Simple Electronic Guitar

DSP Lab Fall 2020 Final Project Report

Names	Jihang Xie	Dajr Alfred
Student IDs	N10907771	N10249121
NetIDs	jx1187	dva240

1. Introduction

The aim of our project is to design an electronic guitar prototype. This will be accomplished by utilizing a Raspberry Pi module and MPR121 capacitive touch sensors to simulate the fretting and strumming of a conventional guitar. Implementation of the Karplus Strong Algorithm on the Raspberry Pi will be used to simulate guitar notes in real-time.

1.1 Basic information on a guitar

The guitar is a fretted musical instrument that usually has six strings[1]. It is typically played with both hands by strumming or plucking the strings with either a guitar pick or the fingers/fingernails of one hand, while simultaneously fretting (pressing the strings against the frets) with the fingers of the other hand. The sound of the vibrating strings is projected either acoustically, by means of the hollow chamber of the guitar (for an acoustic guitar), or through an electrical amplifier and a speaker. Almost all guitars have frets, which are metal strips (usually nickel alloy or stainless steel) embedded along the fretboard and located at exact points that divide the scale length in accordance with a specific mathematical formula. The ratio of the spacing of two consecutive frets is $\sqrt[12]{2}$ (twelfth root of two). In practice, luthiers determine fret positions using the constant 17.817—an approximation to $1/\left(1-1/\sqrt[12]{2}\right)$. If the n_{th} fret is a distance x from the bridge, then the distance from the (n+1)th fret to the bridge is x-(x/17.817)[2] Guitar tunings assign pitches to the open strings of guitars, including acoustic guitars, electric guitars, and classical guitars. Tunings are described by the particular pitches denoted by notes in Western music. By convention, the notes are ordered from lowest-pitched string (i.e., the deepest bass note) to highest-pitched (thickest string to thinnest).[3,4] Standard tuning defines the string pitches as E, A, D, G, B, and E, from lowest (low E2) to highest (high E4),as shown in Figure 1[5]. Standard tuning is used by most guitarists, and frequently used tunings can be understood as variations on standard tuning.

1.2 Raspberry PI and MPR121

MPR121 capacitive touch sensors were selected to receive the touch inputs from the user, which simulate the fretting and strumming of a guitar. The signals from the MPR121 module are transmitted to the Raspberry Pi where they are processed to determine which notes are being played.

1.2.1 MPR121

The MPR121 is a Capacitive Touch Sensor module which has 12 input pins[8]. The MPR121 only supports Serial Communication via I2C, which allows for communication with nearly any microcontroller. The device address is selected as one of 4 possible addresses using the ADDR pin, for a total of 48 capacitive touch inputs on one I2C 2-wire bus. Using this module provides a much simpler alternative to performing capacitive sensing with analog inputs: it handles all the filtering internally and can be configured for more/less sensitivity [8]. These devices utilize the I2C interface to communicate with the Raspberry Pi microprocessor via it's SCL (Serial Clock) and SDA(Serial Data) lines. In this project, 4 MPR121 modules were used to simulate 48 notes of an acoustic guitar. Each input of the MPR121 was connected to a ⅛" wide Pyralux pad which is a type of conductive copper foil. When this conductive foil is touched, an input is triggered and the note specific to that pad is played. The image of MPR121 is shown as Figure 2. The definitions of the pins of MPR121 are shown in following:

	0	**I**	**II**	**III**	**IV**
	Open	**1st Fret (index)**	**2nd Fret (middle)**	**3rd Fret (ring)**	**4th Fret (little)**
6th string	E_2	F_2	$F\sharp_2/G\flat_2$	G_2	$G\sharp_2/A\flat_2$
5th string	A_2	$A\sharp_2/B\flat_2$	B_2	C_3	$C\sharp_3/D\flat_3$
4th string	D_3	$D\sharp_3/E\flat_3$	E_3	F_3	$F\sharp_3/G\flat_3$
3rd string	G_3	$G\sharp_3/A\flat_3$	A_3	A♯3/B♭3	B_3
2nd string	B_3	C_4	$C\sharp_4/D\flat_4$	D_4	$D\sharp_4/E\flat_4$
1st string	E_4	F_4	$F\sharp_4/G\flat_4$	G_4	$G\sharp_4/A\flat_4$

