Project Cyren: Music Effects Device

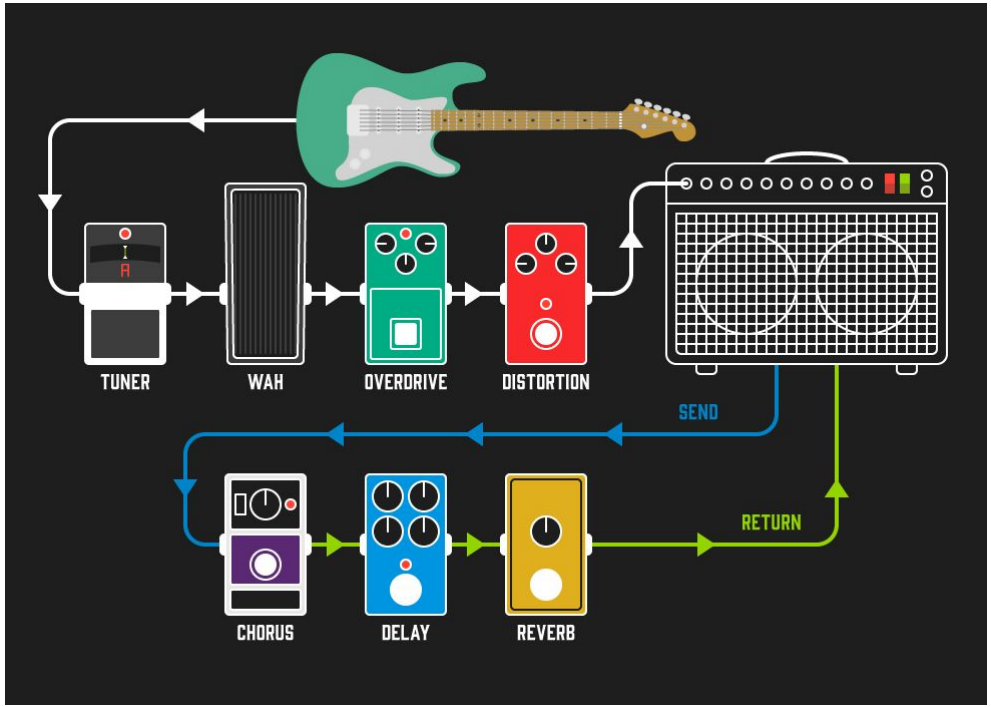
Group 15: Justin Shaver, Thomas Frye, Will Pigg, Chandler Davis, Daniel Bohlke, Caleb Hendrickson

