README for CS4644 project -- Flow

The Github organization is set up into the following repos, with the following purposes

Networking: This repo holds the tools used for communication over network sockets. Uses a basic UDP socket to shove data across a network connection. To use simply include socket_handle.hpp and use send_message(...) to send data and recv_data() to receive a message; note, recv_data will cause the calling thread to halt execution until data becomes available. It should be noted that the test files are now depreciated as our naming and functionality has changed. In reference to Flow, all the code in this repo is for internal use inside our VST plugin and shouldn't be edited.

Led-test: This repo houses the core functionality for controlling leds. To build the files needed for led control simply issue a 'make' command. Note: this code is designed and developed for 32 bit Arm processors running Arch Linux; it will most likely not work on anything else.
CREDITS: Included in this repo are three submodules: rpi_ws281x, arch-rpi-cross, and musl.

- **rpi_ws281x -** BSD-2-clause (Jeremy Garff/jgarff) - This is the framework that is used to control individual leds. While we made minor changes so it would compile on our hardware.
- **arch-rpi-cross** - No license provided (Tavian Barnes/tavinator) - Used to simplify cross-compiling from an Arch Linux host machine. Not required.
- **musl** - MIT license (The MUSL team/musl-libc.org)- A libc implementation that supports being statically linked. Only required when cross compiling.

Hardware: This repo simply contains all of the models for our 3D printed parts. Not really any code here.
CREDITS: Under /hardware/vendors both files were written by a 3rd party.

- **Parametric_involute_gear -** GNU General Public License(Parametric Involute Bevel and Spur Gears by Greg Frost) -
- **Threads.scad -** GNU General Public License (Dan Kirshner)

Juce-plugin: This repo holds the source files for our Midi VST plugin. Code in this repo is what actually intercepts midi notes from a DAW and will transmit them over a socket. Note that this repo is meant to be used in conjunction with the program Projucer to open inside an IDE and set dependencies correctly; anything outside the juce-plugin/VST_MIDI/Source is autogenerated by Projucer. To open, simply install Projucer and open the VST_MIDI.jucer file. Follow the tutorial in this repos README, that can be found at https://docs.juce.com/master/tutorial_manage_projucer_project.html, to set Juce dependencies correctly. From here simply use Projucer to launch inside a supported IDE and compile there. NOTE: Projucer was developed on macOS and while it has versions for Linux and Windows it does not seem always work on these platforms.
CREDITS:

- **TPCircularBuffer -** Custom license in file(Michael Tyson/michaeltyson) - These files implement a lockless circular buffer. This was needed since we needed zero locking, or waiting, on the audio thread for low latency. Note about this library: it will only build on macOS as it uses Mac kernel APIs.
- **JUCE -** GNU General Public License (Roli) - A 3rd party C++ framework we used to develop the VST plugin. Provides API for capturing and processing midi buffers output by DAW.

Servo-test: Now depreciated test repo. This repo was simply to test moving a servo using an Arduino Uno. This code was never used.

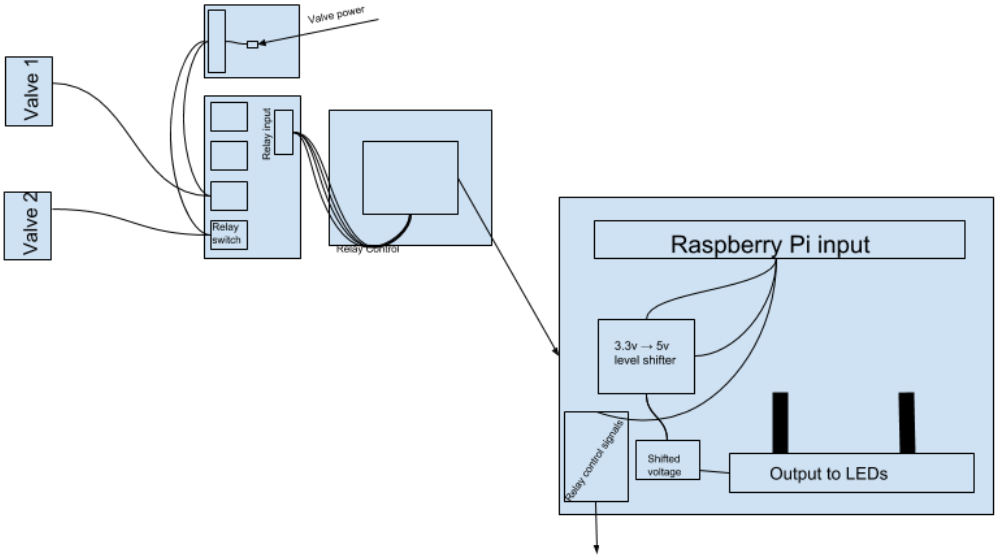
Rpi-ws281x: This is the repo that is included as a submodule in led-test. Again, we wrote none of this code. All credit goes to Jeremy Garff on Github at jgarff. This is simply a fork and should not be edited.

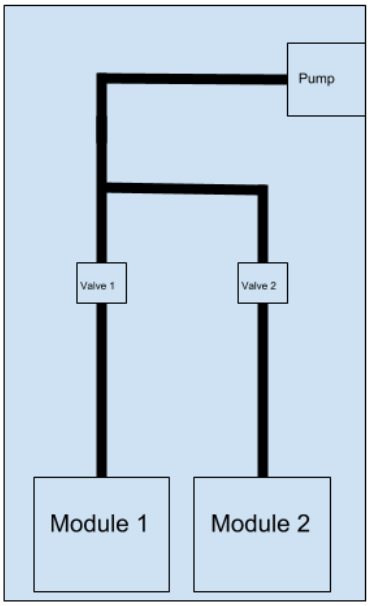
Dsp-blog: This was for the creative practice blog post Reed wrote. This was not used for the final artifact.

Missing elements: Due to the nature of the project, simply having access to the source code is not enough to recreate the artifact. The only missing element is hardware.

- **Raspberry Pi -** The first, and most obvious, missing element is a Raspberry Pi.
- **Raspberry Pi Shield** - We designed and build a shield to be placed on the Raspberry Pi GPIO pins. The shield effectively has two features. The first being it provides 3.3v to 5v level shifting so the 3.3v output on the Raspberry Pi can drive, and light up, the leds which require 5v connections. Secondly, the shield provides reverse polarity protection
- **Relay Switch** - A relay switch is also needed for boolean control over water flow. This is used in conjunction with another board we designed and built. The board takes in a single 24v connection, from a power outlet, and routes the connection to two seperate switches on the relay. Each switch on the relay is also connected to the corresponding water value. This makes it so when each relay is switched on, the loop is completed, trigger a water valve.
- **Water Pump** - The main pump is needed for actual water flow
- **Valves/Modules** - Since Flow is a physical artifact, there needs to be the physical modules as specified in the hardware repo. They need to be routed, with pipes, to individual valves that will connect to a single pump

Below a simple diagram is given to demonstrate how some of the connections work


Valve power
Valve 1
Valve 2
Relay input
Relay switch
Relay Control
Raspberry Pi input
3.3v → 5v
level shifter
Relay control signals
Shifted
voltage
Output to LEDs


Pump
Valve 1
Valve 2
Module 1
Module 2