Figute 1: ***Chromatic Note Progression***

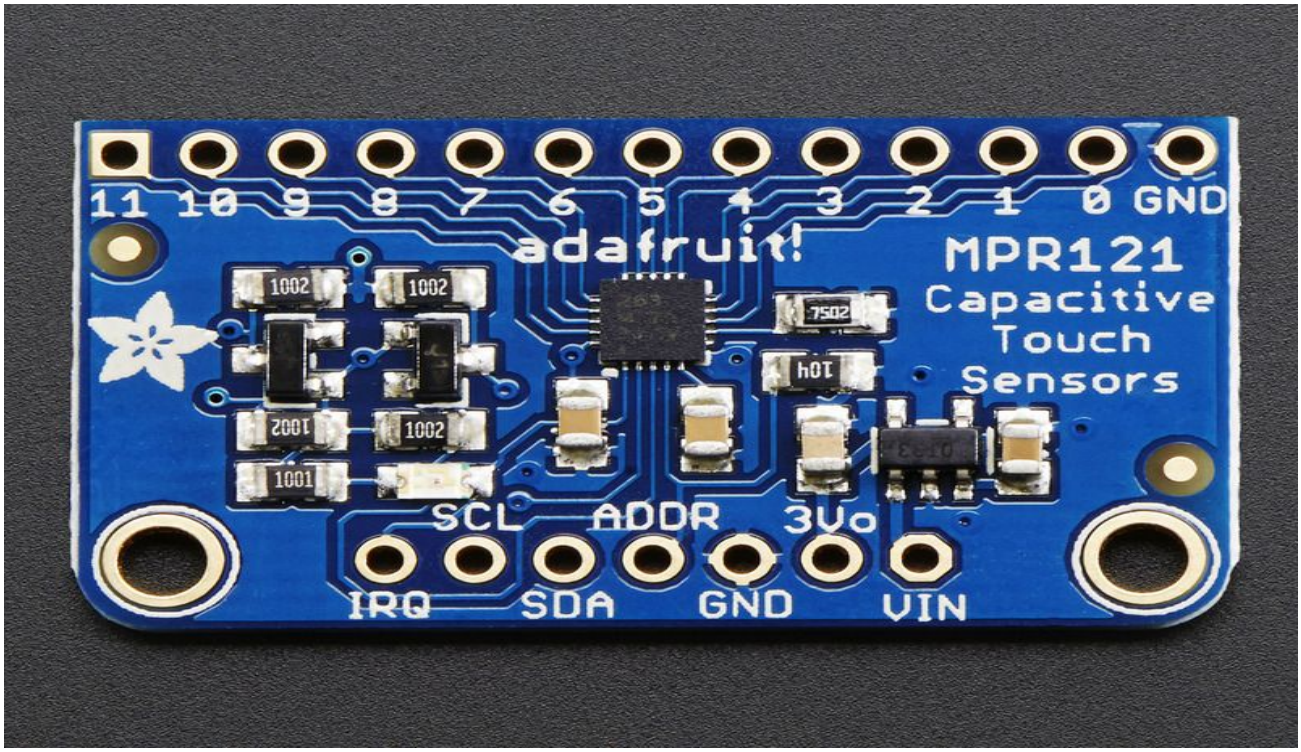


Figure2. MPR121

Power Pins [10]: The sensor on the breakout requires 3V power. Since most users have 5V microcontrollers like Arduino, a 3.3V regulator was included on the board. Its ultra-low dropout allows for input power from 3.3V-5V with no issues.

