Pseudo-Theremin

This is a guide to make a DIY theremin.

The Pseudo-Theremin is a DIY version of the Leo Teremin's Theremin. It uses Time of flight sensors to measure the distance of the user's hands from the instrument. The right hand controls the pitch and the left hand controls the amplitude.

Note: We will be writing code in Supercolider & Arduino. A little bit of programming experience will help but feel free to copy paste the code from the software folder

List of components
1. Raspberry pi 0 two w
2. SD card (16 gb or above)
3. ESP32 Dev Kit Module
4. Female to female jumper wires
5. VL53L0X time of flight sensors x 2
5. OTG cable/converter
6. USB A to micro USB cable x 2
7. Computer 8. Heat Sink x 2

On your computer download the raspberry pi imager from the official raspberry pi website.

https://www.raspberrypi.com/software/

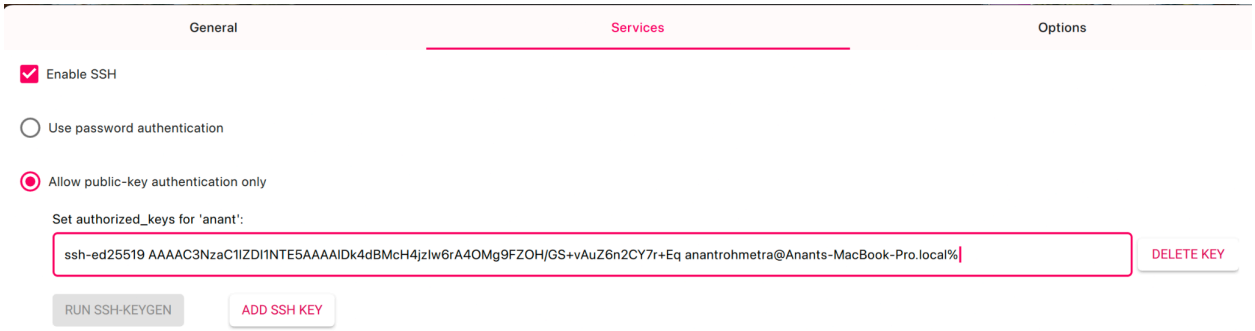
Select the Operating system of your computer. Now install the imager on your laptop. Connect an SD card to your laptop. Once your laptop reads the SD card open the Raspberry pi imager.

Now we will put an Operating System on the SD card and later insert the SD card on the Raspberry pi. The SD card will help us run the operating system on the raspberry pi. We will also put the initial configuration for the raspberry pi on the SD card.

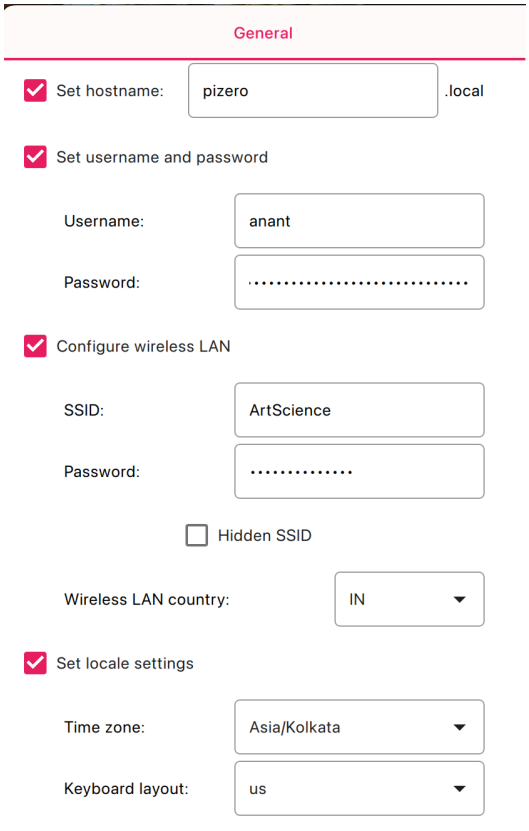
Open the imager and choose the following options

Choose Device : Raspberry Pi zero 2 w
Operating System : Raspberry Pi OS Lite (64 bit)
Choose Storage : "Select your SD Card"

Now in the settings section please put the information as below:


General
Services
Options
Enable SSH
Use password authentication
Allow public-key authentication only
Set authorized_keys for 'anant':
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDk4dBMcH4jzIw6rA4OMg9FZOH/GS+vAuZ6n2CY7r+Eq anantrohmetra@Anants-MacBook-Pro.local%
DELETE KEY
RUN SSH-KEYGEN
ADD SSH KEY


