Gatorscribe

Preliminary Design Report

Scribes Danny Gonzalez (EE) | Tyler Tucker (EE/CS)

# Summary

Gatorscribe is an embedded system that converts musical input into sheet music, designed to simplify the process of creating music. The user will have the freedom to record using any instrument using a microphone or a direct line in. The user will also have the freedom to amplify the input signal and the output signal. The user will record the song along to a metronome, where both signals will be played out during the recording. Song and recording customization options will include song title, BPM, time signature, key signature, and MIDI playback instrument. These options are chosen using a touch screen color LCD. Once the recording is completed and the user wishes to generate a MIDI file, one will be created and sent out to a PC. Once attained, the user can view the sheet music and playback the MIDI file using the playback instrument chosen by the user using third party software.

# Objectives

The main goals of Gatorscribe are to achieve monophonic pitch detection of any instrument or voice, provide a simple user experience using a touch screen LCD and foot switches, and perform automatic MIDI file transfers to a PC. This solution will allow any musician to use Gatorscribe without the need for technical proficiency. Smartphone users will feel right at home on the LCD and should be familiar with a simple micro-USB cable, which will double as a power supply and a channel to transfer the produced MIDI files. Recording can be triggered using a small foot switch in lieu of a spot on the LCD screen so that the musician can retain a hands-free process of signaling the beginning and end of the song. Another small footswitch will be used as a tap-tempo to capture the BPM the user wants to use. We plan to encase the entire device in a professional-looking box made in Solidworks.

# Technology

Gatorscribe uses an Atmel ATSAMV71Q21 ARM Cortex M7 processor clocked at 300 MHz. We decided on this processor because of its high clock rate, abundance of GPIOs including SPI and TWI, and ease of interface to both an external LCD and codec. Sampling music can become very demanding if high resolution is desired, so we wanted a µProcessor that would not bottleneck during operation due to speed limitations. Finally, the ease of testing an external LCD and codec on an ATSAMV71Q21 development kit used for initial testing will accelerate the development process. For example, the development kit features information and basic code for interfacing with a specific codec and external LCD driver, which diminishes the learning curve.

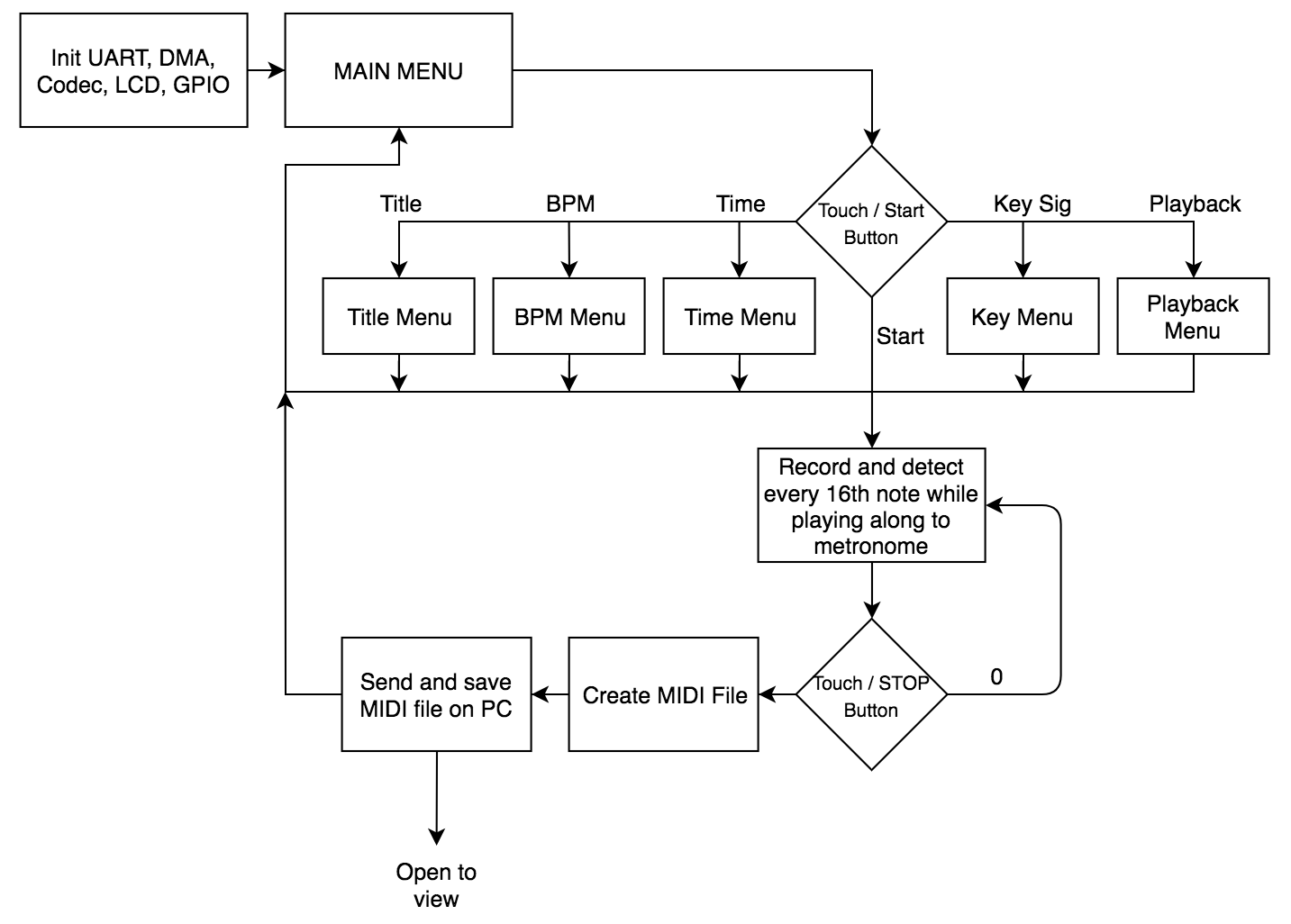
The main input to the product is a ¼” jack that is a music industry standard used to connect instruments to amplifiers. If the instrument is designed to use a 1/8” jack, a simple metal converter can be an easy and cheap solution to connecting the instrument to the board. The output signal is sent to a 1/8” jack (also known as an aux jack), as most speakers use this standard as an input source. Both input and output paths from the codec will feature simple amplifier circuits to ensure a high enough input volume for accurate pitch detection and create an audible output. The two foot-buttons will connect to the board through another ¼” stereo cable, as these cables are both common and robust. Also, they provide the two channels (left and right) necessary to detect each switch. These switches will correspond to starting/stopping the recording and determining the BPM.