- Vin - this is the power pin. Since the chip uses 3 VDC, a voltage regulator was included on board that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V
- 3Vo - this is the 3.3V output from the voltage regulator, it allows for a maximum current draw of up to 100mA.
- GND - Common ground for power and logic

I2C Pins [10]:

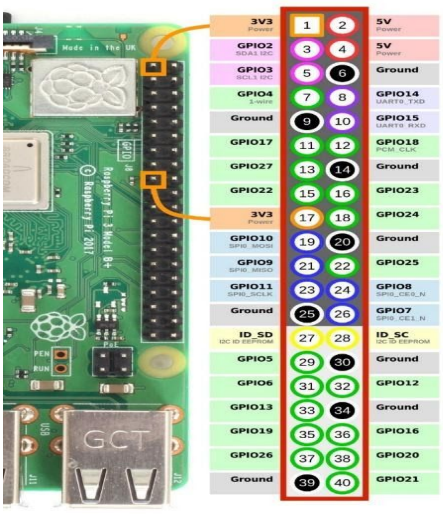
- SCL -- I2C clock pin, connect to your microcontroller's I2C clock line.
- SDA -- I2C data pin, connect to your microcontroller's I2C data line.

IRQ and ADDR Pins [10]:

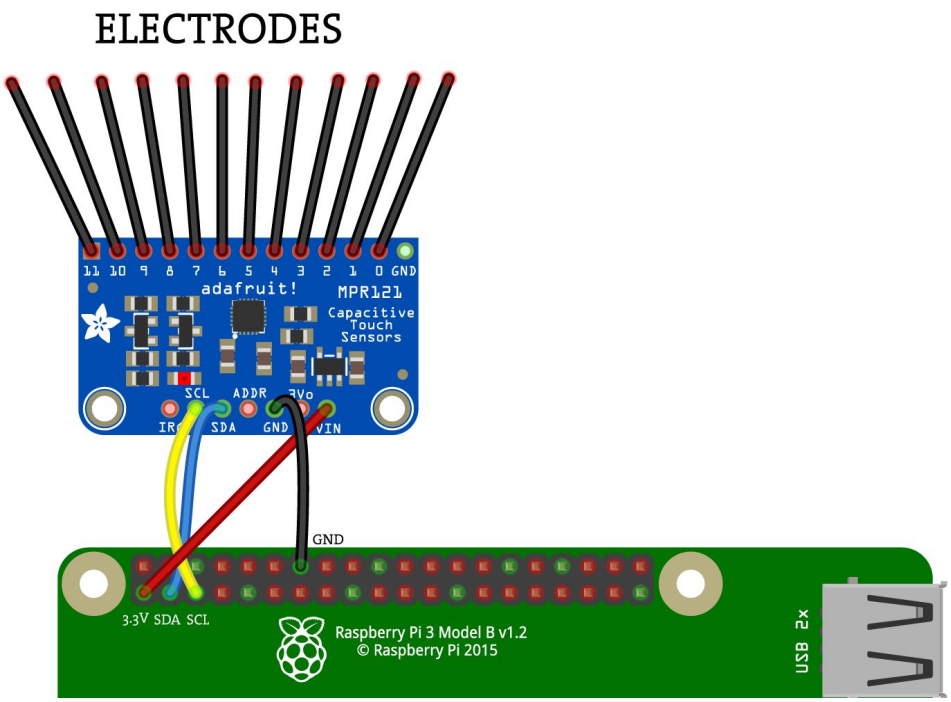
- ADDR is the I2C address select pin. By default this is pulled down to ground with a 100K resistor, for an I2C address of 0x5A. The user can also connect it to the 3Vo pin for an address of 0x5B, the SDA pin for 0x5C or SCL for address 0x5D;
- IRQ is the Interrupt Request signal pin. It is pulled up to 3.3V on the breakout and when the sensor chip detects a change in the touch sense switches, the pin goes to 0V until the data is read over I2C.

