|  |
| --- |
| Gups1 |
| Guitar Pitch Shifter (GuPS) System Requirements Specification (SRS) |
|  |
|  |
| **Prepared by: GuPS1 Team** |
| **November 7th, 2014** |

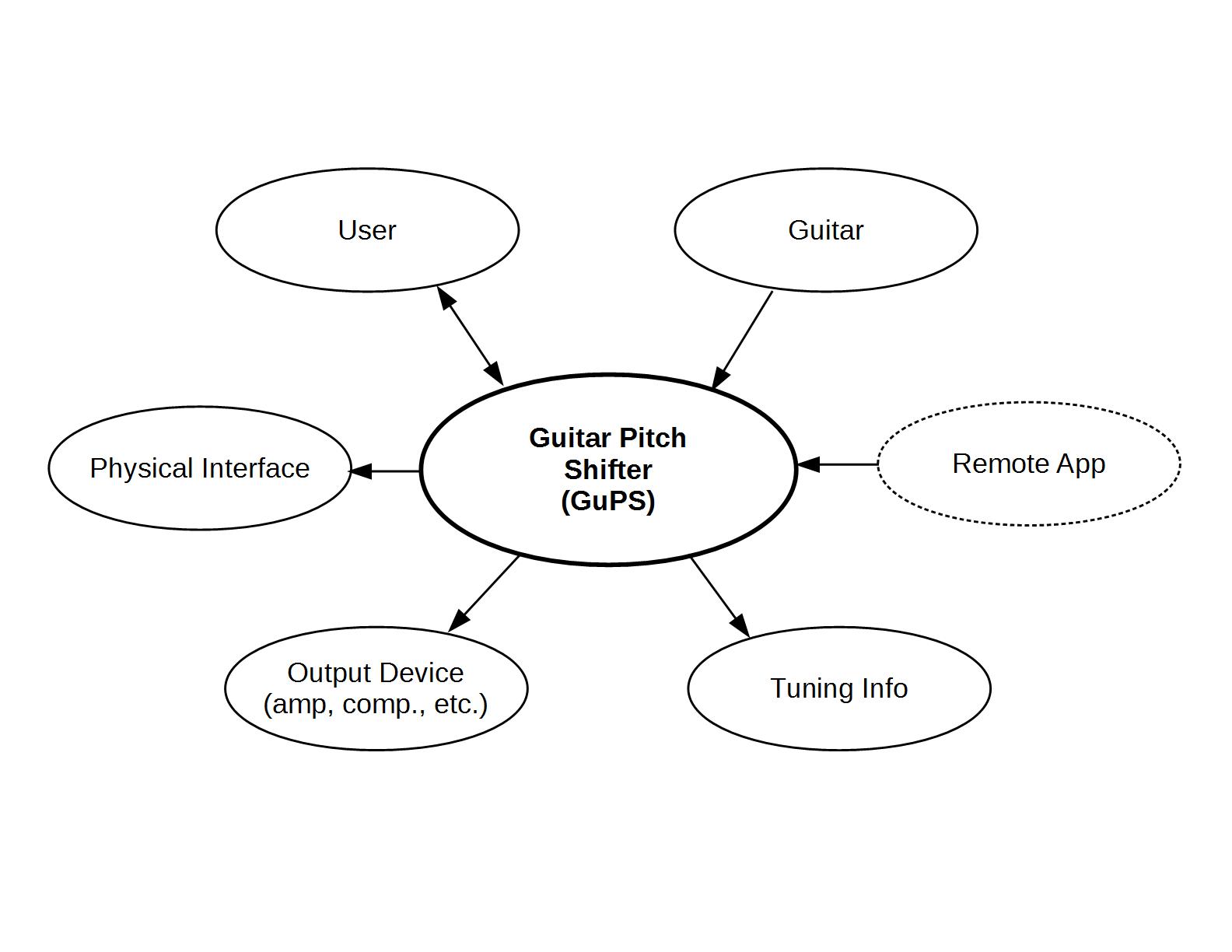
|  |
| --- |
|  |

# bcIntroduction

## Objective and Scope

This document contains the minimum system performance standards for the Guitar Pitch Shifter (GuPS). The performance defined in this document is intended to provide reliable sound quality to the user as well as seamless pitch correction on the order of milliseconds. The device will ensure that the pitch correction will be all-electronic with the user never having to use the tuning pegs of the guitar.

