# Virtual Grand Piano

by

Jeongsub Lee

Zhi Lu

Final Report for ECE 445, Senior Design, Spring 2018

TA: Mickey Zhang

May 2018

Project No. 64

# Abstract

The Virtual Grand Piano is a project that utilizes FPGA to generate music notes with varying pitch and amplitude controlled by the user's fingers. Our design uses force sensitive resistors to collect how hard the user is pressing on their fingers, each of which is mapped to a specific note. The RTL modules on the FPGA board utilize sine tables and clock divider to generate the specific frequencies. Then the force data transmitted wirelessly are used to calculate the amplitude adjustments.

# Contents

# 1. Introduction

## 1.1. Objectives

Digital pianos currently available in the market are expensive, heavy and non-portable devices that require meticulous maintenance and large amounts of power. A piano player may require a portable instrument at short notice to practice or test musical pieces without wanting to travel all the way to a studio. We plan to explore a possible solution to this problem by designing a virtual instrument that contains no moving parts, is extremely portable and contains all the functionality and sound design of a digital keyboard.

We are proposing to implement a virtual keyboard in a glove. We are planning to do this by having the player wear a glove with pressure sensors, and a wireless transmitter on each hand. The player can then virtually play the piano with 10 fingers as each finger is hardcoded to a corresponding note in the main module.

## 1.2. Background

The digital keyboard is an extremely versatile instrument for any musical artist. It can be used as a MIDI controller for a custom synthesizer or as a digital piano that authentically reflects the sounds of a traditional piano. It is often the case that an artist may want to test out a melody on the fly or practice a piece without having access to a physical keyboard. With the increasing sophistication of image processing techniques and fast processing times provided by hardware components we are planning to overcome this problem completely virtually. The only inputs required by the instrument for emulating a digital keyboard are the movements of the player's fingers. While a flat interface may not provide the feel of a traditional piano it could be immensely useful as a portable solution and may be set-up in compact spaces that may not accommodate a full piano. It is especially suitable as a MIDI controller which has become an essential part of the modern music production process [10]. The player's inputs would be wirelessly transmitted to a central control unit that would process the note and velocity of the player's movements and feed it into an audio synthesizer.

## 1.3. High-Level Requirements

● The system must recognize and trigger the correct note played by the user with the appropriate sensitivity reading.

● The system must be portable and may be deployed on any flat surface if calibrated appropriately.

● The system must be reasonably fast, processing the sensor inputs and triggering the appropriate key with minimal delay.

## 2. Design

There are 2 main components in this design: the main module and the glove module.

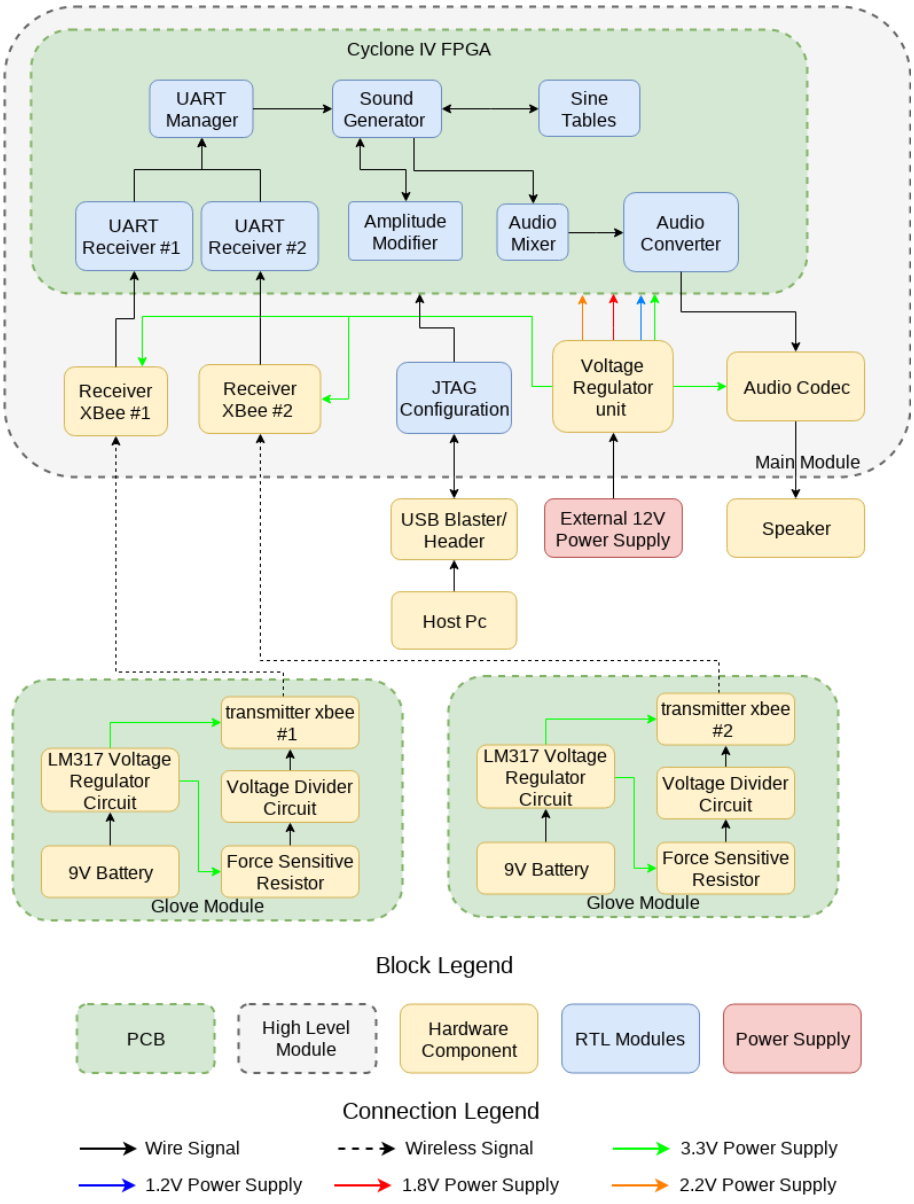


Figure 1. Block Diagram

## 2.1 Main Module

The main module communicate to both of the glove modules, and process the data collected from each in order to generate the desired sound. Specifically, it has the following responsibilities:

1) Produce the digital sound signals for the corresponding notes
2) Processing the force data collected from the glove module
3) Use the force data for each finger to adjust the volume of the note being generated
4) Use audio codec to convert digital sound signal to analog signal, and produce the sound through line-out

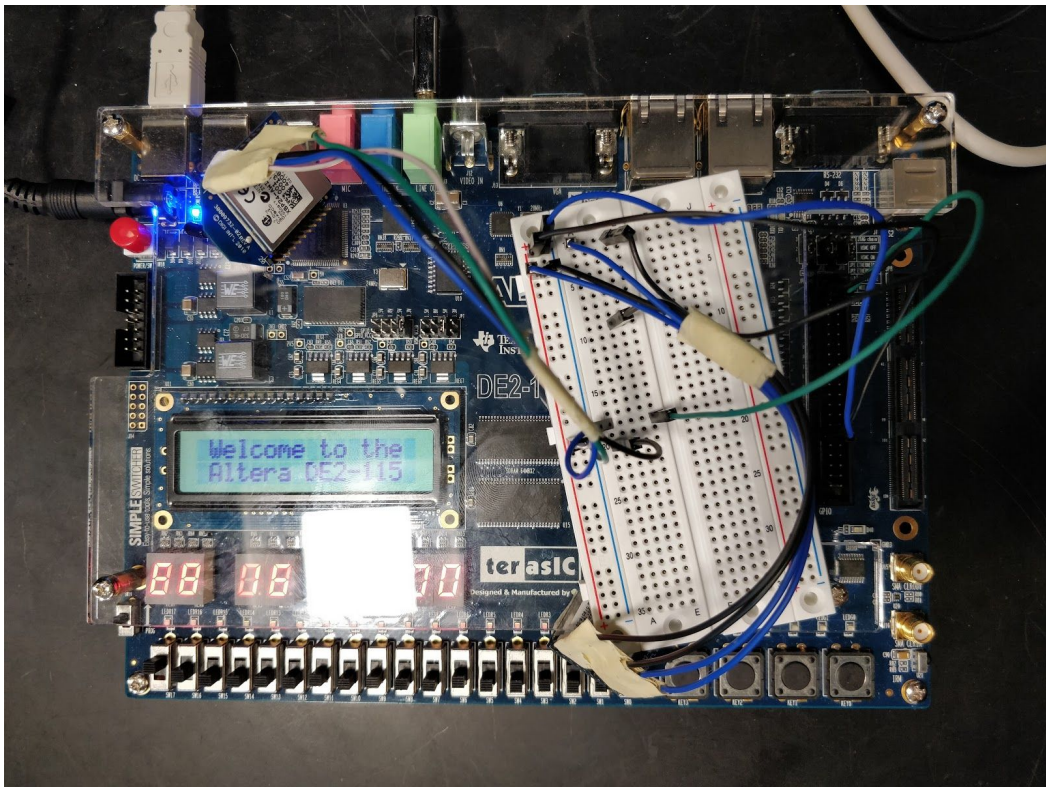


Figure 2.1 An overview of the physical design of the main module

As shown in the figure above, the main module consists of the following:

1) Two XBee RF modules acting as receivers
2) One Altera DE2-115 development board, of which the followings are used:
    a) WM8731 audio codec
    b) Cyclone IV FPGA chip

Digital circuitries are built upon the FPGA chip to facilitate the functionalities described above. Each module's design, configuration and implementation are explained in greater detail below.