1.2.2 Raspberry Pi

The Raspberry Pi Model 3B was released in February 2016 with a 1.2 GHz 64-bit quad core processor, on-board 802.11n Wi-Fi, Bluetooth and USB boot capabilities.[6] On Pi Day 2018, the Raspberry Pi Model 3B+ was launched with a faster 1.4 GHz processor and a three-times faster gigabit Ethernet (throughput limited to ca. 300 Mbit/s by the internal USB 2.0 connection) and 2.4 / 5 GHz dual-band 802.11ac Wi-Fi (100 Mbit/s). Plus, Raspberry Pi 3 Model B+ also has extended 40-pin GPIO header [6], as shown in Figure 3. The Raspberry Pi runs a custom version of Linux known as Raspian. This allowed for the use of the Python programming language to program the Raspberry Pi Model 3B+ for real time analysis of inputs from MPR121 and the output of the corresponding notes in real time.



(a) The GPIO pins of on Raspberry PI 3 Model B+[7]



(b) Wiring Diagram of Raspberry Pi and MPR121 Module

Figure3 The GPIO pins of on Raspberry PI 3 Model B+ [7] and Wiring Diagram of Raspberry Pi and MPR121 Module

To facilitate this real-time application, the Karplus-Strong (KS) algorithm [7] was used to simulate an acoustic guitar. In 1983, the Karplus-Strong (KS) plucked-string algorithm was published. This paper presented a relatively simple, efficient model which is based on the earlier work by McIntyre and Woodhouse (1960). In the Karplus-Strong paper, the plucked string was modelled as having a discrete output y(n) produced by a discrete input x(n) plus the two-point average of previous versions of y(n), delayed by N samples. Thus y(n) = x(n) + ½ y(n-N) + ½ y(n-N-1). This could be modelled as two filters in series with one providing an N sample delay and the other being a two-point averaging filter. These two filters have a total phase delay of (N + ½) samples which corresponds to a fundamental or pitch frequency, f0 = Fs/ (N + ½) in Hertz.

2. Physical Model

2.1 Guitar Model

The physical model of the guitar was modelled in SolidWorks. It was designed to be easily assembled after laser-cutting using ⅛” thick acrylic sheets. The individual parts are held together using M3 screws and stands as well as superglue. The inputs are mounted on individual separable pieces for this initial prototype but will be replaced with a single PCB in future revisions. The guitar assembly features mounting points for the MPR121 modules, the Raspberry Pi as well as a 7” touchscreen display unit.

