Touch Tone Phone Dial-a-Song

Created by John Park

https://learn.adafruit.com/touch-tone-phone-dial-a-song

Last updated on 2022-02-25 12:02:15 PM EST
# Table of Contents

**Overview**
- Parts 3
- Western Electric 2500DM Telephone 6

**Install CircuitPython**
- CircuitPython Quickstart 6
- Safe Mode 8
- Flash Resetting UF2 10

**Code the Dial-a-Song**
- Text Editor 11
- Download the Project Bundle 11
- Custom Phone Numbers 14
- Custom Songs and Messages 14

**Build the Dial-a-Song Circuit**
- The Circuit 15
- FeatherWing Proto Area Circuit 16
- Keypad Preparation 18
- Keypad Header Pins 20
- Attach the Boards 21

**Assemble the Dial-a-Song**
- Earpiece Connection 23
- Switch Hook Connection 25
- USB Power 27
- Mount the Keypad 30
- Shell 31
- Use the Dial-a-Song 32
Overview

Build a self-contained Dial-a-Song using a Western Electric 2500DM telephone and a Feather RP2040 + mono amplifier. People can dial a number to hear a song or message played through the handset earpiece. The CircuitPython keypad library makes it simple to read the Touch Tone keypad matrix.

It's a fun conversation piece -- especially with its accurate dial tone, touch tones, and other teleco messages -- and perfect for escape rooms or other puzzle games.

I'm not going to tell you how to live your life. However, should you choose to put some [They Might Be Giants](https://adafru.it/YDE) songs on there, you'll be a hero.

Parts

**Adafruit Feather RP2040**
A new chip means a new Feather, and the Raspberry Pi RP2040 is no exception. When we saw this chip we thought "this chip is going to be awesome when we give it the Feather..."

[https://www.adafruit.com/product/4884](https://www.adafruit.com/product/4884)

**FeatherWing Doubler - Prototyping Add-on For All Feather Boards**
This is the FeatherWing Doubler - a prototyping add-on and more for all Feather boards. This is similar to our [https://www.adafruit.com/product/2890](https://www.adafruit.com/product/2890)
Adafruit Mono 2.5W Class D Audio Amplifier - PAM8302
This super small mono amplifier is surprisingly powerful - able to deliver up to 2.5 Watts into 4-8 ohm impedance speakers. Inside the miniature chip is a class D controller, able to...
https://www.adafruit.com/product/2130

Lithium Ion Battery - 3.7v 2000mAh
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...

Terminal block - 4 pin Euro-Style
Connect this to that without soldering using these quick terminal blocks. They allow connecting of two sets of four wires together using just a flat-head screwdriver. These are...
https://www.adafruit.com/product/677

Wire Ferrule Kit - 800 pieces
Crimping's not just for 80s hair metal bands! When your project requires repeated insertion and removal with minimal strain, why not check out the reliable wire...
https://www.adafruit.com/product/5131
USB Type A Plug Breakout Cable with Premium Female Jumpers
If you'd like to connect a USB-capable chip to your USB host, this cable will make the task very simple. There is no converter chip in this cable! It's basically a...
https://www.adafruit.com/product/4448

Premium Silicone Covered Male-Male Jumper Wires - 200mm x 40
These premium male-male jumper wires are handy for making wire harnesses or jumpering between headers on PCBs. They're 200mm (~7.8") long and come loose as a pack of...
https://www.adafruit.com/product/4482

Through-Hole Resistors - 220 ohm 5% 1/4W - Pack of 25
ΩMG! You're not going to be able to resist these handy resistor packs! Well, axially, they do all of the resisting for you! This is a 25 Pack of...
https://www.adafruit.com/product/2780

Through-Hole Resistors - 1.0K ohm 5% 1/4W - Pack of 25
ΩMG! You're not going to be able to resist these handy resistor packs! Well, axially, they do all of the resisting for you! This is a 25 Pack of...
https://www.adafruit.com/product/4294
You'll also need a 2.2µF electrolytic capacitor for the RC noise filter circuit, although in a bind you can use this 10µF one (https://adafru.it/xhB).

**Western Electric 2500DM Telephone**

The classic! Grab a vintage Touch Tone phone out of your attic/basement or hit a flea market or yard sale. In a bind, you can find them on an online auction site for a reasonable amount.

Some older phones will come with a mechanical switching matrix, in which case you'll need to get a "newer" (early-1980s) electronic keypad, such as this one (https://adafru.it/YDH).

I haven't opened up the modern Cortelco ITT 2500, so I can't say if it can be wired the same way the older phones could.

---

**Install CircuitPython**

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the CIRCUITPY drive to iterate.