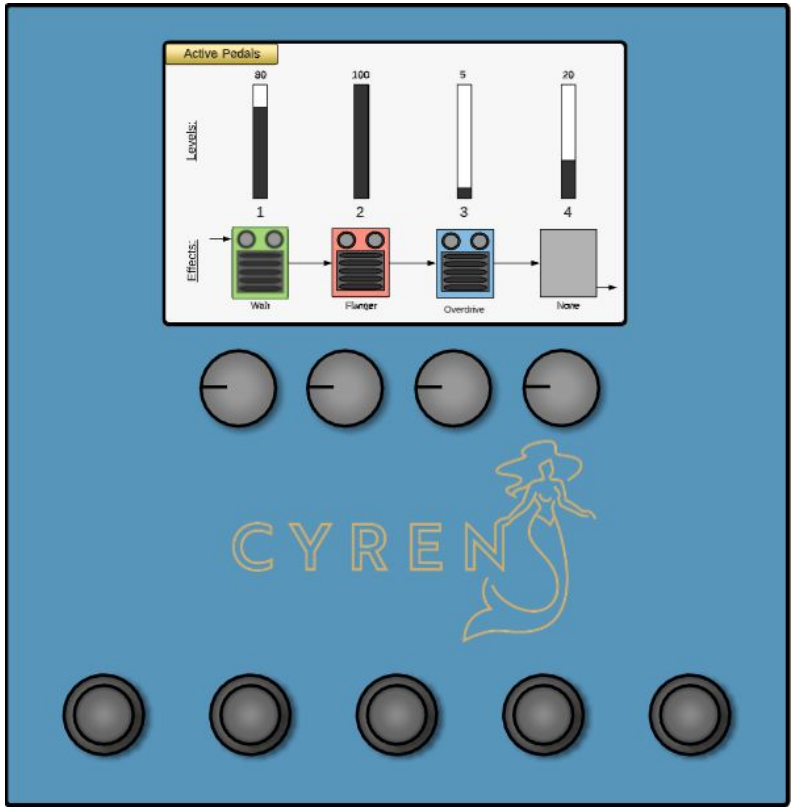
Problem Statement

- Musicians often require many devices in order to achieve desired sounds
  - Combination of several pedals in a particular order
  - Ensure compatibility between devices
- Using many different effects for the first time may be confusing to beginner level musicians
  - User might not have previous knowledge on a wide variety of effects
  - Many different configurations and adjustments for just one effect
- Storage of equipment may pose a problem under tight constraints
  - Ensure equipment remains safe and undamaged during travel
  - Durability Concerns

Cyren Overview

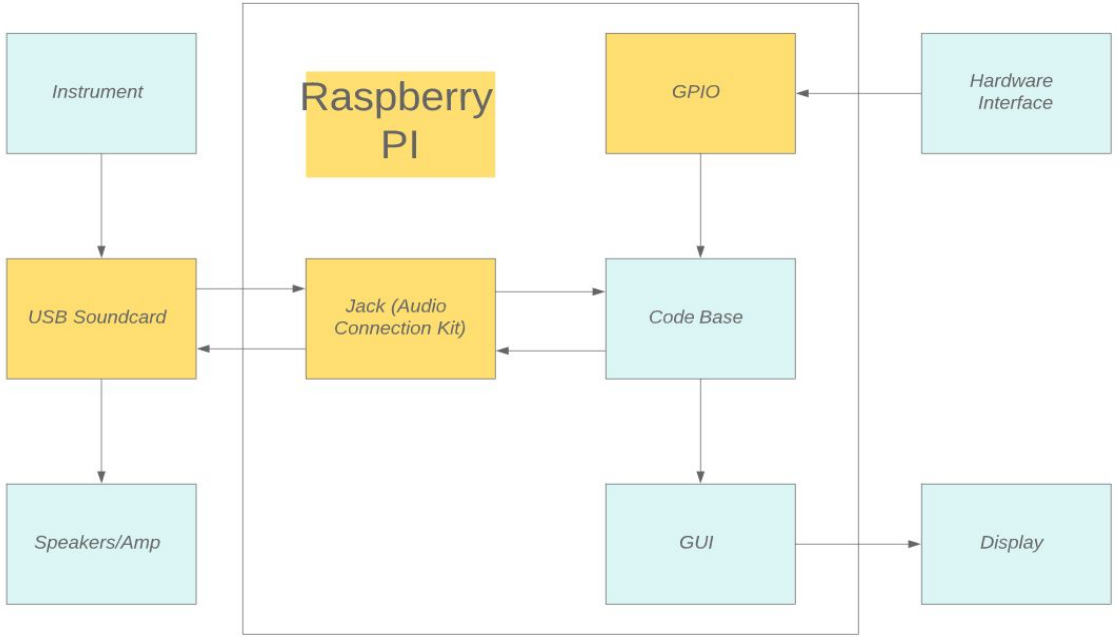
- Condense the musicians required hardware into one device
- Maintain form factor and user interface of existing musical effect devices
- Allows for 2 instruments inputs and exports audio to a multi-channel output or single channel
- Features 4 software emulated effect “pedals” (stomp switches) that apply filters to the incoming audio
    - Allows for application of filters based on user selection
    - Handles the use of multiple filters at once
    - Able to adjust certain variables used for the different filters
    - Visual representation of filters being applied and their adjustable settings
    - Toggle effect(s) on/off