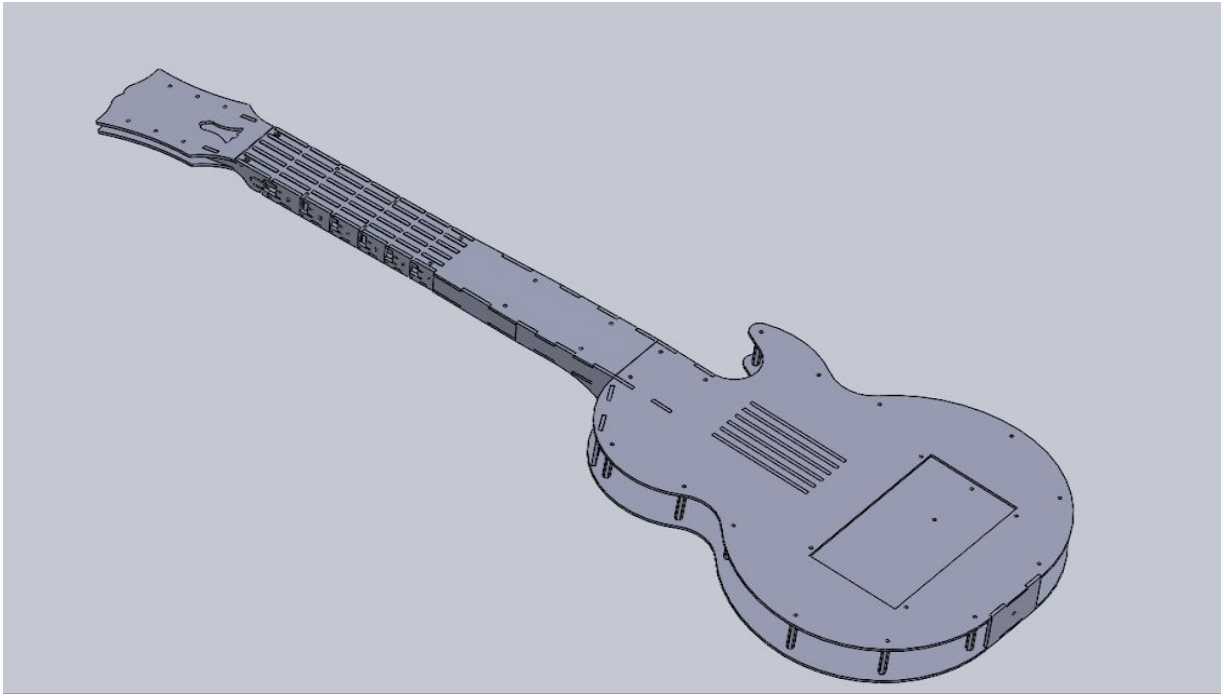


Figure 4: Assembled Guitar Model

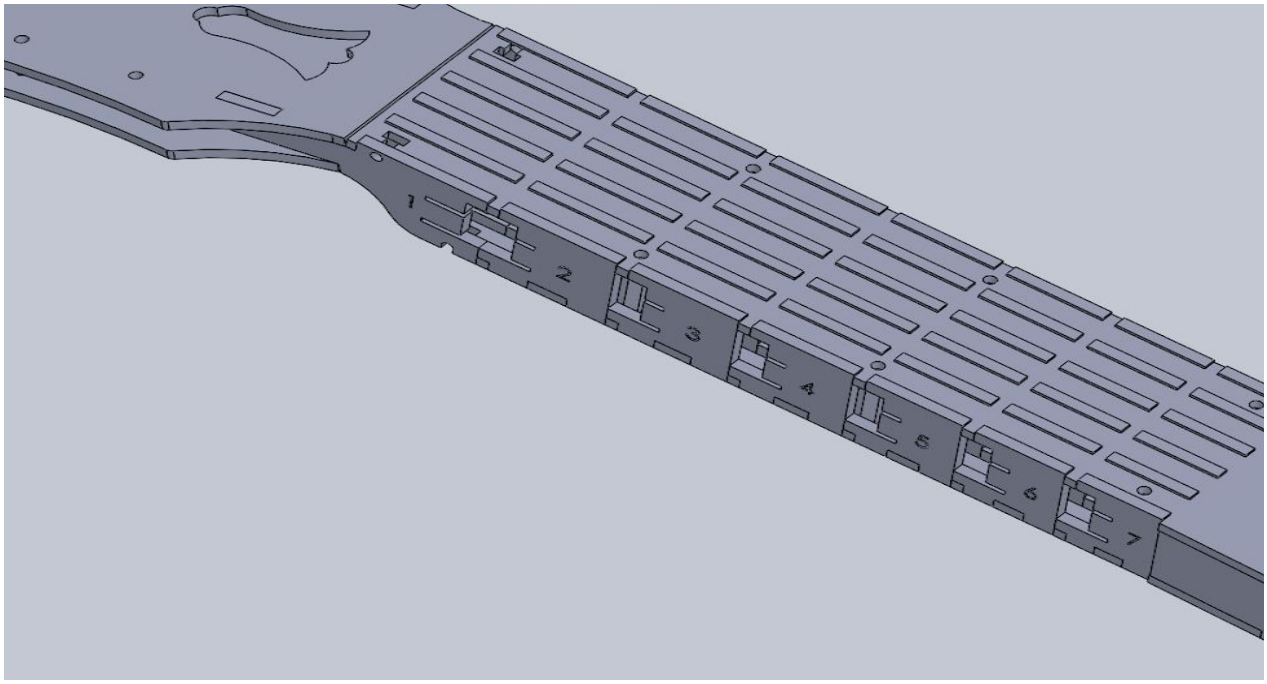


Figure 5: Individual Input Pieces (Labelled 1-7)