**CircuitPython Quickstart**

Follow this step-by-step to quickly get CircuitPython running on your board.

Download the latest version of CircuitPython for this board via circuitpython.org

https://adafru.it/R1D
Click the link above to download the latest CircuitPython UF2 file. Save it wherever is convenient for you.

To enter the bootloader, hold down the BOOT/BOOTSEL button (highlighted in red above), and while continuing to hold it (don't let go!), press and release the reset button (highlighted in blue above). Continue to hold the BOOT/BOOTSEL button until the RPI-RP2 drive appears!

If the drive does not appear, release all the buttons, and then repeat the process above.

You can also start with your board unplugged from USB, press and hold the BOOTSEL button (highlighted in red above), continue to hold it while plugging it into USB, and wait for the drive to appear before releasing the button.

A lot of people end up using charge-only USB cables and it is very frustrating! Make sure you have a USB cable you know is good for data sync.
You will see a new disk drive appear called RPI-RP2.

Drag the adafruit_circuitpython_etc.uf2 file to RPI-RP2.

The RPI-RP2 drive will disappear and a new disk drive called CIRCUITPY will appear.

That's it, you're done! :)

Safe Mode

You want to edit your code.py or modify the files on your CIRCUITPY drive, but find that you can't. Perhaps your board has gotten into a state where CIRCUITPY is read-only. You may have turned off the CIRCUITPY drive altogether. Whatever the reason, safe mode can help.
Safe mode in CircuitPython does not run any user code on startup, and disables auto-reload. This means a few things. First, safe mode bypasses any code in boot.py (where you can set CIRCUITPY read-only or turn it off completely). Second, it does not run the code in code.py. And finally, it does not automatically soft-reload when data is written to the CIRCUITPY drive.

Therefore, whatever you may have done to put your board in a non-interactive state, safe mode gives you the opportunity to correct it without losing all of the data on the CIRCUITPY drive.

**Entering Safe Mode in CircuitPython 6.x**

To enter safe mode when using CircuitPython 6.x, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 700ms. On some boards, the onboard status LED (highlighted in green above) will turn solid yellow during this time. If you press reset during that 700ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

**Entering Safe Mode in CircuitPython 7.x**

To enter safe mode when using CircuitPython 7.x, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 1000ms. On some boards, the onboard status LED (highlighted in green above) will
blink yellow during that time. If you press reset during that 1000ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

In Safe Mode

Once you've entered safe mode successfully in CircuitPython 6.x, the LED will pulse yellow.

If you successfully enter safe mode on CircuitPython 7.x, the LED will intermittently blink yellow three times.

If you connect to the serial console, you'll find the following message.

```
Auto-reload is off.
Running in safe mode! Not running saved code.
CircuitPython is in safe mode because you pressed the reset button during boot.
Press again to exit safe mode.
Press any key to enter the REPL. Use CTRL-D to reload.
```

You can now edit the contents of the CIRCUITPY drive. Remember, your code will not run until you press the reset button, or unplug and plug in your board, to get out of safe mode.

Flash Resetting UF2

If your board ever gets into a really weird state and doesn't even show up as a disk drive when installing CircuitPython, try loading this 'nuke' UF2 which will do a 'deep clean' on your Flash Memory. You will lose all the files on the board, but at least you'll be able to revive it! After loading this UF2, follow the steps above to re-install CircuitPython.

Download flash erasing "nuke" UF2

https://adafruit.it/RLE
Code the Dial-a-Song

Text Editor

Adafruit recommends using the Mu editor for editing your CircuitPython code. You can get more info in this guide (https://adafruit.it/ANO).

Alternatively, you can use any text editor that saves simple text files.

Download the Project Bundle

Your project will use a specific set of CircuitPython libraries, folders of .wav file assets, and the code.py file. To get everything you need, click on the Download Project Bundle link below, and uncompress the .zip file.

Drag the contents of the uncompressed bundle directory onto your Feather board's CI RCUITPY drive, replacing any existing files or directories with the same names, and adding any new ones that are necessary.

```python
# SPDX-FileCopyrightText: 2022 John Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT
#
# DTMF keypad phone Dial-a-Song
import time
import random
import board
```
import keypad
from audiocore import WaveFile
from audiopwmio import PWMAudioOut as AudioOut  # for RP2040 etc
import audiomixer