This document anticipates that the user will run the guitar through the pitch shifter and output the shifted signal to an amplifier or other processing effects pedal. The pitch shifter will have 4 distinct modes that the user can select: single string retuning, electronic capo, dynamic tuning, and auto-tuning. These modes will satisfy different functionalities ranging from electronic tuning, to creative pitch shifting effects and an ability to maintain a given tuning. An external application may be developed to provide the user with detailed utility information about the tuning and to allow for tuning selection as well as product calibration. In the event that building a production pedal is not available, a PCB design will be submitted. Figure 1 shows the System Boundary Diagram (SBD) which details the interfaces between the GuPS system as described and surrounding equipment.



**Figure 1: GuPS System Boundary Diagram**

This document anticipates 3 distinct phases to the project. Phase 1 will consist of developing pitch shifting algorithms in MATLAB. Phase 2 will consist of implementing those algorithms on an FPGA with visualization software for the output signals. Phase 3 will consist of a PCB design and the final construction of the pitch shifter.

## Definition of Terms

This section provides explicit definitions for terms used in this document pertaining to the specified modes of operation. Table 1 below provides an outline of the terminology that will be used throughout this document.

**Table 1: Operational Mode Terminology**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **Single String Retuning** | This is a mode that will take in the full signal of the guitar with all six strings. It will be able to pick out one string that the user selects, and tune it to the desired pitch. It will leave all other strings unaffected. |
| **Electronic Capo** | This mode will take in the full signal of the guitar with all six strings. It will be able to shift the whole signal by a selected amount of the users choosing. |
| **Dynamic Tuning** | Dynamic Tuning will allow the user to define his or her custom tuning. By using a combination of Single String Retuning and Electronic Capo, the user can individually set each string to the desired pitch. |
| **Auto-Tuning** | This mode will keep the user in the tuning that has been chosen while the guitar is played. When the guitar starts to become out of tune, the pitch shifter will detect and correct the pitch of the string to maintain the desired pitch. |

# System Requirements

## General Requirements

The GuPS system shall interface with a standard electric guitar and amplifier using 1/4" connections and will perform retuning of the guitar without mechanical adjustment of the tuning pegs. The system shall support user configuration through an interactive display and control interface. This interface will allow initial calibration of the unit as well as operational mode customization and selection. Additionally, the system shall be a standalone unit which provides its own power either through a [9V] battery or DC power supply.

## Operational Modes

The GuPS pitch shift unit will offer several operational modes that satisfy different user demands. The system shall provide a single string retuning mode for basic drop tunings, and an electronic capo mode for full-signal pitch scaling. Additional features should include a dynamic tuning mode for switching to tunings where multiple strings are effected as well as an auto-tune mode which will be used for maintaining the current tuning of the guitar. All processing should be performed such that latency is minimized below an audibly detectable level of [50ms].

### Single String Retuning

In this mode, the GuPS system shall take in a polyphonic, analog signal and have the capability of isolating the frequency content of a single string. Based on the user selected tuning, the system shall be able to shift only the appropriately isolated string to the desired frequency. The shifted signal and the remaining unaltered signal shall be recombined into a single output signal. This processing should be performed as to preserve the higher order harmonic content of the individual string beyond its fundamental frequency as to reduce the artifacts introduced into the overall output signal.

### Electronic Capo

This operation mode shall take in a single polyphonic, analog signal from the guitar and shift the entire waveform up or down by a desired amount, specified by the user. This feature will allow pitch shifting in either direction by at least one full octave. This mode shall accomplish pitch shifting without truncating the duration of the incoming signal.

### Dynamic Tuning

The GuPS system should provide dynamic tuning of an incoming polyphonic, analog waveform by combining the processing techniques of the single string and electronic capo modes. The string detection and isolation capability of the single string mode may be used to isolate individual strings into separate signals. The electronic capo mode may then be used to shift these signals independently by varying amounts to accomplish more complex tuning changes. This operational mode should recombine the signals into a single output signal. The system should provide dynamic tuning from standard tuning to a list of preset tunings. The system should also allow the user to specify their own tunings not found in the preset list as well as add tunings to the preset list.