### 2.1.1 Cyclone IV FPGA

The Cyclone IV FPGA chip is used to facilitate the digital circuitries of our project, with which the data processing and sound generations are done. The reason for choosing this this particular FPGA chip is due to the availability and familiarity of the DE2-115 development board. The pins/connections used in our design are explained in the following table.

**Table 2.1.1 Cyclone IV FPGA Chip Pin/Signal Descriptions**

| Pin/Signal Name | Pin Number | Description |
|---|---|---|
| CLOCK_50 | PIN_Y2 | 50 MHz clock used for all the RTLs |
| GPIO[0] | PIN_AB22 | Serial communication channel for receiver XBee module #1 |
| GPIO[1] | PIN_AC15 | Serial communication channel for receiver XBee module #2 |
| GPIO_3.3V | Non Programmable | Provide power for both of the receiverpowerpower XBee modules |
| GPIO_GND | Non Programmable | Provide ground for both of the receiver XBee module |
| AUD_DACLRCK | PIN_E3 | Audio codec DAC Left and Right channel clock |
| AUD_DACDAT | PIN_D1 | Audio codec DAC data |
| AUD_XCK | PIN_E1 | Audio codec control clock |
| AUD_BCLK | PIN_F2 | Audio codec bitstream clock |

The compiling and programming of the digital circuitries are done through the Quartus II software version 15.0. The design and implementation details of the digital circuitries are explained in section 2.1.4.

### 2.1.2 WM8731 Audio Codec

WM9831 is needed to convert the digital audio signal into analog signal, which would then be sent to line-out and can be listened to with either headphones or speakers.

In order to use the codec, its' configurations must be set up correctly corresponding to the interface. There are a total of 11 9-bit configuration registers R0, R1, .. , R9 and R15. Each of these registers is responsible for one particular area of the configuration, and each bit or group of bits corresponds to one setting. Each setting has a default value, and can be configured individually through I2C protocol.

In the case of our design, our RTLs are written so we can work with the default codec configuration, which is detailed in table in Appendix B.

### 2.1.3 XBee Series 1 RF Module (Receiving)

The XBee is a series of very popular and reliable wireless networking modules by Digi International [1]. The module we used in our design is the XBee S1 RF, which is responsible for the wireless transmission of the force data from the glove module to the main module.

There are a total of 4 XBee modules in the whole design, with 2 different roles: transmitters and receivers. Each transmitter is paired with a receiver, and each pair communicates the force data of 1 glove module to the main module.

The 2 transmitter XBees are part of the glove modules, used to sample and transmit the force data. They will be explained in greater detail under the glove module section.

The 2 receiver XBees are connected to the FPGA, and responsible for receiving the data sent by the corresponding transmitter XBee module. There are 2 receiver XBees, with each one paired to a transmitter. Each XBee outputs the data packet it receives through its DOUT pin, in the serial UART protocol. The DOUT pins are connected to GPIO pins on the FPGA, and FPGA uses UART receiver to receive the data.

In order to use the XBee modules, they need to be configured and programmed correctly through the XCTU software, provided by Digi [2].

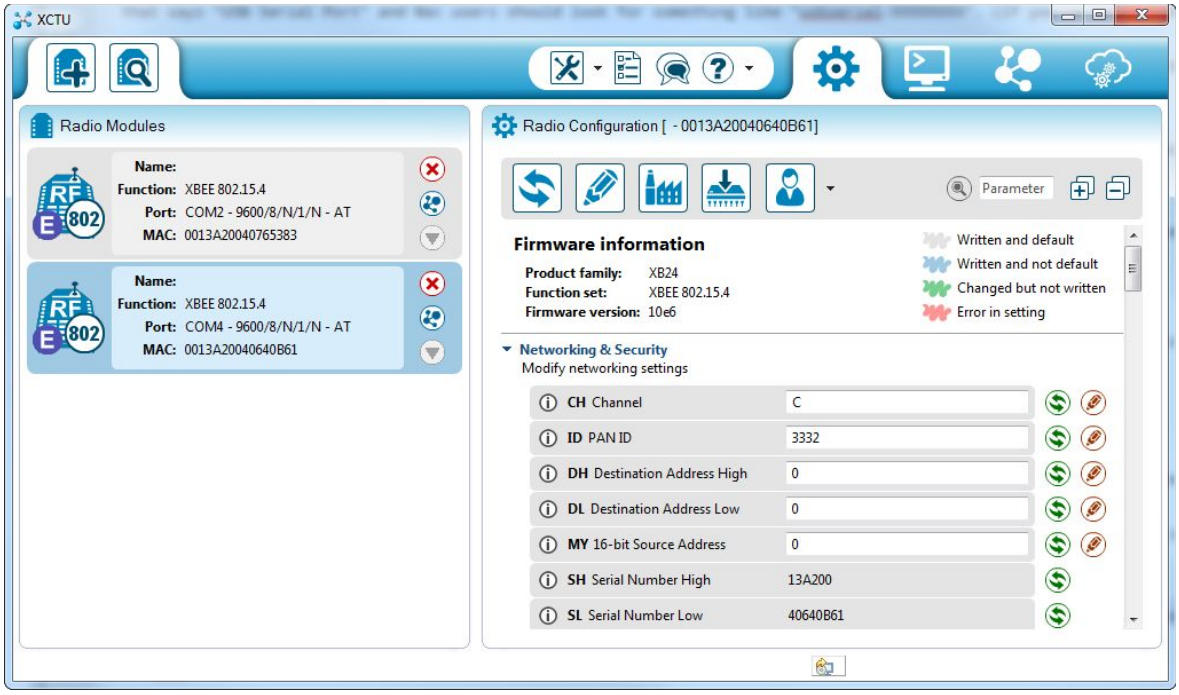


Figure 2.1.3 An overview of individual XBee configurations in XCTU.

To configure the XBee modules as receivers, the following parameters must be programmed:

1) **Channel**: The pair of transmitter and receiver must share the same channel
2) **Personal Area Network (PAN) I**D: The pair must also share the same PAN ID
3) **Destination Address**: The source address of the pairing transmitter
4) **Source Address**: An arbitrary source address for the current XBee module
5) **Baud Rate**: The transmission rate for the UART protocol, must be consistent to the rate for the UART receiver on the FPGA.

All of the above parameters can be arbitrary, as long as consistent throughout the design. However, we chose the maximum baud rate of 115200 bps, in order to minimize the latency caused by data transmission.

### 2.1.4 Digital Circuitry/RTL design

The digital circuitries are the largest part of the main module design. The figure below shows an overview of the data flow and roles of each module. Specifically, how the analog data sampled on the glove modules are transmitted, processed and combined with the audio signal data to produce and modify the sound output.

