# Table of Contents

## Overview 3
- Parts

## Install CircuitPython 6
- CircuitPython Quickstart

## Code the Computer Perfection Synth 8
- Text Editor
- Download the Project Bundle
- How it Works
- Libraries
- NeoPixels
- Keypad
- Notes
- I2S Amp
- Synth Setup
- Waveforms
- Low Frequency Oscillator
- Helper Functions
- Main Loop

## Assemble the Computer Perfection Synth 19
- Open the Panel
- Remove PCB
- Desolder Wires
- Remove Microcontroller
- Wiring Diagram
- DIP Header Wiring
- Cap
- Socket Connection
- Wiring Length
- I2S Amp
- Mount Speaker
- NeoPixel Wiring
- LED Placement
- USB Cable
- Metro Placement
- Magnetic USB Cable

## Play the Computer Perfection Synth 38
Overview

When PT discovered this beautiful artifact from 1979, the Computer Perfection electronic memory game, we knew we had to transform it into a sci-fi drone synthesizer!

Luckily, this coincided with Jepler’s work on the CircuitPython synthio library and the release of the Metro M7 -- a perfect confluence of events!

This project takes the Computer Perfection and reuses its buttons and switches to trigger a polyphonic, multi-timbral wavetable synthesizer for all your spacey jam
sessions. It includes ADSR envelopes and LFO modulation for a beautiful, other-worldly sound.

Parts

**Adafruit Metro M7 with AirLift - Featuring NXP iMX RT1011**
Get ready for our fastest Metro ever - the NXP iMX RT1011 microcontroller powers this board with a 500 MHz ARM Cortex M7 processor. There's 8 MB of execute-in-place QSPI for...
https://www.adafruit.com/product/4950

**Ultra Skinny NeoPixel 1515 LED Strip 4mm wide**
We have all sorts of LED strips for a wide range of needs. Chonky strips? We got those!
https://www.adafruit.com/product/4368

**Proto-Screwshield (Wingshield) R3 Kit for Arduino**
The next generation Proto-ScrewShield is a dual-purpose prototyping shield. Not only does it have a large 0.1" grid prototyping area but it also extends the Arduino pins to...
https://www.adafruit.com/product/196
Adafruit I2S 3W Class D Amplifier Breakout - MAX98357A
Listen to this good news - we now have an all in one digital audio amp breakout board that works incredibly well with the https://www.adafruit.com/product/3006

Mono Enclosed Speaker with Plain Wires - 3W 4 Ohm
Listen up! This single 2.8'' x 1.2'' speaker is the perfect addition to any audio project where you need 4 ohm impedance and 3W or less of power. We... https://www.adafruit.com/product/4445

28-position DIP Connector ()
You'll use this to wire the buttons and switches to the Metro M7 using the original DIP socket.

1 x DIY USB Cable Parts - Straight Type C Plug
DIY USB Cable Parts
1 x DIY USB Cable Parts
Straight Micro B Jack
1 x DIY USB or HDMI Cable Parts
20 cm Ribbon Cable
Install CircuitPython

CircuitPython () is a derivative of MicroPython () 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

Click the link above to download the latest CircuitPython UF2 file.

Save it wherever is convenient for you.
Plug your board into your computer, using a known-good data-sync cable, directly, or via an adapter if needed.

Click the reset button once (highlighted in red above), and then click it again when you see the RGB status LED(s) (highlighted in green above) turn purple (approximately half a second later). Sometimes it helps to think of it as a "slow double-click" of the reset button.

On some very old versions of the UF2 bootloader, the status LED turns red instead of purple.

Once successful, you will see the RGB status LED(s) turn green (highlighted in green above). If you see red, try another port, or if you’re using an adapter or hub, try without the hub, or different adapter or hub.

If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!

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 METROM7BOOT.

Drag the adafruit_circuitpython_etc.uf2 file to METROM7BOOT.

The BOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

That's it!

