2. - Original LM41-based Buffer

57 To connect the device to the guitar I spliced a ¼ inch mono cable and use
58 alligator hooks to connect it to the preamp. The preamp's output was connected to
59 a spliced 1/8 (3.5mm) cable. This cable was connected to the sound card's
60 microphone input. The headphone output of the sound card was connected to a 1/8
61 to ¼ adapter. The adapter is connected directly into the guitar amplifier. Two push
62 buttons were also connected to two GPIO pins on the BeagleBone Black. Fig 3.
63 Shows the overall system while it's running. Fig. 4. Shows the GPIO button circuit.
64 Fig 5. Shows the preamp circuit.

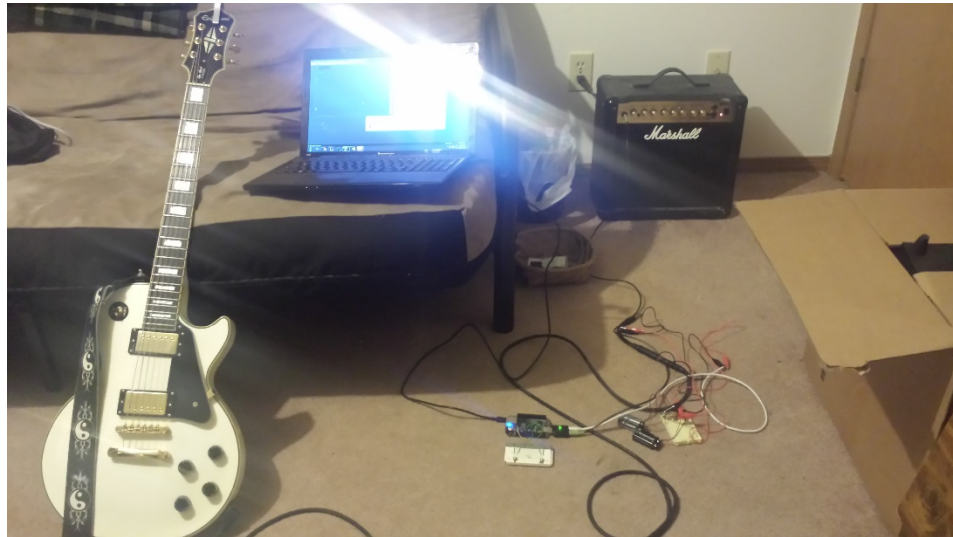


Fig. 3 - Overall System

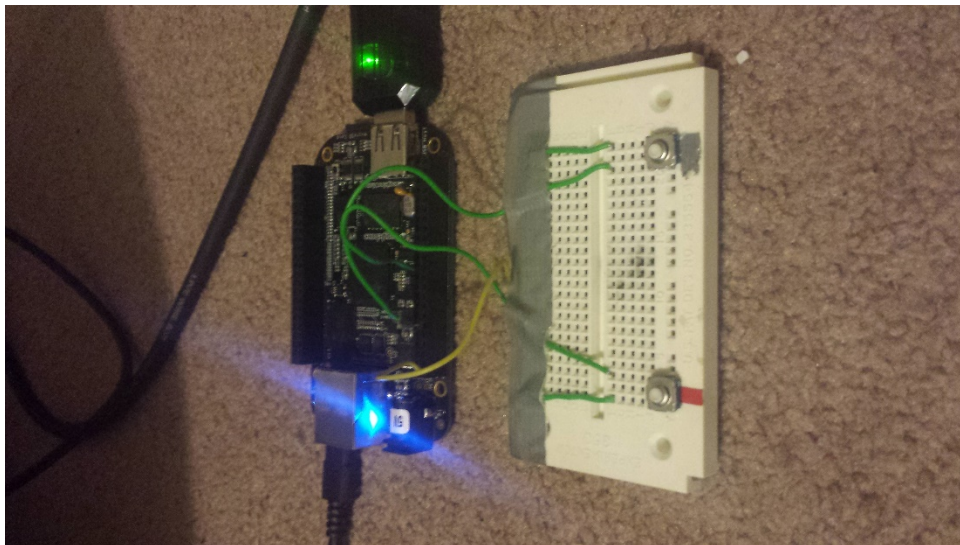


Fig 4. - GPIO buttons and BeagleBone Black

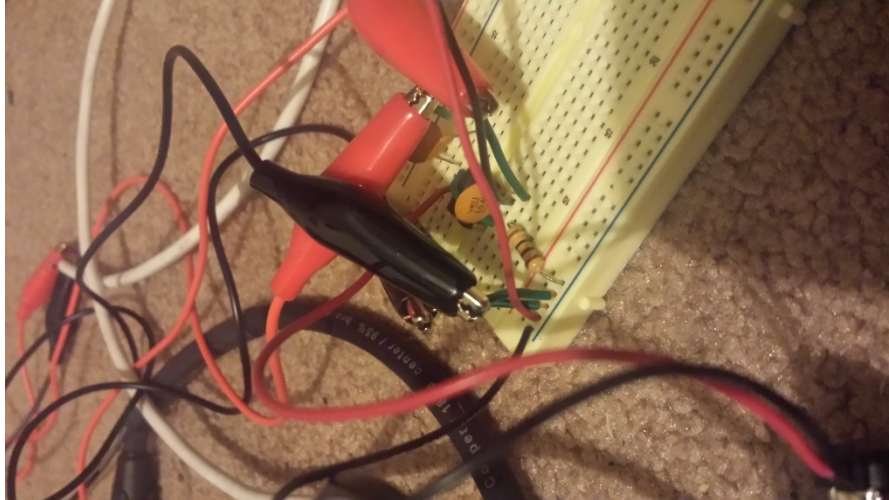


Fig. 5 - Preamp Circuit

So that's the hardware. The software was written in C++. The push buttons were read using the simpleGPIO library which is provided by Derek Molloy. This library offers simple functions like `gpio_get_value()` and `gpio_set_value()` which allows for polling of the buttons. Accessing the sound card's internal buffers in real-time was facilitated using the JACK audio API. The API sets up a real-time callback system for DSP. Basically, JACK runs in the background reading and writing to the sound card's internal buffers for/from ADC/DAC. JACK stores the data in its own buffers. Every time there is a certain amount of data ready in these buffers, a user defined callback routine is called. The routine can modify the buffers, sending data from the input to the output for audio pass through, as an example.

My program first initiates JACK and the GPIO ports then sits in a loop, polling the buttons. If the record button is pressed then a flag is triggered. Inside my callback process I check this flag. If it is asserted then data is recorded from the input into a loop buffer. When the recording button is pressed again, the flag is lowered and the index of the final position in the loop is saved. When the playback button is pressed, if there is a loop recorded, then the playback flag is asserted. This flag is also checked in the callback process. If it is set then the input signal is mixed with the loop that was saved. A pointer keeps track of the position in the loop. When the end is reached, the pointer is reset making it a circular buffer. The signal is mixed by simply adding the two floating point numbers which represent the voltages of the analog signal that is the input and loop. Adding has the potential of clipping, however this was not an issue. Adding is also truest to the physical addition of two sound waves. If the playback flag is not set the input is simply copied to the output for pass through.

Experiments and Results