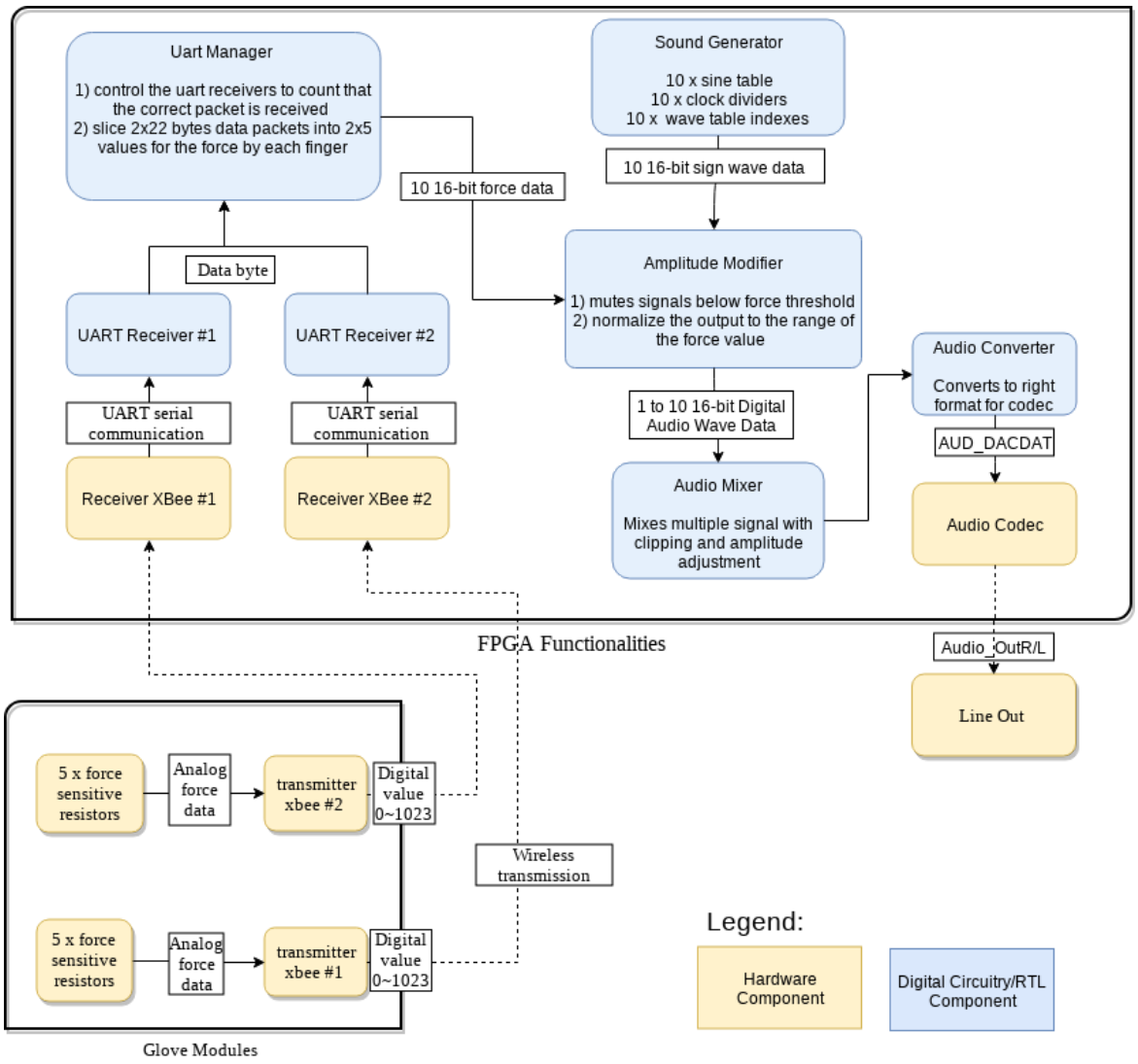


Figure 2.1.4 Data Flowchart

The individual modules and their functionalities are as follows:

1) **UART Receiver:** Receive a byte of data from a serial channel with UART
2) **UART Manager:** Controls the UART receivers and extract and index the individual force data from each finger
3) **Sound Generator:** Divide the 50MHz clock to create the 10 desired notes, and control the Amplitude Modifiers and Audio Mixers

4) **Amplitude Modifier**: Use the force data for 1 finger to adjust the amplitude of the audio signal
5) **Audio Mixer**: Mix multiple audio signals together

#### 2.1.4.1 UART Receiver

This module samples 8 bits of data from the serial data line every complete cycle, and outputs the byte of data to its top level.

There are 2 inputs and 2 outputs for this module.

Input:

- **Clock** - the 50 MHz clock from the top level
- **Serial Data Line** - The channel on which the data will be transmitted and sampled by the receiver

Output:

- **Received Byte** - The received byte of data
- **Data Valid** - Indicates a whole byte of data has been received

The design of this module is based on and modified from the online tutorial [4]. It basically consists of a clock dividing counter and a state machine with 5 states: IDLE, START, DATA, STOP and CLEANUP.

The IDLE state is the default state, for when there's no incoming data. The START state is when a "start bit" is detected on the serial line (the line goes from high to low), indicating the start of the transmission. The DATA state samples 1 bit from the data line each time, and is repeated 8 times. The STOP state is entered after the 8 bits of data have been sampled, and it waits until the "stop bit" is finished on the serial line (line goes from low to high). The CLEANUP state outputs the Data Valid bit, indicating the end of a UART transmission. The clock counter is used to wait for the correct number of clock cycles for the selected baud rate, before the data line is sampled. The number of clock cycles needed to wait is calculated as follows:

$$count = \frac{Clock\ Frequency}{Desired\ Baud\ Rate} = \frac{50\ MHz}{115200\ bps} = 432 cycles/bit$$

Equation 2.1. Calculation of clock divider count for UART receiver

#### 2.1.4.2 UART Manager

The UART manager is responsible for the following functionalities:

1) Controlling the UART receiver modules to receive the correct number of bytes
2) Extract the force data corresponding to each finger from the received packets

The receiver XBee module receives data in the form of data packets separated by each byte, and the packets have pre-determined formats identified by the position of the byte. The size and interpretation of the data packet varies according to the configuration. With the configuration in our design, as mentioned in 2.1.3, each XBee data packet is 22 bytes long.

Among the 22 bytes of a packet, we are only interested in the bytes that represent the digital data of each FSR. As indicated by the frame interpreter tool in the XCTU software [2], each FSR data is 2 bytes long, with the first one located at byte 10 and 11, second one at byte 12 and 13, and so on.

After receiving a complete packet, the XBee module sends the packet to the UART manager through the serial data line with UART protocol. The UART manager keeps a counter that not only counts how many bytes have been received so far, but also acts as an index for where the current byte's location is.

Since we are collecting the data from both glove modules, with each having 5 FSRs, we have a total of 10 16-bit registers that are used to save and pass the received FSR data. When the byte counter/index hits the corresponding location for each FSR, the received byte is passed into the corresponding FSR data register.

After 22 bytes have been received, the UART manager indicates that a whole packet has been received and resets the counter.

#### 2.1.4.3 Sound Generator

The functionality of the Sound Generator module, as the name suggests, is to generate and output digital audio signals of desired pitch and amplitude.

As an overview, the sound generator creates 10 notes by using 10 sine tables and 10 different clock dividers to generate 10 sine waves of different frequencies. Moreover, it uses 10 Amplitude Modifier (2.1.4.4) modules, to modify the amplitude according to the 10 FSR data passed in by UART manager. At last, it uses a Audio Mixer (2.1.4.5) to combine the signals with modified amplitudes, and generate one output signal.

The pitch of a note is determined only by the frequency of the sound wave, and the tone is determined by the shape of the wave [5]. As a result, we decided to use sine tables with different indexing frequencies to generate our notes. Sine tables are look up tables with

values that represent a sine function. In our design, the sine tables have 8 bit resolution, meaning each sine wave is represented by 256 values. Each sine table is traversed with a table index register, and by changing the speed of the traversal, we can generate sine waves of any frequency. The number of clock divider cycles needed for any frequency can be calculated as follows:

$$Cycles = \frac{Clock\ frequency}{note\ frequency \times sine\ table\ period}$$

Equation 2.2. Sine table clock dividing cycles calculation

#### 2.1.4.4 Amplitude Modifier

The purpose of this module is to modify the amplitude of the a sine wave according to the corresponding FSR data.

To perform the modification, we simply perform a min-max normalization of the sine wave value according the FSR data. The FSR value, sampled using the transmitter XBee module's onboard ADC (analog to digital converter) range from 0 to 1023 (0x03FF) [2]. As a result, we can calculate the normalized value as follows:

$$normalized = original \times \frac{fsr\ data - min}{max - min}$$

Equation 2.3. Min-Max Normalization for Amplitude Modification

In order to reduce noise, we introduced a threshold for the FSR data. If the FSR data is below the threshold, the corresponding finger is counted as unpressed. The value is determined by the user according to the specific hardware, since every resistor might perform slightly differently. For signals with FSR data below the threshold, we simply mute them by assigning 0 to the output signal.

#### 2.1.4.5 Audio Mixer

Since the audio codec only takes one signal at a time, we need to mix the audio signals in order to play multiple notes simultaneously.

The audio mixing is done by adding the values of multiple sine waves together. The resulting waves look similar to the figure below.

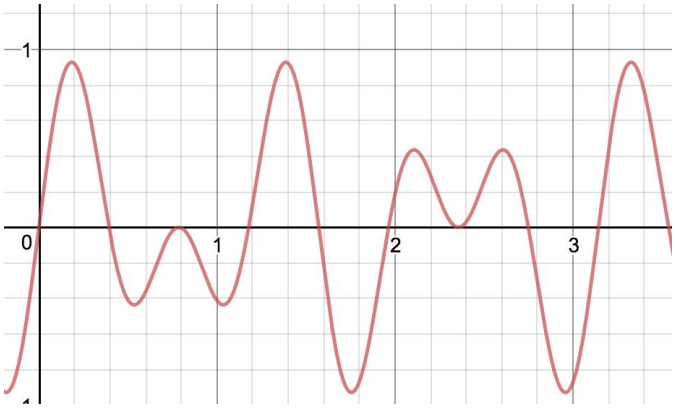


Figure 2.1.4.2 Resulting wave of sin(10x)+sin(6x)

However, the resulting signal's amplitude needs to be adjusted in order to avoid overflow. We first tone down individual signals that have FSR value above the threshold by 50%, by multiplying 0.5 to the individual signal. Then we tone down the overall signal's amplitude by 20% similarly. At last, if the resulting signal is still above the max value of 1023, we clip the value by assigning it as 1023.

### 2.1.5 Voltage Regulation Unit for Main Module

Another voltage regulation unit is designed to supply correct amount of voltage to the main module. Unlike we used single voltage regulator chip for glove module, implementation of the voltage regulation unit for the main module includes multiple regulators with added complexity. In the Main module, the voltage regulator unit is designed to create different voltages as suggested in the table 4 below. Instead of having parallelly listed voltage regulators that create different voltage output. Some of the smaller voltages were created making use of larger regulated voltage as an input.