General
Services
Options
Play sound when finished
Eject media when finished
Enable telemetry


General
Set hostname:
pizero
.local
Set username and password
Username:
anant
Password:
Configure wireless LAN
SSID:
ArtScience
Password:
Hidden SSID
Wireless LAN country:
IN
Set locale settings
Time zone:
Asia/Kolkata
Keyboard layout:
us

Tick set hostname
Fill in: pizero // this makes an alias of the IP address of the pi by the name of pizero.local

Fill username & password

tick Configure Network

SSID : "Put the name of the wifi that your computer is connected to" // This will connect the raspberry pi to the wifi
Password : "Put in password for your wifi" // This will help with authentication.

Please enter your timezone for the pi.

In the services section,

Enable SSH and select public key authorization

In the terminal of your computer run:

cd ~/.ssh/ ls -al cat id_rsa.pub

This will return your computer's SSH key. Copy and paste it in “set authorized key for username”

In the options menu tick Eject media when finished and enable telemetry.

Save the settings and upload on the SD. Once the uploading is finished take out the SD card and insert it in the Raspberry pi. Power up the raspberry pi using a micro USB cable. It will take a minute for the pi to start.

Make sure your computer is connected to the same SSID as the raspberry pi pi.

Open the terminal in your laptop and run

the command has to be as follows ssh “username”@”hostname/IP address of your pi”

ssh anant@pizero.local

It might ask you for a password.

Now you can access the pi using your laptop.

```
anantrohmetra@Anants-MacBook-Pro ~ % ssh anant@pizero.local
Linux pizero 6.12.47+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.12.47-1+rpt1 (2025-09-
16) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Nov 20 20:21:20 2025 from fe80::1ce1:13b4:ffc3:b7ae%wlan0
anant@pizero:~ $
```

NOTE: If you are not able to SSH into the pi run the below commands

This command will list available wifi devices

sudo nmcli device wifi list

See which wifi your computer is connected to and SSH again

sudo nmcli device wifi connect "SSID_NAME" password "YOUR_PASSWORD"

Now is a good time to solder the PCM5122 audio board onto the raspberry pi. You can reference the pin diagrams in the hardware section to make the right connections. There is also a picture of how the boards look once soldered. Once this is done place the heat sink on the raspberry pi 0 side which is exposed. Apply the heat sink as shown in the image Raspberry-pi-Heat-Sink. This will absorb the heat from the Raspberry pi.

After this you can connect the Time of Flight sensors to the ESP 32 Dev Module. Please refer to the ESP32->Sensors-circuit-diagram image in the hardware section and make the connections.

Make the following connections from the ESP32 to the sensors:

- D18/GPO18 > SDA pins on both the Tof sensors
- D23/GPIO23 > SCL pins on both the Tof sensors
- D14/GPIO14 > Xshut pin on sensor 1
- D27/GPIO27 > Xshut pin on sensor 2
- 3v3 > VCC on both the Tof sensors
- GND > GND on both the Tof sensors

Now, connect ESP32 to your laptop using a Micro USB data cable. We will offload the code on the ESP32 to read the sensor data.

Download Arduino IDE for your operating system:

https://www.arduino.cc/en/software/

Now, install the Arduino IDE and run it.

Go to your board manager and download ESP32 by Espressif Systems. Now go to your library manager and download adafruit_VL53L0X.

Now in your menu bar click on arduino, then go to settings. In the Additional Board Manager URLs add the below url.

https://espressif.github.io/arduino-esp32/package_esp32_index.json

Now in your menu bar click on tools. In the dropdown menu, click on boards and select “ESP32 dev module”. After that in the same dropdown menu click on port and select the port for your esp32. Make sure in the dropdown menu that upload speed is selected at 230400hz. Select port similar to "/dev/cu.usbserial-0001"

Now we need to write code in order to read both the VL53L0X sensors. You can refer to the code in the ESP32.ino file for the code. Now you need to build and upload the code using the right arrow button on your arduino IDE ->. Once the build and upload is done you can check the values in serial print.

The input data is read in the below format:

“Distance of first sensor”a”Distance of second sensor”b

102a23b
99a44b
106a50b

We want to print in the above format so that we can read the data in SC and figure out which number is associated with which sensor, that is sensor a or sensor b.