# time.sleep(3)  # let USB settle during development, remove when on battery

km = keypad.KeyMatrix(
    column_pins=(board.A3, board.A2, board.A1,)
    row_pins=(
        board.D24,
        board.D25,
        board.SCK,
        board.MOSI,
        ),
)

numbers = {
    "8675309": "songs/beepbox.wav",
    "6358393": "songs/streetchicken.wav",
    "5551212": "songs/carpenter.wav",
    "7654321": "songs/daisy.wav"
}

ringing = "songs/full_ring.wav"
wrong_number = "songs/blank_number.wav"
dial_tone = "songs/dial_tone_loop.wav"
busy_signal = "songs/busy_loop.wav"

button_tones = [
    "dtmf/tt_1.wav", "dtmf/tt_2.wav", "dtmf/tt_3.wav",
    "dtmf/tt_4.wav", "dtmf/tt_5.wav", "dtmf/tt_6.wav",
    "dtmf/tt_7.wav", "dtmf/tt_8.wav", "dtmf/tt_9.wav",
    "dtmf/tt_star.wav", "dtmf/tt_0.wav", "dtmf/tt_pound.wav"
]

digits_entered = 0  # counter
dialed = []  # list of digits user enters to make one 7 digit number
dialed_str = ""  # stores the phone number string for dictionary comparison

audio = AudioOut(board.TX)  # PWM out pin
mixer = audiomixer.Mixer(
    voice_count=4,
    sample_rate=22050,
    channel_count=1,
    bits_per_sample=16,
    samples_signed=True,
)

audio.play(mixer)
mixer.voice[0].level = 1.0  # dial tone voice
mixer.voice[1].level = 1.0  # touch tone voice
mixer.voice[2].level = 0.0  # song/message voice
mixer.voice[3].level = 0.0  # busy signal

wave_file0 = open(dial_tone, "rb")
wave0 = WaveFile(wave_file0)
mixer.voice[0].play(wave0, loop=True)  # play dial tone

wave_file2 = open(wrong_number, "rb")
wave2 = WaveFile(wave_file2)

wave_file3 = open(busy_signal, "rb")
wave3 = WaveFile(wave_file3)
mixer.voice[3].play(wave3, loop=True)  # play dial tone

def reset_number():

```python
```
```python
# pylint: disable=global-statement
global digits_entered, dialed, dialed_str

digits_entered = 0
dialed = []
dialed_str = ""
km.events.clear()

while True:
    event = km.events.get()  # check for keypad presses
    if event:
        if event.pressed:
            mixer.voice[0].level = 0.0  # mute the dial tone
            wave_file1 = open(button_tones[event.key_number], "rb")  # play Touch Tone
            mixer.voice[1].play(wave1)
            if event.key_number == 9 or event.key_number == 11:
                # check for special keys
                if event.key_number == 9:
                    # pressed the '*' key
                    reset_number()  # or make some cool new function for this key
                if event.key_number == 11:
                    # pressed the '#' key
                    reset_number()  # or make some cool new function for this key
            else:
                # number keys
                if digits_entered < 7:
                    # adding up to full number
                    # convert event to number printed on the keypad button, append
                    to string
                    if event.key_number < 9:
                        # 1-9 on keypad
                        dialed.append(event.key_number+1)
                    if event.key_number == 10:
                        # the 0 key, ignore '*' and '#'
                        dialed.append(0)
                    dialed_str = "".join(str(n) for n in dialed)
                    digits_entered = digits_entered + 1  # increment counter
                    if digits_entered == 7:
                        # a full number has been entered
                        if not mixer.voice[2].playing:
                            dialed_str = "".join(str(n) for n in dialed)
                            # check if dialed string is one in the directory
                            value = numbers[dialed_str]
                            time.sleep(0.6)
                            wave_file2 = open(ringing, "rb")  # ring before it answers
                            wave2 = WaveFile(wave_file2)
                            mixer.voice[2].level = 1.0
                            mixer.voice[2].play(wave2, loop=True)
                            time.sleep(random.uniform(4.0, 9.5))  # random ring before "answer"
                            wave_file2 = open(value, "rb")  # answered
                            wave2 = WaveFile(wave_file2)
                            mixer.voice[2].level = 1.0
                            mixer.voice[2].play(wave2, loop=True)
                        else:
                            # number is not in directory
                            time.sleep(0.5)
                            weighted_coin_toss = random.randint(0, 4)
                            if weighted_coin_toss < 3:
                                # favor the "not in service" message
                                mixer.voice[2].level = 1.0
                                mixer.voice[2].play(wave2)
                            else:
                                mixer.voice[3].level = 1.0
                                reset_number()
```

©Adafruit Industries
if mixer.voice[2].playing:
    reset_number() # stop #s dialed during message play from
    doing anything

Custom Phone Numbers

Create your own directory of valid numbers and their associated .wav files in the `numbers` dictionary:

```python
numbers = {
    "8675309" : "songs/beepbox.wav",
    "6358393" : "songs/streetchicken.wav",
    "5551212" : "songs/carpeter.wav",
    "7654321" : "songs/daisy.wav"
}
```

Custom Songs and Messages

To create your own song and message files (great for granting/denying entrance to your secret speakeasy) convert your audio files to 16-bit mono WAV files at 22KHz sample rate. [This guide](https://adafru.it/CQM) shows how.