3.3V output, used to supply voltage to FPGA, WM8731 Codec and Xbee receiver, was created using the same network that we used in the glove module voltage regulator unit. Other voltage inputs like 2.5 V, 1.8V or 1.2V were created using a switching voltage controller chip LM3150 and a linear voltage regulator LP38692 along with rectifier IRF7455. In order to obtain a reliable network various choice of capacitors were used in the circuit in addition to various resistors. The network refers to the voltage network circuit of the DE2-115 Development board voltage regulator units since FPGA and audio codec in our design possess identical specification to that of DE2 Development board.

| Components | Requirements | Observations | Error ratings |
|---|---|---|---|
| XBee RF module | 3.3V | 3.310V | 0.3% |
| WM8731 Codec | 3.3V | 3.310V | 0.3% |
| FPGA(VCCINT) | 1.2V | 1.219V | 1.6% |
| FPGA(VCCIO 3.3V) | 3.3V | 3.317V | 0.5% |
| FPGA(VCCIO 1.8 V) | 1.8V | 1.810V | 0.6% |
| FPGA(VCCA) | 2.5V | 2.508V | 0.3% |
| FPGA(VCCD_PLL) | 1.2V | 1.219V | 1.6% |

Table 2.1.5 Voltage Requirements and observed Voltages of Main Module

The observation of the output voltage of the voltage regulator unit was tested using prototypes of each segments providing 12V output from the oscilloscope. However, it was not testable to check whether the voltage supply was correctly supplying the suggested voltage shown in table in actual PCB prototype. Constraints could vary from error ratings of resistors, wrong path in the pcb design, to current ratings. However, since the design itself is nearly identical to that of DE2 board power supply, the components, and the design was not considered as a factor of failure of the main module. While reviewing the design of PCB, we could observe that some of the path were closed in the schematics however were open in the actual pcb design due to overlapping wires. Perhaps, this could have led to failure or short of main module itself in the end, and stopped us from confirming the tested result in our actual hardware module.

The figures below is part of our main module schematics that is responsible for supplying voltage to different components. For LM3150 output voltage calculation, following equation was used.

$$V_{out} = V_{fb} \times \left(\frac{(R_{fb2} + R_{fb1})}{R_{fb1}}\right)$$

Equation 2.4. LM3150 Voltage Output Calculation

Here, Rfb1 and fb2 refers to two of the right most resistors, Rfb1 being the one connected to the ground and Rfb2 being the one parallely connected to the capacitor of 1.2nF, and 390pF respectively.

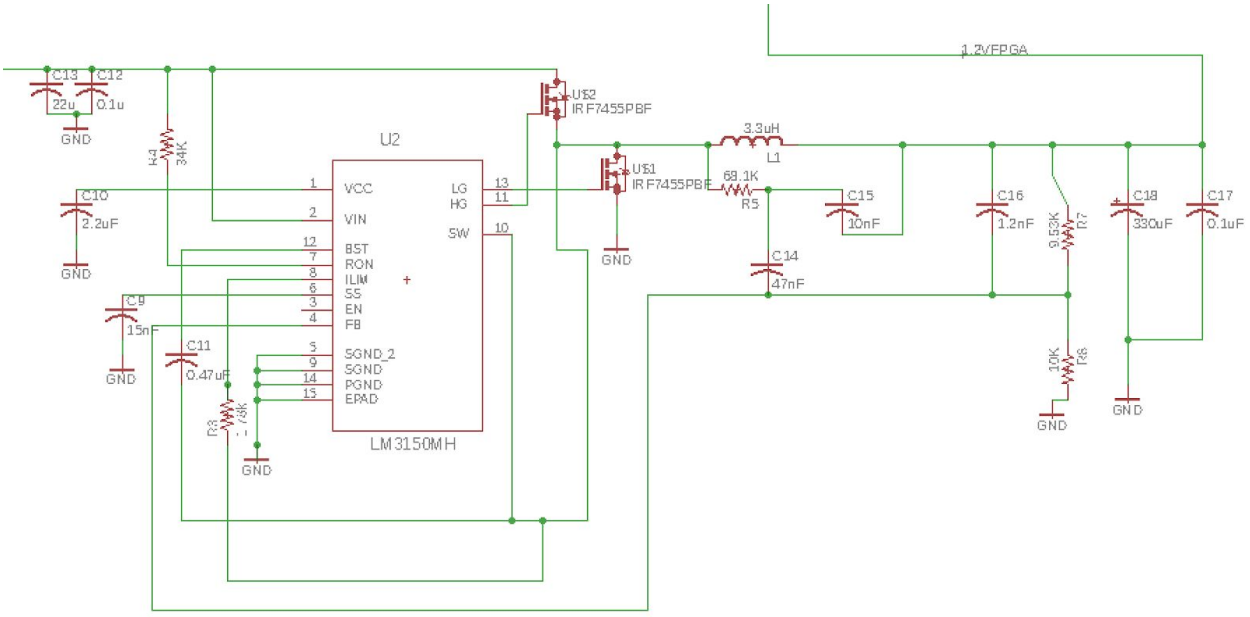


Figure 6. LM3150 (12V input to 1.2V output)

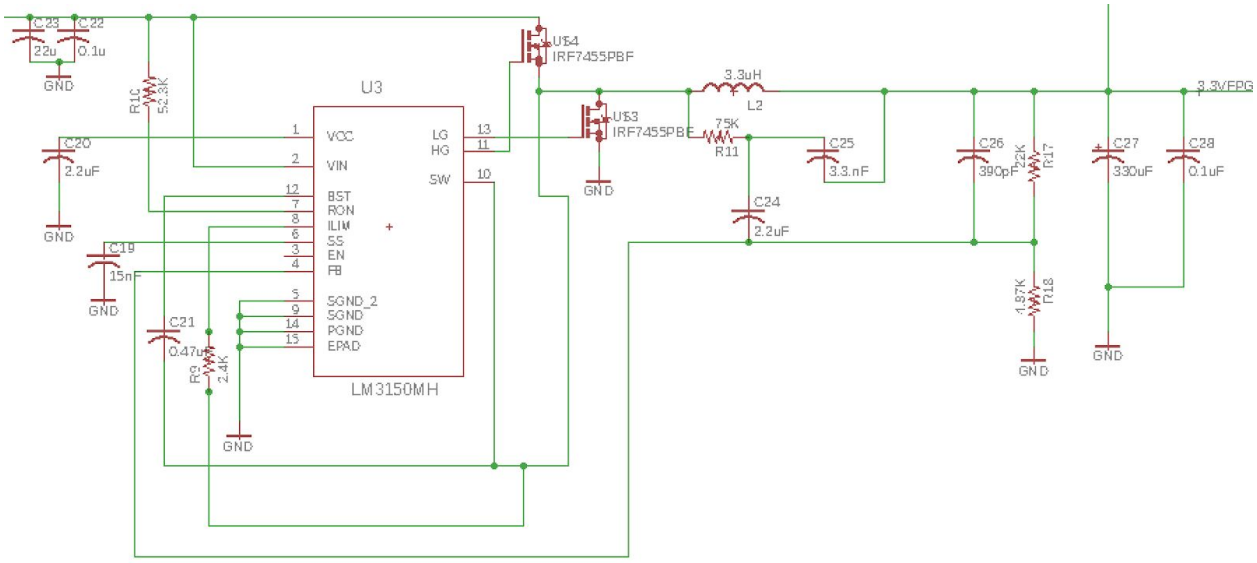


Figure 7. LM3150(12V input to 3.3V output)

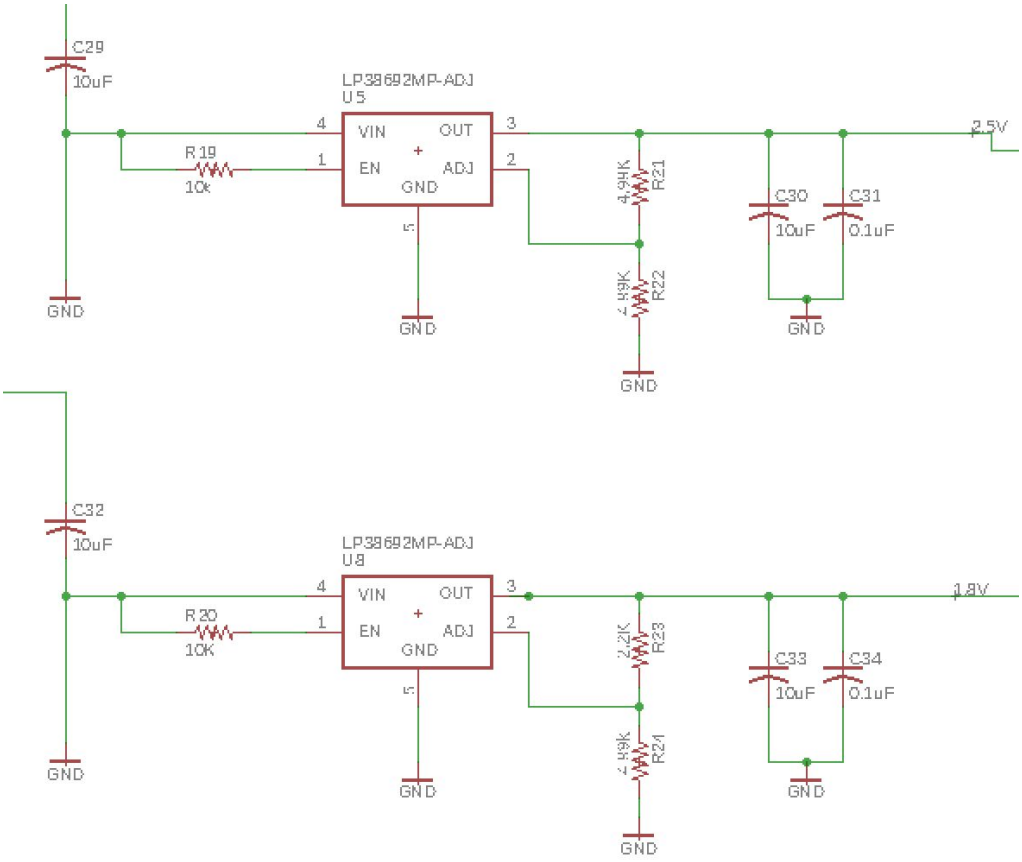


Figure 8. LP38692(LM3150 3.3V input to 2.5V , 1.8V output)

## 2.2 Glove Module

The glove Module is responsible for wirelessly transmitting the pressure data of fingers to the main module, in a digital form. First, the glove recognizes user's finger pressure from five different fingers, then at a optimum resolution, this finger pressure is be translated into several different levels of analog signal. After analog signal is generated, this analog signal then flows into five Analog to Digital converting unit of Xbee. Using XBee communication, the digital data packet is then transmitted to the receiver in the main module.