The experiment for this project is simply to see if it works as intended. I set the maximum loop length at 30 seconds in my calculations for memory so this limit had to be tested. Also, pressing the buttons repeatedly and randomly had to be tested to make sure it didn't effect performance. Another thing to test was all of the different states of the machine. There are three main flags that control the state of

the device. The LOOP_AVAILABLE flag is set after a loop has been recorded. The PLAYBACK flag is set when the device is playing back a loop. The RECORDING flag is set when a loop is being recorded. Since there are 3 bits there are 8 possible states. Each state was tested to make sure that the device was operating as intended. I flushed out a few bugs with this, which were corrected. Now, the device works completely as intended, besides a hardware bug described below.

While conducting testing I began to run into an issue. After a few minutes of testing, the sound got very crackly and distorted. Sometimes, the problem would only last a few seconds, sometimes it never went away until everything was taken apart and rebooted. I tried to make sure there wasn't a bug in my program which would cause some sort of modulation in the signal. I eventually tested the sound card independently. I used audacity to record and playback from the sound card when it was plugged into my laptop. I found that the recording would get distorted. From this I concluded that it was the sound card and not my program. To verify my conclusion, I modified my looper to use my laptops sound card and keys as button inputs. This had no issue whatsoever but wasn't portable and wasn't controlled by push buttons. I think the problem lied in my preamp circuit. I neglected to limit the current coming out of the op amp. The alternate, improved circuit is shown above in fig. 1.

Discussion and Conclusion

This project served as a good proof of concept. I was able to construct a fully functional live looping station that was portable and could be controlled by a footswitch. I learned that you should be very careful in your circuit design instead of just trying to make it work. My simple op-amp circuit overloaded my cheap hardware which lead to a failure. I would have liked to have redesigned the buffer circuit earlier in time to order the parts and try it out. I would have also liked to have bought another sound card to try out the new buffer circuit on the BeagleBone. Another feature I would have liked to implement is layered looping and/or effects processing. These would have been trivial if the hardware issue hadn't delayed everything.

Appendix

LINKS

Derek Molloy - [GPIO Library for BeagleBone Black](#)

JACK API - [Playback / Recording Real-time Interface](#)

JFET Design - [Improved JFET based preamp design](#)

CODE