Now you can disconnect the ESP32 from your laptop and connect it to your Raspberry pi. In order to do so take the OTG connector and connect it on the raspberry pi’s data port. Then plug in one end of the Micro USB data cable to the OTG connector and the other end to the ESP board.

Now we will download the kernel modules and overlays manually.

Raspberry pi needs the codec and kernel modules of the PCM5122 board in order to send out audio but Debian (Linux operating system on pi) does not have them. So we will download from the Raspberry Pi kernel tree online.

Clone the below github repo using the command below to your raspberry pi.

sudo apt update sudo apt install git -y

git clone --depth=1 https://github.com/raspberrypi/firmware.git

Assuming you cloned the firmware repo into ~/firmware:

cd ~/firmware/boot/overlays

Check files:

ls iqaudio* hifiberry*

Now copy:

sudo cp iqaudio-dacplus.dtbo /boot/overlays/

4. Add the overlay manually to Debian's /boot/firmware/config.txt

Open the config file so that we can configure the codec, enable I2c and I2s. This will enable the pi to speak to the ESP32 and send out audio from the PCM5122 board

sudo nano /boot/firmware/config.txt

Now in the config file make the following changes. You can also refer to the config.txt file in the repository.

Add the below lines to the config file.

dtparam=i2s=on // add this below the line # Uncomment some or all of these to enable the optional hardware interfaces

dtparam=audio=off // add this under # Enable audio (loads snd_bcm2835)

dtoverlay=iqaudio-dac // add this in the end under # Waveshare PCM5122 / HiFiBerry DAC-compatible setup

enable_uart=1 // add this under enable_uart=1

otg_mode=1 // add this under [cm4]

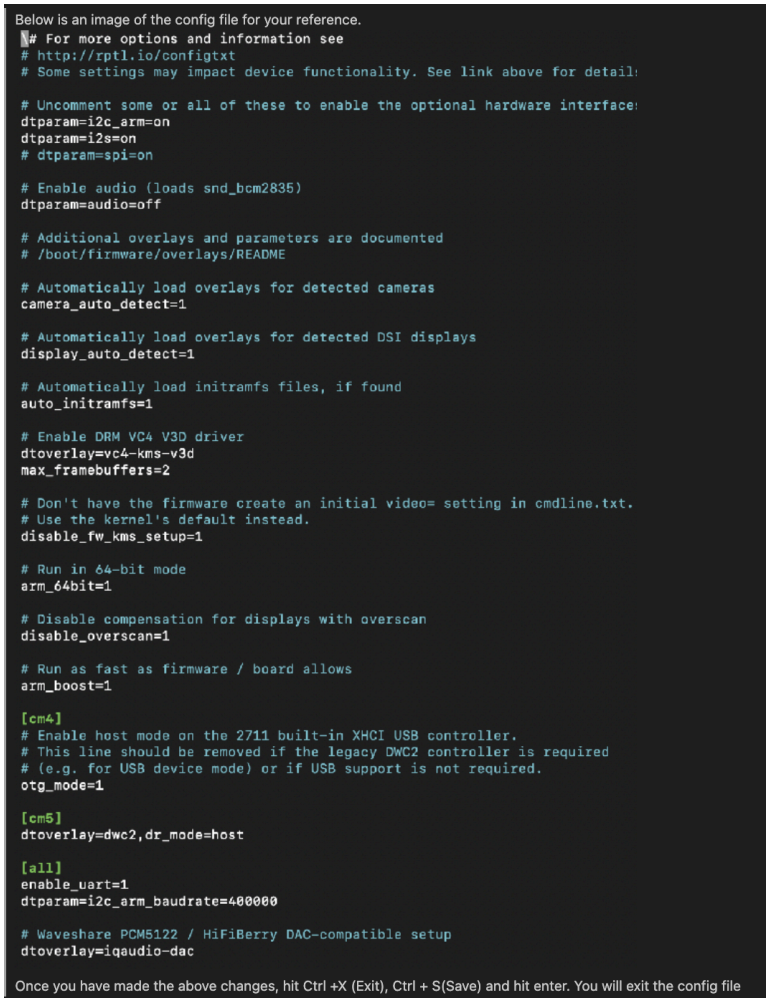
dtoverlay=dwc2,dr_mode=host [cm5]

enable_uart=1 // // add it under [all]

dtparam=i2c_arm_baudrate=400000 // add it under [all]

Below is an image of the config file for your reference.