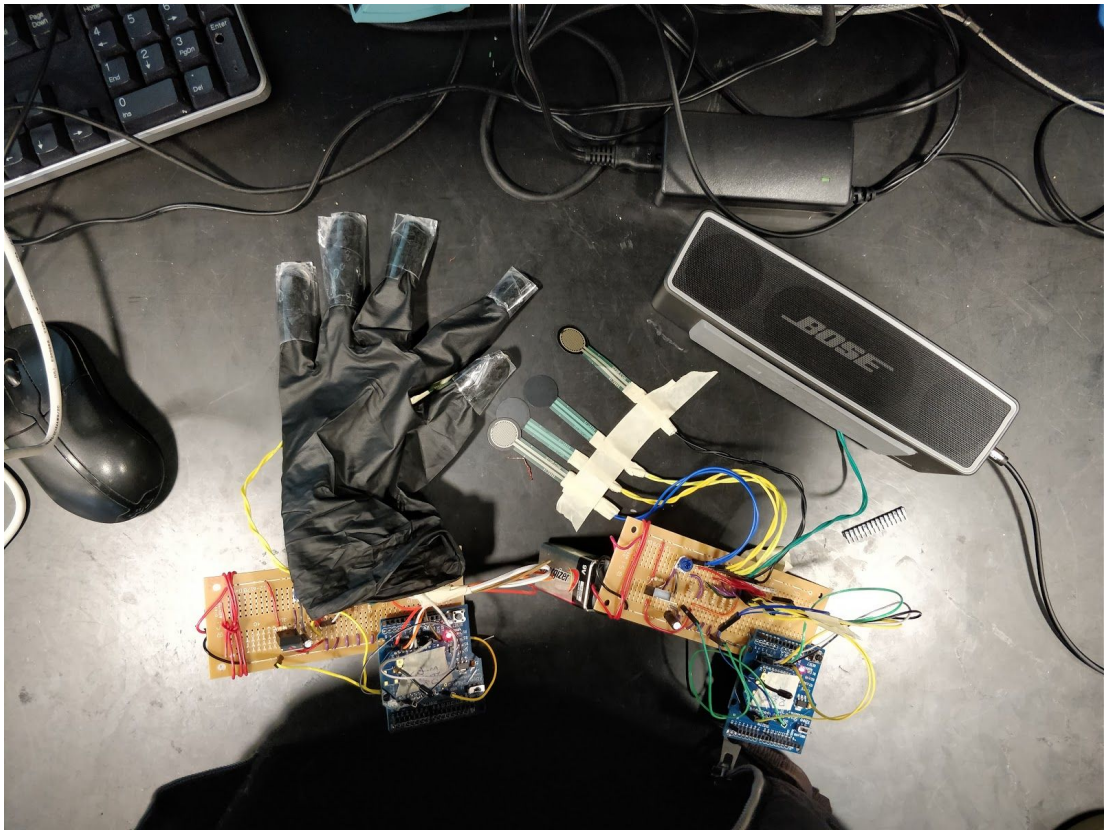


Figure 9. Overview of the glove module's physical design

### 2.2.1 Force Sensitive Resistor (SEN-09375)

For sensing pressure of user's fingers, five force sensitive resistors,(SEN-09375), were used in each glove. The goal of using these resistors is to provide different voltage values into ADC of xbees at high resolution in order to differentiate the pressure of the user's finger on to the sensor. First of all, the maximum output from the FSR network shall not exceed 3.3V +/- 0.3V since this is the required specification of the XBee. Therefore, 3.3 V output voltage

from the voltage regulation unit is used to supply voltage to the FSR sub-circuitry. Then in order to achieve the maximum resolution rating following equation was used to calculate the reference resistance.

$$Rref = Rfsr, min \times \sqrt{(\frac{Rfsr,max}{Rfsr,min})}$$

Equation 2.5 Finding reference Resistance

Although the resistance of FSR varies from 0 to 1M Ohms depending on the pressure, using the entire range of resistance was not an efficient way to create resolution that suits for playing piano. Therefore, the calculation only considered range of (600 Ohms, 600 KOhms ) where 600 Ohms is minimum resistance of FSR at maximum pressure, and 600K Ohms is maximum resistance when nearly no pressure was detected. After calculation, preferred resistance turned out to be about 20 KOhms.

After finding the appropriate reference resistance, the resistance was used in combination with FSR to create different level of voltage outputs based on the resistance of the FSR using simple voltage divider. Although voltage divider implementation possesses limitation as the range of the voltage is not entirely linear, as we decided to make threshold pressure at 100~200g (0x0100), such nonlinearity was not a concern.

$$V\,out = V\,in \times \frac{Rref}{(Rref+Rfsr)}$$

Equation 2.6 Voltage Divider

The Below figure shows the voltage response of FSR with different reference resistance value, by looking at the graphs, we can observe that most of the response are nearly linear after 200g of pressure. However, it was hard to map and plot the actual voltage variation using 20K ohms and our FSR. as the rate of change was to fast, the actual resolution and response was hard to observe. Instead, we instead tested the resolution by hearing to the sound output. At Lower resolution, the pressure sensor could not differentiate the pressure, and hence the volume of sound did not vary greatly. In other words, fromm the smallest to largest volume, the difference was not very recognizable. However, with 20K Ohms resistance, we could easily tell the difference between different pressures as the sound got louder or smaller depending on the finger pressure.

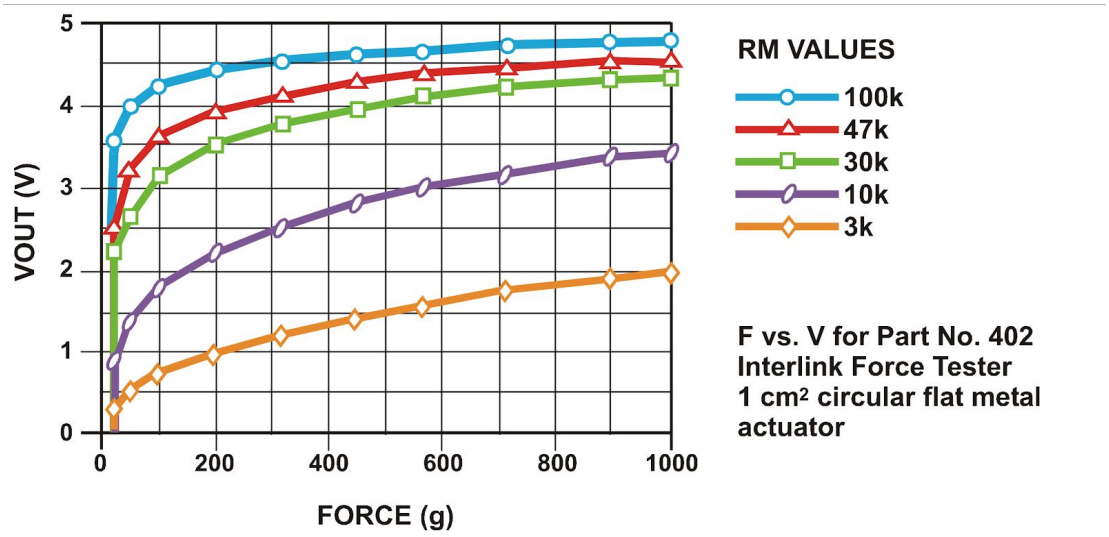


Figure 10. Voltage response of FSR

| FSR Resistance(Ohms) | Analog Voltage (V) | Digital Value | Applied Force(g) |
|---|---|---|---|
| R >1M Ohms | 0V | 0x000 | 0 |
| R < 300 Ohms | 3.3V | 0x3FFF | ~10,000g |

Table 2.3

Following schematic shows how force sensitive resistor is connected to xbee.