TUNER
WAH
OVERDRIVE
DISTORTION
SEND
RETURN
CHORUS
DELAY
REVERB


Active Pedals
Levels:
80
100
5
20
1
2
3
4
Effects:
Wah
Flanger
Overdrive
None
CYREN

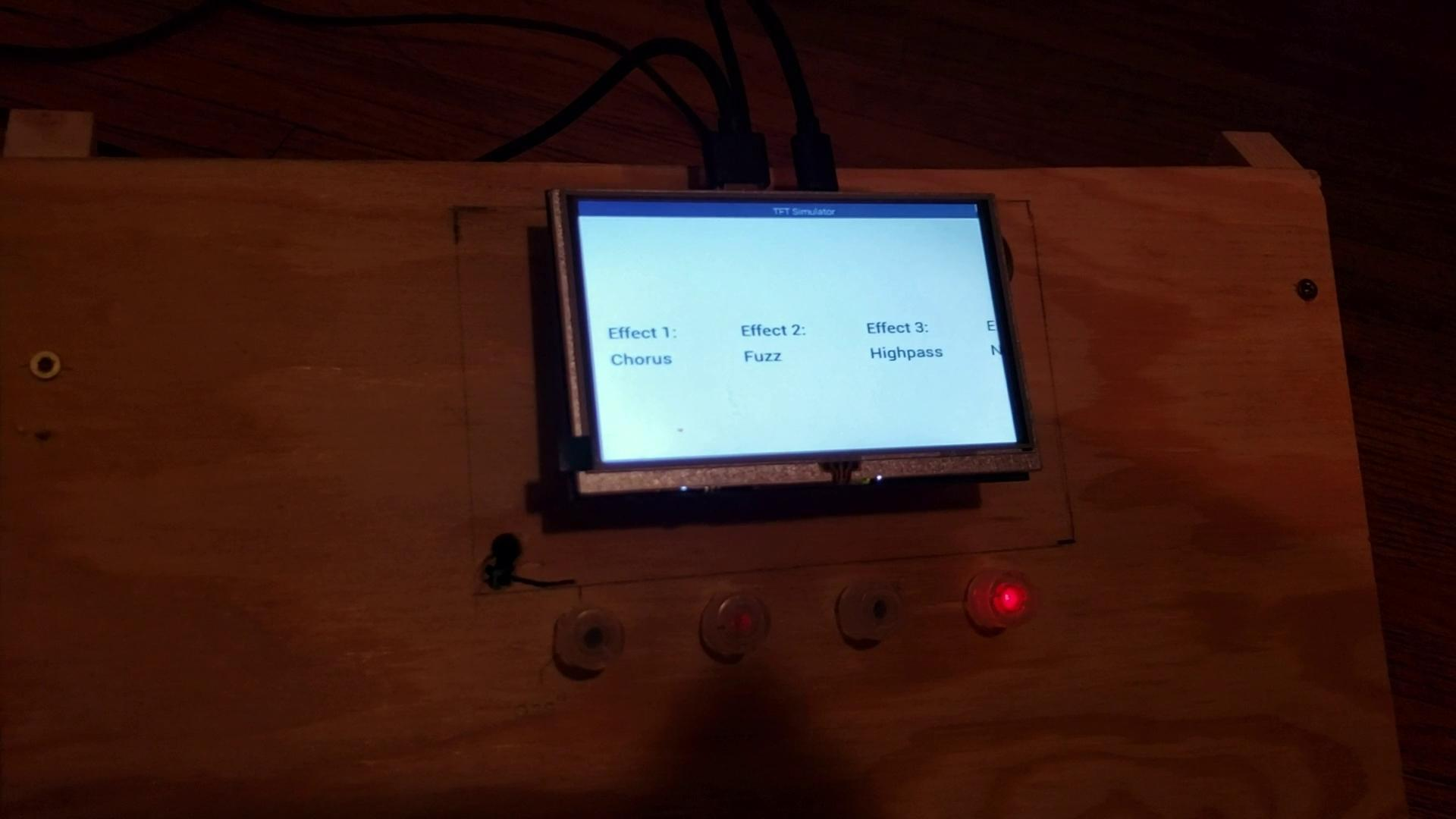
High Level Block Diagram

Yellow indicates components changed/added from a design pivot (more on this later)