---

**Code the Computer Perfection Synth**

**Text Editor**

Adafruit recommends using the Mu editor for editing your CircuitPython code. You can get more info in [this guide](#).
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, and the code.py file. To get everything you need, click on the Download Project Bundle link below, and uncompress the .zip file.

Connect your computer to the M7 board via a known good USB power+data cable. A new flash drive should show up as CIRCUITPY.

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

```python
import time
import random
import board
import audiobusio
import audiomixer
import synthio
import ulab.numpy as np
import neopixel
import keypad

# NeoPixel setup
num_pixels = 34
pixels = neopixel.NeoPixel(board.D11, num_pixels, brightness=0.7, auto_write=False)
pixels.fill(0x0)
pixels.show()
ten.sleep(0.25)
pix_map = [26, 23, 19, 16, 13, 10, 7, 4, 32, 29]  # map the LEDs to the numbered panel sections 0-9
```

# SPDX-FileCopyrightText: 2023 John Park, Jeff Epler, and Tod Kurt for Adafruit Industries
# SPDX-License-Identifier: MIT
# Computer Perfection Synth
# * 10 numbered buttons play notes
# * SET button to increase LFO rate, long press to decrease LFO rate
# * SCORE button to add lower octave
# * MODE switch changes wavetable set
# * SKILL switch toggles sustain
# * GAME switch must stay in position 1 or it messes with the other switches

©Adafruit Industries
for p in range(len(pix_map)):
    pixels[pix_map[p]] = 0xff0000
    pixels.show()
    time.sleep(0.1)

note_buttons = keypad.Keys(
    value_when_pressed=False,
    pull=True
)

switches = keypad.Keys(
    (board.A1, board.A0),
    value_when_pressed=False,
    pull=True
)

octave = 3  # octave multiplier
note_list = (0, 4, 6, 7, 9, 12, 16, 18, 19, 21)  # Lydian scale

mod_buttons = keypad.Keys(
    (board.A4, board.A3),  # SET and SCORE buttons
    value_when_pressed=False,
    pull=True
)

SAMPLE_RATE = 48000  # clicks @ 36kHz & 48kHz on rp2040
SAMPLE_SIZE = 200
VOLUME = 12000

# Metro M7 pins for the I2S amp:
lck_pin, bck_pin, dat_pin = board.D9, board.D10, board.D12

# synth engine setup
waveform = np.zeros(SAMPLE_SIZE, dtype=np.int16)  # intially all zeros (silence)

amp_env = synthio.Envelope(  # default (0.1, 0.05, 0.2, 1, 0.8)
    attack_time=1.0,
    decay_time=0.05,
    release_time=3.0,
    attack_level=1.0,
    sustain_level=0.8
)

synth = synthio.Synthesizer(sample_rate=SAMPLE_RATE, waveform=waveform, envelope=amp_env)
audio = audiobusio.I2SOut(bit_clock=bck_pin, word_select=lck_pin, data=dat_pin)
mixer = audiomixer.Mixer(voice_count=1, sample_rate=SAMPLE_RATE, channel_count=1,
                       bits_per_sample=16, samples_signed=True, buffer_size=8192)
audio.play(mixer)
mixer.voice[0].level = 0.55
mixer.voice[0].play(synth)

led = neopixel.NeoPixel(board.NEOPIXEL, 1, brightness=0.3)  # on board neopixel

# waveforms setup
wave_sine = np.array(np.sin(np.linspace(0, 2*np.pi, SAMPLE_SIZE, endpoint=False)) * VOLUME, dtype=np.int16)
wave_saw = np.linspace(VOLUME, -VOLUME, num=SAMPLE_SIZE, dtype=np.int16)
wave_weird1 = np.array((198, 2776, 5441, 8031, 10454, 12653, 14609, 16333, 17824, 19130, 20260, 21227, 22043,
                       22721, 23269, 23699, 24019, 24243, 24385, 24461, 18630, -26956, -28048, -29175, -30249, -31227, -32073, -32631, -32359, -31817, -30941, -29663, -27900, -25596, -22591, -18834, -14291, 9016, 3212, 2794, 8624, 13943, 18544, 22353, 25408, 27780, 29553, 30855, 31751, 32315, 32611, 32687, 32593, 32351, 31983, 31491, 30871,
wave_noise = np.array([random.randint(-VOLUME, VOLUME) for i in range(SAMPLE_SIZE)], dtype=np.int16)