# Hardware Diagram

 Hardware Components

* Atmel ATSAMV71Q21 ARM Cortex M7
  + Running at 300MHz
  + Single- and double-precision HW Floating Point Unit (FPU)
  + DSP Instructions, Thumb®-2 Instruction Set
  + 2048 Kbytes embedded Flash
  + 384 Kbytes embedded Multi-port SRAM
  + LQFP144, 144-lead LQFP, 20 x 20 mm, pitch 0.5 mm Package
* WM8904 Audio Stereo Codec
  + Stereo digital microphone input
  + 3 single ended inputs per stereo channel
  + 1 fully differential mic / line input per stereo channel
  + I2S Support
* FT230X USB to UART FTDI IC
* maXTouch Xplained Pro LCD
  + ILI9488 LCD Driver
  + 480x320 Resolution
  + maXTouch capacitive touch screen controller
  + Parallel interface (up to 18-bits)

# Software Flowchart



# Pitch Detection

To perform monophonic pitch detection in real-time for music transcription, a robust and computationally efficient method is required. We plan to implement the YIN pitch detector. The YIN pitch detector is known to be robust in finding the fundamental frequency of any signal. This is vital since musical instruments and voices have many harmonics present, however we wish to find the fundamental. The algorithm, in its original implementation is very computationally expensive, involving an autocorrelation of the input signal. The complexity of an autocorrelation is O(N2). So, if 2048 samples of data are used for pitch detection, that means we will have over 4M multiply accumulates. However, this computation can be reduced by computing the autocorrelation function in the frequency domain. Since the autocorrelation function is simply the Fourier transform of the power spectrum, we compute it by taking the FFT of the signal, multiply it by its complex conjugate (pointwise), and then take the inverse FFT to obtain the autocorrelation. This improves the complexity to be O(3\*N\*log2(N)). So, for 2048 samples, we have just under 70,000 computations, a huge improvement to the original autocorrelation.

To detect music notes, we will be detecting every 16th note played, using a metronome to know when to perform the detection on which segment of audio. So, if four 16th notes are consecutively recorded, it will be interpreted as a quarter note. We will also be computing the average power of each note. This is useful when the musician plays the same note consecutively and the note needs to be separated into two distinct notes. We can do this by simply comparing the power of each 16th note.

# MIDI Creation

While playing along to the metronome, a log of MIDI events (individual notes containing their pitch, velocity, and rhythm) is created. Once the user has ended the recording, a standard MIDI track file is created in binary and sent over UART to a PC. This file is received from the PC that has a Python script constantly waiting for data to be received over serial. Once all of the information is transferred, the script correctly formats and saves the MIDI file, making it instantaneously ready to be opened in 3rd party software to view the sheet music and playback the recording. The sheet music will also display the time signature, key signature, song title, and BPM, and be played back with the playback instrument specified by the user.

# Responsibilities

* Danny
  + DSP
  + Pitch Detection
  + MIDI Transcription
  + Automatic MIDI File Script
  + Altium
  + GUI Design
  + LCD
* Tyler
  + Altium
  + GUI Design
  + Audio Amplifiers
  + Solidworks
  + Foot Switches

# Timeline