Light Explanation of How Project works

- The device is powered by a 5V standard wall plug.
- Our design takes input from a guitar through ¼ inch jack in a usb interface, and transmits this via usb to the Pi
- Our code establishes an input port, an output port, and determines the capture ports and broadcast ports and routes these ports appropriately. Once these are established, Jack listens to the data coming in from the capture port and copies it to the input port.
- Once the data has been copied to the input port, the data is available for alteration. The incoming sound is transformed in real time, and the characteristics of the transformation are determined by the buttons/parameters that are selected/adjusted on the hardware/gui.
- The transformed data is copied to the output port and routed back to the usb interface to the broadcast ports(ie. Speakers, amp, etc.)


TFT Simulator
Effect 1:
Chorus
Effect 2:
Fuzz
Effect 3:
Highpass

Implementation/Testing

Client Application (Implementation)

Main Application (gui_client)

- Upon execution, spins up main thread
- GUI, Effect Application, and Audio Routing is all tracked from this thread

Jack Audio

- Allows for configuration of sample rate and frames per period
- Detects audio inputs and outputs of device
- Configures routing between audio inputs and outputs
- During routing, we direct the audio through our various filters based off selections

Client Application (Testing)

Setup

- Wired Pi into audio interface via usb, plugged in guitar into audio interface, and setup speakers out of the audio interface

Testing Jack Configurations

- When configuring routes and settings, we would use “qjackctl”, a popular Jack Audio Client, to confirm our results
- The hardest part of testing the client was verifying that the filters were being applied correctly
- We often would compare to other devices’ versions of said filters to ours by ear

Electrical/Hardware (Implementation)

- Hardware components connected and powered through GPIO bus on the Raspberry Pi
  - Rotary Encoders - 3.3V soldered board (qty. 4)
    - Pushbutton, bit A and bit B, and Red/Green LED selection require 5 pins on GPIO
  - Stomp switches - 5V pull down circuit (qty. 5)
    - Each stomp takes a single GPIO pin
  - See appendix for circuit diagrams
- LCD screen connected through HDMI and has separate power of 5V/2A
- USB sound card connected and powered through USB port on Raspberry Pi

Electrical/Hardware (Testing)

- Each pin connection was studied to check proper voltage level changes (e.g. stomp switches rising from 0 → 5V)
  - These test points consisted of the following:
    - Bits A and B on the rotary encoder
    - Button Press pin on encoder
    - Stomp switch output
- Initially, guitar signals were studied under a lab oscilloscope
  - Signals were between 0 and 2.5V depending on the strum
  - Determined that a sound card would be implemented to guarantee a clean audio signal
  - ADC and filtering were taken care of
- Rigorous testing was done with GPIO interaction and code base

GUI (Implementation)

Library

- The Graphical User Interface was created using LittlevGL, a C library for creating GUIs.
- This library's structure was such that all of the necessary code base was contained within an lv_conf.h file and was well documented at: https://docs.littlevgl.com/en/html/index.html

Design

- Our design incorporates the percentage of each filter along with the ability to change the effect in the same screen.

GUI (Testing)

PC Testing

- We began testing by setting up a simulator for the LCD Screen.
- This simulator was created using XServer and allowed us to make changes quickly to the GUI code so that when it came time for LCD testing it would be ready.

LCD Testing

- Once the GUI was ready, the code was downloaded onto our Raspberry Pi.
- The Raspberry Pi was then connected to our 5” LCD Screen and the code was run to make sure it fit the screen well.