# map s range a1-a2 to b1-b2
def map_range(s, a1, a2, b1, b2):
    return b1 + ((s - a1) * (b2 - b1) / (a2 - a1))

# mix between values a and b, works with numpy arrays too, t ranges 0-1
def lerp(a, b, t):
    return (1-t)*a + t*b

waveform[:] = wave_saw
wave_mix = 0.0
lfo_rates = (0.1, 0.5, 0.8, 1.5, 3.0, 6.0, 7.0, 8.0)
lfo_index = 0
lfo1 = synthio.LFO(rate=(lfo_rates[lfo_index]), waveform=wave_sine)  # rate is in Hz
synth.lfos.append(lfo1)
hold = False  # state of note hold
octaves = False

def light_button_pixels(button_number):
    pixels[pix_map[button_number]+1] = 0xFF0000
    pixels[pix_map[button_number]-1] = 0xFF0000
    pixels.show()

def reset_button_pixels(button_number):
    pixels[pix_map[button_number]+1] = 0x000000
    pixels[pix_map[button_number]-1] = 0x000000
    pixels.show()

def clamp(v, low, high):
    return min(max(v, low), high)

print("-Computer Perfection Synth-")

note = None
mod_key = 0
last_mod_button_event_time = 0
waveset = 0

while True:
    # watch for mod buttons to be pressed
    mod_button_event = mod_buttons.events.get()
    if mod_button_event:
        mod_key = mod_button_event.key_number
        if mod_button_event.pressed:
            if mod_key == 0:  # SET switch
                last_mod_button_event_time = time.monotonic()
if mod_key == 1:  # enable octaves
    octaves = True

if mod_button_event.released:
    if last_mod_button_event_time and mod_key == 0:  # short press-release
        increase LFO rate
        lfo_index = clamp(lfo_index+1, 0, len(lfo_rates)-1)
        print(lfo_index)
        lfo_rate = lfo_rates[lfo_index]
        lfo1.rate = lfo_rate
        last_mod_button_event_time = 0
if mod_key == 1:  # disable octaves
    octaves = False
# long press slows the LFO rate
if last_mod_button_event_time != 0 and time.monotonic() -
    last_mod_button_event_time > 1.0:
    last_mod_button_event_time = 0
    lfo_index = clamp(lfo_index-1, 0, len(lfo_rates)-1)
    lfo_rate = lfo_rates[lfo_index]
    lfo1.rate = lfo_rate
# watch for note buttons to be pressed
note_button_event = note_buttons.events.get()
if note_button_event:
    i = note_button_event.key_number
    if note_button_event.pressed:
        if octaves:
            synth.press((note_list[i]+(octave*12), note_list[i]+(octave*12)-12))
        else:
            synth.press((note_list[i]+(octave*12),))
        light_button_pixels(i)
    if note_button_event.released:
        if not hold:
            reset_button_pixels(i)
            synth.release((note_list[i]+(octave*12), note_list[i]+
                (octave*12)-12))
            reset_button_pixels(i)
# watch for switches to be changed
switch_event = switches.events.get()
if switch_event:
    sw = switch_event.key_number
    if switch_event.pressed:
        if sw == 0:  # MODE toggle right
            mixer.voice[0].level = 0.45
            # wave_mix = 0.5
            waveset = 0
        if sw == 1:  # SKILL toggle center
            hold = True
            if sw == 0:  # MODE toggle center
                mixer.voice[0].level = 0.95
                waveset = 1
                if sw == 1:  # SKILL toggle right or left
                    hold = False
                    for r in range(len(note_list)):
                        # turn off all notes
                        # if octaves:
                            synth.release((note_list[r]+(octave*12), note_list[r]+
                                (octave*12)-12))
                        for h in range(len(pix_map)):
                            # turn off held pixels
                            reset_button_pixels(h)
        lfo_val_for_lerp = map_range(lfo1.value, -1, 1, 0, 1)
        if waveset == 0:
            waveform[:] = lerp(wave_sine, wave_weird1, lfo_val_for_lerp)
        else:
            waveform[:] = lerp(wave_saw, wave_noise, lfo_val_for_lerp)
How it Works