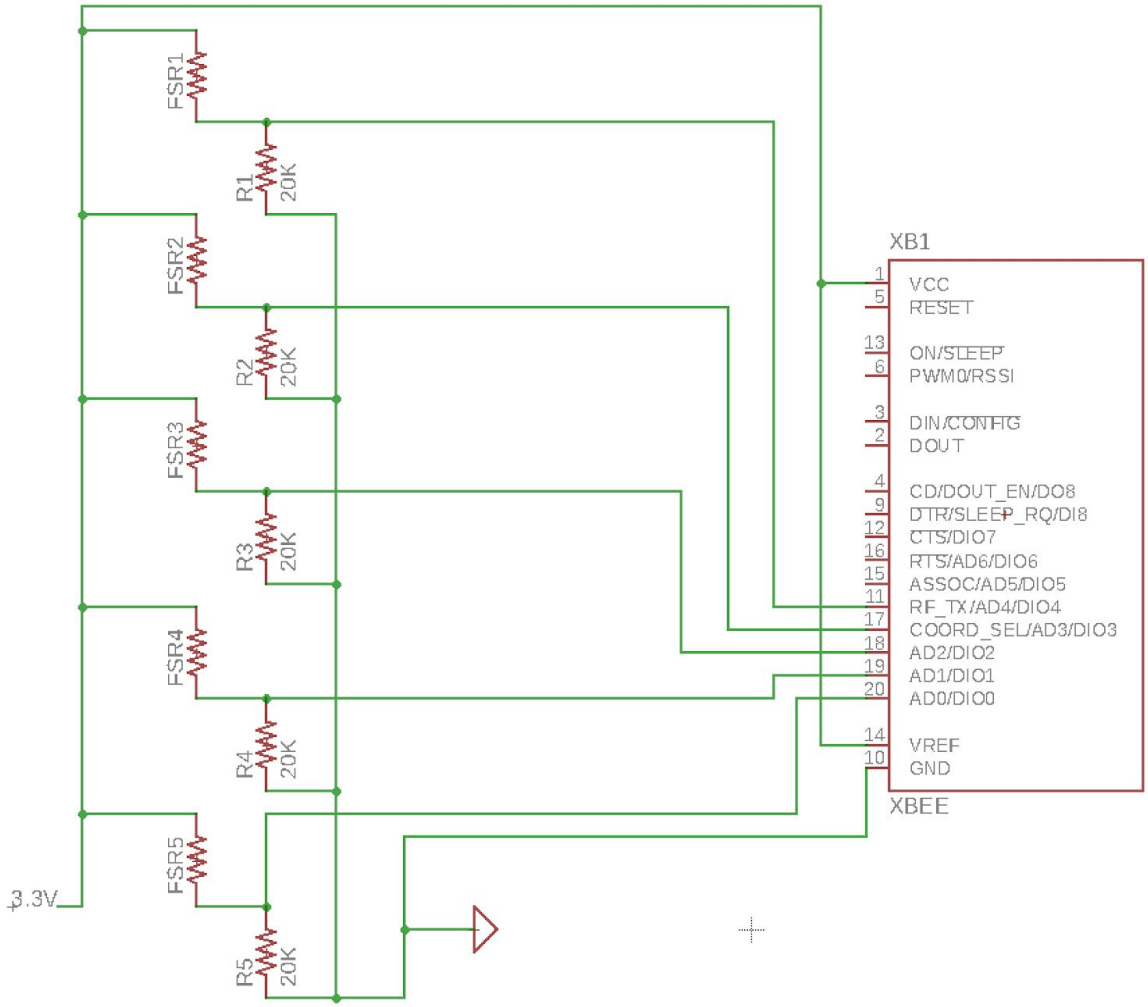


Figure 11. FSR to XBee

### 2.2.2 XBee Series 1 RF Module (Transmitting)

There are one transmitter XBee module on each of the glove modules. Similar to 2.1.3, the modules are configured through the XCTU software. They are configured to sample the analog data from each FSR, convert them to digital data then transmit them to the pairing receiver XBee modules.

Each transmitter XBee module is configured as follows:

- **Channel**: The pair of transmitter and receiver must share the same channel
- **Personal Area Network (PAN) I**D: The pair must also share the same PAN ID
- **Destination Address**: The source address of the pairing receiver

- **Source Address**: An arbitrary source address for the current XBee module
- **DI0...DI4:** 5 input pins configured as ADCs, connected to each of the FSR
- **Sampling Rate:** Maximum sampling rate of 1 kHz is selected in order to minimize latency

### 2.2.3 Voltage Regulation Unit

Throughout our design of modules, the supply voltage requirement varies.

Our gloves module makes use of single 9-V battery per module, and our main module was designed to have 12V power supply initially. However, within the module, sub-components(Xbee, Cyclone IV FPGA, WM8731 Codec) have different input voltage requirements. In order to provide the necessary voltage to each of components, the design makes use of more than one design of voltage regulator circuit.

### 2.2.3 Voltage Regulation Unit for glove Module

First, a single glove module is composed of a battery, voltage regulator unit, xbee, and Force sensitive resistor network using voltage dividers.

Whereas the Xbee's input voltage and reference voltage strictly requires 3.3V, having minimal room for error is very crucial for correctly operating the xbee.

In order to provide 3.3V to XBee, LM317, an adjustable voltage regulator was used with single 0.1$u$F capacitor, a choice of resistor of R1(240 Ohms), 1uF capacitor, and a 1K Ohms Potentiometer. The design made use of potentiometer in order to calibrate the resistance while looking at the output voltage instead of simply relying on the equation to generate the precise output. According to equation 3 below, the voltage output depends on the ratio of R2 to R1. However, even though we have used 240 Ohms resistor for our design, the actual resistance of the resistor could vary. Since small difference of the fixed resistor values could discourage the regulator unit from creating 3.3V output, potentiometer was used to minimize the errors. After several adjustment, the potentiometer was adjusted to within range of 390 +/- 3% Ohms for both modules.

Theoretically, according to equation 3 at choice of 240 Ohms for R1, and choice of 393 Ohms should result in 3.3 Ohms since Iadj is negligible current flowing through the chip.

$$V_{out} = 1.25 * (1 + \frac{R_2}{R_1}) + I_{adj} * R_2(I_{adj} = 0)$$

Equation 2.7 LM317 Output Voltage Calculation

However, as the below table suggests, 3% to 5% errors in registers makes the circuit unreliable for creating precise 3.3V output. So, instead, by looking at the oscilloscope, we adjusted R2 until we found acceptable range of voltage. Such observed R2 values were noted in the following table.

| Vout | R1(240 Ohms) | R2(1K-Pot) | R2(390 Ohms) |
|---|---|---|---|
| 3.24+/- 0.2 | 233 Ohms | 382 Ohms | 371 Ohms |
| 3.303 +/- 0.02 | 233 Ohms | Not Measured | X |
| 3.25+/- 0.2 | 234 Ohms | X | 375 Ohms |
| 3.309 +/- 0.02 | 234 Ohms | 385 Ohms | X |

Table 2.4

The schematics below shows how the LM317 chip, capacitors and resistors were used for designing the voltage regulation unit for glove module.

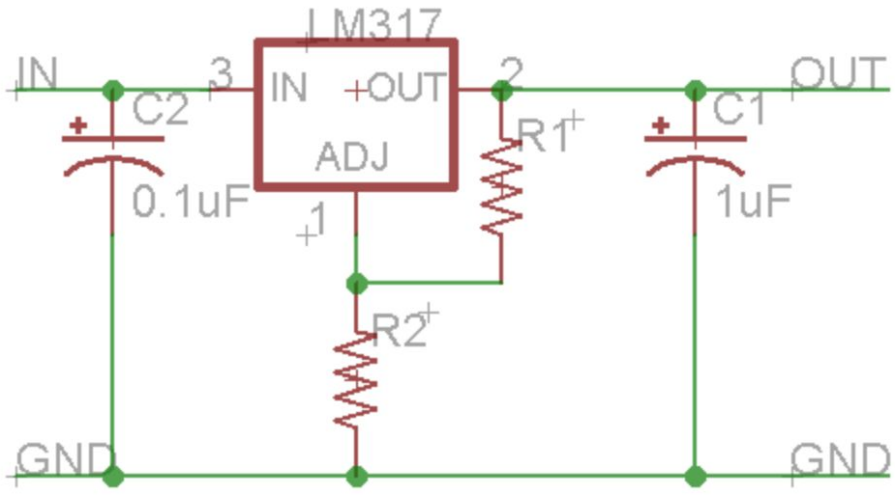


Figure 2.2.3. Eagle Schematics for LM317 Voltage Regulator

# 3. Design Verification

Each subsection of section 3 discusses the verification processes and results for each individual module. Refer to appendix A for more detail regarding the requirements and verifications of the specific module.

## 3.1 Main Module

### 3.1.1 Cyclone IV FPGA

As mentioned in 2.1.1, the RTL modules are compiled and programmed to the FPGA chip through Quartus II and the USB Blaster. When connected to the PC through USB, the USB Blaster works as intended and programs the chip successfully through Quartus II.

The functionalities of the digital circuitries, as mentioned in 2.1.4, can only be verified after programming them onto the FPGA chip. Refer to Appendix A for further details.

The FPGA should be able to receive the FSR data through the XBee module. The pipeline and implementation is explained in 2.1.4.1 and 2.1.4.2. In order to verify this functionality, we mapped the received data to the onboard 7-segment hex display of the DE2-115 board.

The FPGA should also be able to generate notes of required frequencies, it is verified by listening to the sound output and showing the output waveform on the oscilloscope.