Below is an image of the config file for your reference.



```
# For more options and information see
# http://rptl.io/configtxt
# Some settings may impact device functionality. See link above for detail

# Uncomment some or all of these to enable the optional hardware interface
dtparam=i2c_arm=on
dtparam=i2s=on
# dtparam=spi=on

# Enable audio (loads snd_bcm2835)
dtparam=audio=off

# Additional overlays and parameters are documented
# /boot/firmware/overlays/README

# Automatically load overlays for detected cameras
camera_auto_detect=1

# Automatically load overlays for detected DSI displays
display_auto_detect=1

# Automatically load initramfs files, if found
auto_initramfs=1

# Enable DRM VC4 V3D driver
dtoverlay=vc4-kms-v3d
max_framebuffers=2

# Don't have the firmware create an initial video= setting in cmdline.txt.
# Use the kernel's default instead.
disable_fw_kms_setup=1

# Run in 64-bit mode
arm_64bit=1

# Disable compensation for displays with overscan
disable_overscan=1

# Run as fast as firmware / board allows
arm_boost=1

[cm4]
# Enable host mode on the 2711 built-in XHCI USB controller.
# This line should be removed if the legacy DWC2 controller is required
# (e.g. for USB device mode) or if USB support is not required.
otg_mode=1

[cm5]
dtoverlay=dwc2,dr_mode=host

[all]
enable_uart=1
dtparam=i2c_arm_baudrate=400000

# Waveshare PCM5122 / HiFiBerry DAC-compatible setup
dtoverlay=iqaudio-dac
```

Once you have made the above changes, hit Ctrl +X (Exit), Ctrl + S(Save) and hit enter. You will exit the config file

Now reboot your computer:

sudo reboot

After reboot the config settings have been enabled.

To verify run:

lsmod | grep iqaudio

Now list out the USB devices connected to the Pi using the below command

ls /dev/ttyUSB*

It should return something like:

/dev/ttyUSB0 // this means the pi is reading ESP32

Print incoming data:

sudo stty -F /dev/ttyUSB0 115200 raw -echo
cat /dev/ttyUSB0

You can stop the printing using Ctrl + C

Now let's download Jack which is an audio routing repository for linux.

sudo apt install jackd2 alsa-utils

Verify if jack has been downloaded properly:

jackd --version

Now let's download supercollider. This is the audio engine which will generate sound.

Update the raspberry pi by running the below command

sudo apt update
sudo apt upgrade -y

Install SuperCollider:

sudo apt install supercollider supercollider-server sc3-plugins

Verify if the installation happened successfully:

sclang -v

Installed Qt headless deps, this will enable you to run supercollider without a GUI from your terminal.

sudo apt install qt5-default qtwayland5

Now run the below command to list out audio output options

aplay -l

It should return something like

**** List of PLAYBACK Hardware Devices ****
card 0: vc4hdmi [vc4-hdmi], device 0: MAI PCM i2s-hifi-0 [MAI PCM i2s-hifi-0]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 1: IQaudIODAC [IQaudIODAC], device 0: IQaudIO DAC HiFi pcm512x-hifi-0 [IQaudIO DAC HiFi pcm512x-hifi-0]

Subdevices: 1/1
Subdevice #0: subdevice #0

For me the correct output will be QuadIODAC which can be written as hw:1,0

Now open a new terminal window and ssh into the pi.

From there you can start jack using the below command

jackd -R -P 70 -d alsa -d hw:1,0 -r 48000 -p 256 -n 2 // hw:1,0 is my device it might be slightly different for you

Run SuperCollider headless:

QT_QPA_PLATFORM=offscreen sclang

This will launch supercollider.

Run the below command to boot the server

s.boot;

run the below command to input the values from the ESP into the pi and store them in the variable ~pot.

~pot = SerialPort.new("/dev/ttyUSB0", 115200);

Please refer to the supercollider code and paste each block on your terminal and then press enter

The input from the ESP32 is being read as ASCII characters. We need to convert the ASCII character into numbers. This is important because the numbers/integers can be assigned to the frequency and amplitude of the theremin sound. This cannot be done with raw ASCII characters

We have also added a smoothing function to avoid drastic or steppy frequency changes. This is because theremin changes in pitch smoothly

For sound design, we have a basic sine wave to begin with. There is slight detune with soft clipping on the sine wave in order to give vacuum tube like effect. We have also added subtle ring modulation to add to the character of the sound

Now if everything works you should have a pseudo theremin to play with. You can connect the audio board to a speaker using the RCA or the minijack outputs on the audio board.

Have fun!