### Auto-Tune

The GuPS system should include a real-time auto-tune feature to maintain the current tuning of the guitar. If implemented, this mode should evaluate the incoming analog signal at least once every [5 seconds] and scan all six strings to check and possibly correct the pitch of each string if it is out of tune from the current tuning setting.

## External Interface Requirements

In addition to a user interface on the prototype and production unit, the GuPS system should provide an additional external user interface for maintenance, calibration and utility functions. This may be implemented as software that runs on a user's laptop or as a smart phone application.

### Utility Application

The utility application of the GuPS system should allow wireless connection to a user device through Bluetooth Low Energy (BTLE) for sending and receiving data from the stand alone unit. This should include detailed tuning information not presented to the user through the standard interface on the pedal itself such as how far off the tuning of any string is from the nearest note in cents. Other information should include a differential calculation between the input source signal and the shifted output signal. The application may also be used when first calibrating the device with the user's guitar. Additionally, the utility application of the GuPS system should include a graphical representation of the input and output signals, similar to an oscilloscope.

# System Verification

## Verification Techniques

In this section several verification requirements have been defined to assist in testing whether the system requirements for the Guitar Pitch Shifter have been met. The development timeline has been broken up into three phases:

1. Software Simulation
2. Hardware Simulation
3. PCB Layout

Each of the three phases has different requirements, but all three phases shall demonstrate single string retuning, should provide an electronic capo mode and may include a dynamic mode or mobile utility application.

## Phase I - Software Simulation

Phase 1 will assist in verifying that each system requirement works in theory through software simulation. Inputs shall be prerecorded and shall not be processed in real time.

The simulation program should be created using a simulation software package, such as Octave or MATLAB. A library of functions should be created to support all programs needed and all relevant data which verifies each algorithm should be plotted.

Where required, all code referenced or used shall be properly sited. All workspaces should utilize revision control applications, such as CVS, Git or GitHub, to maintain all code. A developer’s manual shall be created and shall include all function inputs and outputs. The developer’s manual should include a brief functional description for all functions. Most of phase 1 is required prior to the start of phase 2, but some of the components of phase 2 can begin prior to phase 1.

## Phase 2 - Hardware Implementation

Phase 2 will assist in verifying that each system requirement works realistically through hardware simulation. Phase 2 has three major components: interface, hardware algorithm and visualization.

There shall be an interface between the guitar and device model and also between the amplifier and the device model. There shall be an algorithm that can behave similar to the software simulation, but modified as needed for any hardware devices. There shall be a method for verifying that the hardware implementation is correctly functioning. Inputs should be processed in real-time and any delay should be minimal.

The hardware algorithm could be developed on a simulation hardware package, such as Xilinx ISE or Altera Quartus, and the completed program should contain a library of modules, that support all programs as needed. All relevant data should be plotted on tools such as ISim or ModelSim, to verify each algorithm meets the system requirements.

For visualization, pre-existing hardware devices may be used, such as oscilloscopes or spectrum analyzers, but may also be implemented with a graphics toolkit such as GNU Radio, OpenGL or FLTK/QT. The visualization method should be able to show both the input and outputs in real-time.

Where required, all code referenced or used shall be properly cited. All workspaces should utilize revision control applications, such as CVS, Git or GitHub, to maintain all code.

Both a developer’s manual and a user’s manual shall be created. The developer’s manual shall include all modules, inputs and outputs. The developer’s manual should include a brief functional description for all functions or modules and block diagrams for each module. The user manual shall include instructions on how to power, connect and run a live demo. The user manual shall include figures and tables where needed.

## Phase 3 - Production Unit Testing

Phase 3 is the final phase and requires phase 2 be complete. Phase 3 can be broken into three major components: PCB design, production quantity and cost.

The PCB design should be no more than twice the final volume of the phase 2 devices. There will be a limited production phase to build 100 units, if the functionality of the limited production devices goes well, the design will be expanded to run lots of 1000, 3000, and 6000 units. And the cost per unit should be no more the $500.