Figure 6: Acrylic Sheets - Material Used to Build Guitar

Figure 7: Laser-cut & Partially assembled Physical Model

Figures 4 to 7 show both virtual and physical components of the project. The physical model was able to be partially assembled by the project deadline. Sadly, due to the fact that the team member with hardware expertise was located outside of the US, the model was not able to be completely assembled. Therefore, completion of the guitar assembly will be addressed as part of future work on this project. The software required to play the guitar, however, was successfully completed and is available on the project GitHub repository (featured below).

2.2 Circuit Connection

The circuit connections of the hardware are shown in Figure 8:

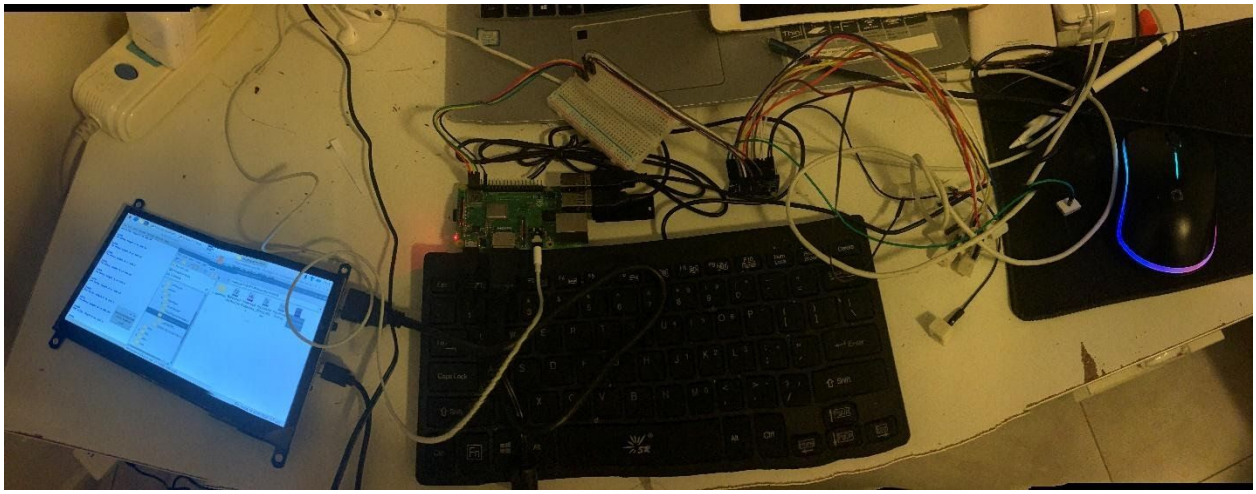


Figure 8: Connections of the hardware

The 7” display screen is connected to the Raspberry Pi through the full-size HDMI port. The keyboard and the mouse are connected through the USB 2.0 ports. The earpods, used for audio output due to budgetary constraints, were connected through the D/A converter jack. Finally, the MPR121 was connected through the GPIO pins. Using Figure 2 as a reference, it is evident that the MPR121’s Vin pin connects the GPIO pin1, MPR121’s GND pin connects to GPIO pin 9, MPR121’s SDA pin connects to GPIO pin 3 and MPR121’s SCL pin connects to GPIO pin 5. Only a single MPR121 module was used to demonstrate the guitar sound effects of this project. Additional MPR121s can be connected to the I2C bus after first connecting the MPR121’s ADDR pin to one of the MPR121 pins specified in **section 1.2.1** to change the address of MPR121. The MPR121s can then be connected in parallel to Raspberry Pi GPIO pins.