### 3.1.2 WM8731 Audio Codec

The audio codec is responsible for converting the 16-bit digital audio signal to analog signal, and outputting them through line out. To verify this functionality, we connected the line out to a separate speaker, and we were able to hear the desired note being generated. Moreover, we can connect the line out to the oscilloscope and look at the sound wave being generated. The figure below shows an sine wave of 1 kHz frequency being generated.

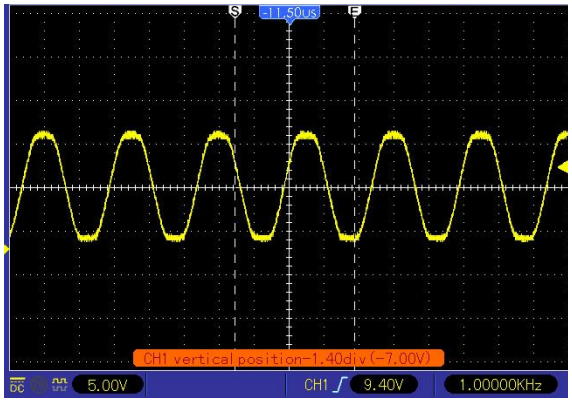

Figure 3.1.2 Image of 1kHz Sine Wave Captured at Oscilloscope

### 3.1.3 XBee S1 RF Modules

With the help of the XCTU software's terminal tool, real time XBee communication can be visualized, as shown in the figure below. Being able to visualize the communication makes it easy to verify the reliability as well as the contents of data transmission

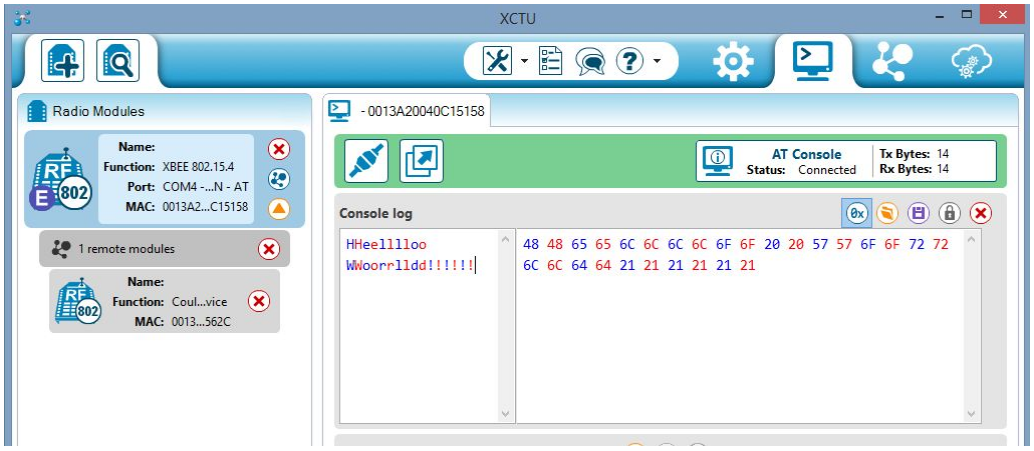


Figure 3.1.3 XCTU XBee Communication Terminal

The first requirements for the XBee is its transmission reliability. It needs to sustain transmission with a reasonable range of at least two meters. It was verified by starting arbitrary continuous transmission with the two XBee modules next to each other, and slowly moving them away until they are about two meters apart. The data transmission was undisturbed.

### 3.1.4 Voltage Regulation Unit

Voltage regulation unit in the main module used test method that is similar to test method we used for Similar to the voltage regulation unit of glove module. Although using xbee shield for testing was limited to 3.3 V, other output voltages could still be tested using the oscilloscope. For all output voltage values of 1.2V ,1.8V, 2.5, and 3.3V had extremely low rate of error, at less than 2%. (table 2.2)

## 3.2 Glove Module

### 3.2.1 LM317 Voltage Regulator Unit

The glove module's functionality was tested using several methods. First, the voltage regulator subcircuit is responsible for generating 3.3V output voltage. In order to check that the subcircuit is generating correct output, oscilloscope was used to check the output voltage at the output node of the regulator circuit. Then, as acceptable output value was

observed, we checked whether the light on the XBee lights up or not. With correct input voltage, red light on the XBee shield blinks. After several trials, we could finally observe the light blinking continuously, and could confirm that the voltage regulator Unit is generating correct output.

### 3.2.2 Force Sensitive Resistor (FSR)

Once the circuit is provided with appropriate voltage, the analog signal fed into analog to digital converters of XBee should be converted into digital values. When no pressure was applied, voltage that we observed on the oscilloscope was closed to 0V, and as pressure was added, the voltage kept increasing until it reached 3.3V. Then, to check whether these values are correctly interpreted into digital signals, we started again from 0 pressure and slowly increased pressure to the maximum. This time, instead of looking at the oscilloscope, we used XCTU to visualize the digital data, which is converted from the analog signal that FSR circuitry has created. As the pressure increased, we could observe digital value increasing, and when the maximum pressure was applied, the digital value reached 0x3FFF the largest digital value that XBee ADC could represent.

# 4. Costs

## 4.1 Parts

**Table 4.1 Parts Costs**

| Part | Quantity | Manufacturer | Unit Price ($) | Actual Cost ($) |
|---|---|---|---|---|
| Perforated Board | 3 | JCLPCB | 1.62 | 4.86 |
| Main Module PCB 1st Tria | 5 | Unnamed Chinese Manufacturer | 3.40 | 15.40 |
| Main Module PCB 2nd Trial | 10 | Unnamed Chinese Manufacturer | 10 | 100 |
| Glove Module PCB | 5 | Unnamed Chinese Manufacturer | 5 | 25 |
| Sen-09375 FSR-400 (10) | 10 | Interlink | 6.95 | 89.21 |
| XBee | 4 | Digi international | 25 | 100 |
| WM8731 CLSEFL | 2 | Cirrus Logic | 3.64 | 7.28 |
| LM317LD | 10 | On Semiconductor | 0.354 | 3.54 |
| ADA 4627-1 | 12 | Analog Devices | 7.761 | 93.13 |
| Arduino for Xbee | 2 | Elemgo | 10 | 20 |
| CB3LV-3I-27M0000 | 2 | CTS Electronic Components | 1.35 | 2.7 |
| CB3LV-3I-50M0000 | 2 | CTS Electronic Components | 1.11 | 2.22 |
| 9V Battery Snaps & Contacts 9V Battery Snaps & Contacts 4" BATTERY SNAP | 4 | Keystone Electronics | 0.70 | 2.8 |
| EP4CE6EE22C8N | 2 | Altera | 11.95 | 23.9 |
| 33DCJ-0202-A | 2 | CONEC | 0.73 | 1.46 |
| 35RASMT2BHNTRX | 2 | Switchcraft INC | 0.92 | 1.84 |
| IRF7455PBF | 6 | Infineon Technologies | 1.43 | 8.58 |
| LP38692MP-ADJ/NOPB | 3 | TI | 1.58 | 4.74 |
| LM3150-500 EVAL | 2 | TI | 25.96 | 51.92 |
| M/F JUMPER WIRES | 36 | N/A | 0.30 | 10.80 |
| Shipping and miscellaneous | N/A | N/A | N/A | 80.24 |
| Total | | | | 649.62 |

## 4.2 Labor

Table 4.2 Costs of Labor

| Participant | Hourly Wage ($) | Estimated Time (hr) | Total ($) |
|---|---|---|---|
| Zhi Lu | 20 | 150 | 3000 |
| Jeongsub Lee | 20 | 150 | 3000 |
| Sum: | | | 6000 |

# 5. Conclusion

## 5.1 Accomplishments

In fact, we have gone through unexpected difficulties while working on the project. One of our teammate left the team, and part of work that he was responsible for remained empty. However, after serious considerations, the current project came up as an alternative to the original design that we projected. Our design was able to play 10 different notes with 10 different fingers. We could also control the volume of the notes that we play by applying different pressure using fingers. Moreover, the wireless communication of our main module and glove module was reliable within 2 meter range.

## 5.2 Uncertainties

Our biggest uncertainty regarding this project is its flexibility. Since our design only has 10 notes, it only covers a little more than a whole octave. We worry about the limited number of songs that the user is able to play with only 10 note. However, with more hardware components, such as a button, and software adjustments, we can add the functionality of changing the octave mapped to the FSRs.

## 5.3 Ethical considerations

As our project makes use of RF signal transmission and reception, we must abide by FCC regulations. This may result in problems such as jamming signals, which can be illegal.

In addition, Since we will very likely be dealing with copyrighted music, we will make sure that what we do comply with the Digital Millennium Copyright Act (DMCA).

Finally, to comply with IEEE Code of Ethics #1, “to strive to comply with...sustainable development practices” [9]. it is our best interest to use materials that are sustainably sourced or recycled

## 5.4 Further work

Our original goal of this project to replicate most of the features of a piano. This includes extended range of keyboard, piano like sound, piano like amplitude modification.

Currently, the design does not have the virtual keyboard. With visualized virtual keyboard, users will be able to play extended range of notes. Instead of using seperate camera module, the virtual keyboard could also be made using webcams on laptops or computers. As the webcam detects the location of our finger, we can use frequency library as we did for 10 fingers, to play the corresponding note. Adding this functionality will increase the design’s utility by great amount.

Moreover, the current design does not give a piano-like touch to users. Change in pressure does modify the amplitude however, the only way of understanding how much pressure one exerted on the key is by listening to sound. Use of springs, or electromagnetic force with varying current may create feedback of the input pressure and provide more piano-like touch to users.