Build the Dial-a-Song Circuit
The Circuit

The circuit above is the breadboarded version of the circuit you'll build on a FeatherWing Doubler. You may want to test it on a breadboard before moving to soldering it together.

The sections of the circuit are:

- Feather RP2040 A1-M0 connected to keypad matrix
- Feather RP2040 TX pin connect to audio RC filter circuit to reduce noise, then to A+ input of the amp breakout
- Amp output connect to phone handset speaker
- Phone switch hook connected to Feather En and GND
- USB cable breakout power and GND to Feather USB and GND
FeatherWing Proto Area Circuit
Transfer the circuit from breadboard to the proto area of the FeatherWing Doubler.

The seven position header will be soldered to the underside of the board so it can be plugged into the back of the matrix keypad later (see below).

Solder two black lead wires with ferrule crimped ends to the Doubler GND.

Solder another wire to En (this will enable/disable the board, essentially powering it off when the handset is in the cradle).

Solder a red wire with ferrule end to the USB pin on the Doubler. This will be connected to USB power for battery charging.
Fit the amp and Feather onto the Doubler.
Keypad Preparation

The keypad assembly is composed of two assemblies -- the DTMF circuit board, and the keypad matrix itself. We don't need the DTMF board, so we'll desolder the ribbon cable and then add header pins to connect to the FeatherWing.

Use a solder sucker and/or desoldering wick to remove the ribbon cable cleanly.
Here you can see how the keypad works -- the plastic buttons press down on the elastomer pads, causing the conductive rubber pills to short the associated row/column traces.

Note, the pcb is single-sided with the exception of the row two and row three bridges routed as wires through the back of the board!
Keypad Header Pins

Solder in the header pins -- the matrix pcb has eight pads, but we only need to connect to the first seven, based on how the matrix is designed, using three columns and four rows.
Attach the Boards
Plug in a LiPo battery to the Feather, then fit it behind the keypad.

Press the Doubler socket in to the first seven pins of the keypad.
Next you'll integrate the Feather board with the rest of the phone.

Assemble the Dial-a-Song

The 2500 has a network block (potted bolus of electronics that make it tick) with all of the wiring we need attached via screw terminals.

You'll disconnect some of these wires to connect the switch hook and receiver earpiece to the Feather.
Earpiece Connection

Locate the wires running to the earpiece from the RJ9 jack. You can unscrew the handset mouthpiece or earpiece to see which wires run to the speaker, in this case, green and white wires.

Unscrew those two wires from the phone's network block and screw them into the terminal block on the amplifier board.
Switch Hook Connection

Use a multimeter to find two connection points on the network block that are opened and closed when the switch hook is engaged.

Run two blue wire leads with ferrule connectors on their ends to these two points and screw them in place. (Fragile bits of plastic snapped off on mine, hence the Kapton tape).

Use the Euro block connector to connect the two blue switch hook wires to the two black En/GND wires.
USB Power
You'll run USB power to the Feather in order to charge the LiPo battery when needed.

Prepare a black jumper wire and a red jumper wire with ferrule ends that can be used in the Euro block on one end and DuPont connector pins on the other (premium silicone jumper wires (https://adafruit.it/NXB) work well for this).

Run the header end of the USB breakout cable through the RJ11 jack, then connect the two power extension wires. You can ignore the USB data lines.

Connect the power and ground lines to their respective leads on the Feather Doubler using the Euro block.
The receiver and USB power are now connected to the Feather, which is plugged into the keypad. You should test the following:

- switch hook works to enable/disable the Feather
- audio comes through the earpiece speaker
- keypad entry is properly registered
- USB charges the battery

You can now start re-assembling the phone.
Mount the Keypad
Secure the keypad in the metal brackets, then tighten the screws.
Shell
Place the shell onto the base, making sure the RJ jacks fit their respective places.

Tighten the two captive screws in the base.
Use the Dial-a-Song

Using the phone is simple! Pick up the receiver, wait for the dial tone, and then call one of the seven digit numbers in the directory.

The phone will ring a randomized number of times and then "pick up", playing the song you wanted!

If you dial a wrong number you'll hear either a busy signal or the "not in service" message.