3. Software

The software is divided into three parts: the main program— "ElectronicGuitaTest1.py", Karplus-Strong Algorithm function— "my_Karplus_Strong_Alg.py", and Vibrato effect function—"my_vibrato_func.py"[12].

3.1 Karplus-Strong Algorithm function

The implementation of the Karplus-Strong algorithm can be viewed in our python code. The purpose of this function is to map the input pins of the MPR121 to corresponding notes, i.e. frequencies, as shown in Table 1.

Table 1 The relationship between input pin and notes (octave 3)

Pin 0	C(**130.81Hz**)
Pin 1	C♯/D♭(**138.59Hz**)
Pin 2	**D(146.83Hz)**
Pin 3	**D♯/E♭(155.56Hz)**
Pin 4	**E(164.81Hz)**
Pin 5	F(**174.61Hz**)
Pin 6	**F♯/G♭(185.00Hz)**
Pin 7	G(**196.00Hz**)
Pin 8	**G♯/A♭(207.65Hz)**
Pin 9	A(**220.00Hz**)
Pin 10	**A♯/B♭(233.08Hz)**

Therefore, once the fundamental frequency is obtained, (Here, the notes are the fundamental frequencies), the K-S algorithm can compute the delay length and synthesize the Guitar sound [7].

3.2 Vibrato effect function

The vibrato effect can also be optionally implemented via this Python script.[9] The MPR121's input pin 11 was used as the enable trigger of the vibrato effect. When pin 11 is touched, the vibrato effect is added to the output notes.

3.3 the Main Program

The main program implements the synthetic guitar sound effect through analysis of inputs from the user. This is based on the basic function of MPR121. In order to use the MPR121, the latest version of Adafruit CircuitPython and Adafruit CircuitPython MPR121 library [11] must be installed. The details of installation can be viewed in [11].

Once the libraries are installed on the Raspberry Pi, Python can drive the MPR121 and read the input information from these sensors. Combining the formerly introduced scripts, the Raspberry Pi can produce synthetic guitar sounds. The details can be viewed in the code linked below.

3.4 Project in GitHub

All the code written for this project can be found on the project GitHub which can be accessed via the following link and QR-Code.

GitHub Link: https://github.com/dajralfred/Guitar-Simulation [12]

Figure 9: ***QR-Code with Project GitHub Embedded***

4. References

[1] https://en.wikipedia.org/wiki/Guitar#Frets

[2] *Mottola, R.M.* "Lutherie Info—Calculating Fret Positions"

[3] https://en.wikipedia.org/wiki/Guitar_tunings

[4] Brennan, Maureen (2008). "Linen lites - drThe drones of swedish folk music". Dirty Linen: 15.

[5]https://en.wikipedia.org/wiki/Guitar_tunings#/media/File:Standard_diagonal_shifting_of_C_major_chord.png

[6] https://en.wikipedia.org/wiki/Raspberry_Pi#Generations

[7] Karplus, Kevin, and Alex Strong. "Digital Synthesis of Plucked-String and Drum Timbres." Computer Music Journal, vol. 7, no. 2, 1983, p. 43., doi:10.2307/3680062.

[8] https://www.adafruit.com/product/1982

[9] Joshua D. Reiss author. Andrew P. McPherson author. Chapter 2 of Audio Effects: Theory, Implementation and Application[M], Boca Raton, FL : CRC Press 2015: pp 21-58

[10] The pinouts of MPR121: https://learn.adafruit.com/adafruit-mpr121-12-key-capacitive-touch-sensor-breakout-tutorial/pinouts

[11] Python & CircuitPython of MPR121: https://learn.adafruit.com/adafruit-mpr121-12-key-capacitive-touch-sensor-breakout-tutorial/python-circuitpython

[12] Project GotHub Repository: https://github.com/dajralfred/Guitar-Simulation