# References


[1] "Digi XBee/RF Solutions," *XBee/RF Wireless Solutions & Radio Modules - Digi International*. [Online]. Available: https://www.digi.com/products/xbee-rf-solutions. [Accessed: 02-May-2018].

[2] "XCTU," XCTU - Next Gen Configuration Platform for XBee/RF Solutions - Digi International. [Online]. Available: https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu. [Accessed: 02-May-2018].

[3] Intel Corporation, "Cyclone IV FPGA Device Family Overview, Cyclone IV Device Handbook," *Intel FPGA*, Mar-2016. [Online]. Available: https://www.altera.com/en_US/pdfs/literature/hb/cyclone-iv/cyiv-51001.pdf. [Accessed: 02-May-2018].

[4] "UART, Serial Port, RS-232 Interface," *Nandland*. [Online]. Available: https://www.nandland.com/vhdl/modules/module-uart-serial-port-rs232.html. [Accessed: 02-May-2018].

[5] "Pitch (music)," *Wikipedia*, 25-Apr-2018. [Online]. Available: https://simple.wikipedia.org/wiki/Pitch_(music). [Accessed: 02-May-2018].

[6] Texas Instruments, "LM3150 Wide-VIN Synchronous Buck Controller", LM3150 datasheet, Sept. 2008 [Revised Sept. 2015].

[7] Texas Instruments, "LM317 3-Terminal Adjustable Regulator", LM317 datasheet, Sept. 1997 [Revised Sept. 2016].

[8] FSR Integration Guide [PDF]. (n.d.). Interlink Electronics. Accessed May 01,2018. https://www.sparkfun.com/datasheets/Sensors/Pressure/fsrguide.pdf

[9] Institute of Electrical and Electronics Engineers, Inc. Code of Ethics IEEE, http://www.ieee.org/. Retrieved at Feb 8, 2018, from the website temoa : Open Educational Resources (OER) Portal at http://www.temoa.info/node/23284

[10] "Why MIDI Matters." The MIDI Association. Accessed February 07, 2018. https://www.midi.org/articles/why-midi-matters.

[11] Altera "Cyclone IV Device Datasheet", Cyclone IV Device Datasheet, March 2016

# Appendix A Requirement and Verification Tables

Table A.1. Cyclone IV FPGA Requirements and Verifications

| Requirement | Verification |
|---|---|
| Should be able to process data captured from OV7670 camera sensor in the image buffer at a rate of at least 30 frames per second (222 Mbps). | Cyclone IV internal clock frequency is20 MHz and should be able to process data in parallel for our design. This maximum frequency can be verified on the Quartus timing analysis report. |
| Should be able to store image data captured in a frame buffer and relay it to a VGA controller at 30fps. | The Cyclone device will be interfaced with an off-chip 128 Mb SDRAM with a max clock frequency of 127 MHz. This should be adequate to store image data. The transfer speed on the SDRAM can also be verified through Quartus timing analysis. |
| Should be able to configure the FPGA SRAM logic elements in JTAG configuration with the host PC. | We can verify proper configuration by analyzing the uploaded bitstream on Quartus. |
| Should be able to receive pressure sensor data from the XBee receiver at a rate of 16 | Verify communication with XBee controller on the FPGA is working as specified in the |

| | |
|---|---|
| Kbps for required resolution. | datasheet through debugging tools in Quartus. |
| Waveform generator must be able to provide 16 bit digital audio samples generated at a PWM frequency to codec to output the required analog waveform. | The correct functionality can be verified by testing on a line out speaker or through debugging tools in Quartus. |

Table A.2. Force Sensitive Resistor Requirement and Verification

| Requirement | Verification |
|---|---|
| FSR is connected to reference resistor Rref. In order to map the voltage range we want to have, we first need to check the varying resistance of FSR. | Ohm-meter was used to to check varying resistance of FSR from ~0 Ohms to 1.2 MOhms. However, from 500K ohms to 1.2 M Ohms, change was so quick that 1.2 M Ohms is not ideal to be used as maximum resistance. |
| XBee module has range of analog input that won't be cut off. The Maximum is nearly the VCC+0.3V of the XBEE module which is going to be 3.6V in our design. Therefore, resulting voltage output from the voltage divider should not exceed 3.6V for proper mapping of voltages. | Input to the voltage divider circuit is 3.3V, identical to reference voltage input of Xbee. As the Voltage divider cannot create output greater than input, the analog signal to Xbee will always be smaller than 3.3V |
| Force Sensitive Resistor Module is part of glove module and it is required to be mobile. In that sense, the entire module needs to be powered by battery. The battery should be able to supply 3.3V to Xbee module and FSR and finger reflectors. We use 9V Batteries. | 9V battery output is checked using oscilloscope before plugging it into glove module. |
| All of FSRs needs to have same voltage and approximately 3.3V. Otherwise, the pressure data may not be reliable. Moreover, maximum output of all FSR should also be close 3.3V. | Used 3.3V voltage regulator circuit using LM317 chip. The output voltage of regulator was 3.302, read by oscilloscope. The voltages at all ADCs were also nearly 3.3 when maximum pressure was applied. |
| For proper voltage mapping and greater sensitivity, we need to have linear response with broad range of voltage outputs, and any nonlinearity should be removed. | Any non-linear region was removed by setting the threshold after non-linear region. Used 20K Ohms resistors to have FSRs perform at best resolution. |

Table A.3 Voltage Regulation Unit Requirements and Verifications

| Requirement | Verification |
|---|---|
| LM317 shall convert 12V and 9V battery input into 3.3 V for main module and glove module.(table 3) | The output voltage of LM317 voltage regulation unit at output node read as 3.32V +/-1% using oscilloscope. |
| FPGA requires 4 different values of input voltages: 1.2V, 1.8V, 2.5, and 3.3 V. Using LM3150, and LP38692, a voltage regulator circuit should provide 4 above voltages.(table 4.) | LM3150 creates two different voltages 1.2 and 3.3V. The voltage value read at output node was in range of less than 2% error rating. LP38692 converts 3.3V into 1.7 and 2.5V. Here as well, the error rating was less 2%. (table 4.) |
| LM3150's minimum input voltage is 6V, and maximum input voltage is 42V. Any voltage input to LM3150 should be in this range. | We are using 12V external power supply. 12V is within the range. |
| LM317's minimum input voltage is 1.25V, and maximum input voltage is 37V. Any voltage input to LM317 should be in this range. | We are using 9V Battery. Batteries that we used were proven that they provide 9V using oscilloscope. |
| LP38692's minimum input voltage is 2.7V, and maximum input voltage is 10V. Any voltage input to LP38692 should be in this range. | LP38692's input voltage is the 3.3V output from the LM3150. |

Table A.4 WM8731 Audio Codec Requirements and Verifications

| Requirement | Verification |
|---|---|
| Should be able to receive digital samples at a modulated frequency from the FPGA waveform generator and produce a corresponding analog signal. | This can be verified separately from the FPGA by sending 16 bit samples at different frequencies for different waveforms and verifying the analog output through a speaker. |
| Should be able to process input digital samples at a high frequency required for frequency response of piano notes. | Sampling rates from 8 kHz to 96kHz are accepted by the codec and can be verified similar to above by testing analog output. |

Table x. XBee RF Module Requirements and Verifications

| Requirement | Verification |
|---|---|
| Transmit data within at least 2 meters reliably. | 1. Set up the transmitter and receiver with the transmitter connected to the FSRs, and receiver connected to the microcontroller and hooked up to the FPGA.<br>2. Move the transmitter around the receiver within a 2m radius. Check the data reading through the FPGA. |
| Convert the analog data from the pressure sensors (SEN-09375) to digital using on-board ADCs. | 1. Same as step 1 from above. Make sure the transmitter and receiver are working properly.<br>2. Press the FSRs, and make sure the data read on the FPGA is correct. |
| Transmit data at 80 Kbps for the processing of pressure sensor readings | 1. Connect both of the transmitting and receiving XBee modules to Arduinos, which will be connected to a PC.<br>2. Use the XCTU software by Digi, send payloads of predetermined size through the Xbee network, and calculate the performance. |

# Appendix B Audio Codec Configurations with Description

**Table B.1 WM8731 Audio Codec Configurations with Description**

| Register Name | Register Value | Description |
|---|---|---|
| R0: Left Line In<br>R1: Right Line In | 010010111 | Simultaneous Load, Enable Line Input Mute, Input Volume at 0dB |
| R2: Left Headphone Out<br>R3: Right Headphone Out | 001111001 | Disable Simultaneous Load, Disable Zero Cross, Output Volume 6dB |
| Analogue Audio Path Control | 000001010 | Sidetone Attenuation -6dB, Disable Sidetone, Don't select DAC, Enable Bypass |
| Digital Audio Path Control | 000001000 | Clear dc offset, DAC softmute, De-emphasis control, Enable High Pass Filter |
| Power Down Control | 000000000 | Disable Power-down |
| Digital Audio Interface Format | 000001010 | I2S format for audio data, 16 bits audio data bit length |
| Sampling Control | 000000000 | Disable clock out divider, Normal sampling mode |
| Active Control | 000000000 | Activate interface |
| Reset Register | 000000000 | No reset |