Digital Effects (Implementation)

The audio effects and theory used in this project were gathered from the following sources:

- *Audio Effects: Theory, Implementation, and Application* by Andrew McPherson and Joshua Reiss
- *Digital Audio Effects* by Orchisama Das, a document from the Stanford Center for Computer Research in Music and Acoustics

Effects Implemented:

Distortion Effects, Time Delay Effects, Filter Effects, and a Modulation effect.

Digital Effects (Testing)

Two stages of functional testing were done on the effects:

- The first stage was done manually in Visual Studio
  - Data from a wav file was sampled, transformed, and output to a new wav file
- The second stage of testing was done manually on the Raspberry Pi with the Jack library set up
  - Data was sampled from looping guitar audio samples, transformed, and output to speakers
  - Several effects were considered non-functional temporarily after this stage

Other Considerations

Crucial Design Decisions

Switching from the RockPro to the Raspberry Pi, Scarlett USB Interface, and JACK routing utility

- We learned that we were going to have a hard time interacting and programming the RockPro's onboard sound card.

Switching GUI libraries from GTK to LittlevGL

- LittlevGL is designed for embedded software whereas GTK was intended for desktop environments.

Removing the looper functionality from our device

- Added too many requirements to the device such as storage requirements, storage navigation, a compression algorithm, alternative routing, etc.

Challenges & Setbacks

Scheduling

- Our team could not find a consistent meeting time during the week where everyone could participate in the meeting.

Proximity

- Only 3 of our team members live in Ames during the week, and only 2 are around on the weekends.

Design Changes

- We made several design changes late in the semester that caused us to slide backwards in time & resources

Limited hardware

- Because we had a limited number of USB interfaces, time with access to the device was limited and laborious

Lessons Learned

Lack of Hardware

- We only ordered enough parts to make one workbench
- With people living in various parts of Iowa it made it difficult to debug various parts of the project
- We learned we needed more workstations to eliminate the “bottlenecked” rate of development

Planning

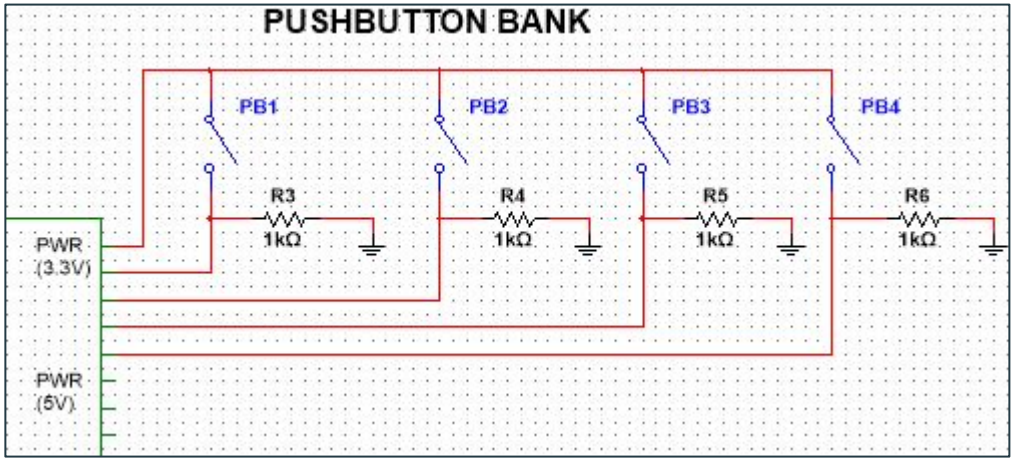
- Second semester, a few members had co-ops or jobs that took them to different parts of Iowa
- Provided difficult to coordinate things like in-person meetings and workshops amongst the team
- We learned to combat this by using our communication channels to constantly remind each other of expectations and events before they arrived

Scheduling

- Struggled to get a weekly meeting scheduled with our advisor/client second semester
- We learned that being more persistent and flexible helps to resolve that issue

Appendix

Stomp Switch Circuit



Encoder Circuit