Libraries

First, we import the required libraries:

```python
import time
import random
import board
import audiobusio
import audiomixer
import synthio
import ulab.numpy as np
import neopixel
import keypad
```

NeoPixels

Next we initialize the NeoPixel strip with 34 pixels connected to pin D11 on the board.

We also set up a list of the physical pixels that correspond to the game panel's 0-9 locations.

```python
# NeoPixel setup
num_pixels = 34
pixels = neopixel.NeoPixel(board.D11, num_pixels, brightness=0.7, auto_write=False)
pixels.fill(0x0)
pixels.show()
time.sleep(0.25)
pix_map = [26, 23, 19, 16, 13, 10, 7, 4, 32, 29]  # map the LEDs to the numbered panel sections 0-9
for p in range(len(pix_map)):
    pixels[pix_map[p]] = 0xff0000
    pixels.show()
    time.sleep(0.1)
```

Keypad

We'll use the keypad library to read the ten note buttons, two modifier buttons, and two switches.

```python
note_buttons = keypad.Keys(
    (board.D0, board.D1, board.D2, board.D3, board.D4,
    value_when_pressed=False,
    pull=True
)
switches = keypad.Keys(
    (board.A1, board.A0),
    value_when_pressed=False,
    pull=True
)
value_when_pressed=False,
pull=True)

Notes

The synthio library can use MIDI note numbers or frequency to specify a note to play. In this project we'll use MIDI note numbers as they're easier to adjust for interval/scale choices.

This `note_list` contains the ten notes we'll play, and the `octave` variable specifies which octave to play them in.

```python
octave = 3  # octave multiplier
note_list = (0, 4, 6, 7, 9, 12, 16, 18, 19, 21)  # Lydian scale
```

I2S Amp

The I2S amp is set up on three pins of the Metro M7 (this can vary for other boards).

```python
lck_pin, bck_pin, dat_pin = board.D9, board.D10, board.D12
```

Synth Setup

The `synthio` object is set up with the sample rate, sample size, volume, initial waveform, and audiobus/audiomixer objects.

```python
SAMPLE_RATE = 48000  # clicks @ 36kHz & 48kHz on rp2040
SAMPLE_SIZE = 200
VOLUME = 12000

# synth engine setup
waveform = np.zeros(SAMPLE_SIZE, dtype=np.int16)  # initially all zeros (silence)
amp_env = synthio.Envelope(  # default (0.1, 0.05, 0.2, 1, 0.8)
    attack_time=1.0,
    decay_time=0.05,
    release_time=3.0,
    attack_level=1.0,
    sustain_level=0.8)

synth = synthio.Synthesizer(sample_rate=SAMPLE_RATE, waveform=waveform,
envelope=amp_env)
audio = audiobusio.I2SOut(bit_clock=bck_pin, word_select=lck_pin, data=dat_pin)
mixer = audiomixer.Mixer(voice_count=1, sample_rate=SAMPLE_RATE, channel_count=1,
bits_per_sample=16, samples_signed=True, buffer_size=8192)
```
Waveforms

Different waveforms have different harmonics, which is what provides the "character" or timbre of an instrument. We'll use sine, saw, and the weird1 wavetable, and noise waveforms, which can be mixed between using the LFO modulator.

All of these waveforms are defined mathematically, except for the weird1 which is defined with an array of discreet points.

```python
# waveforms setup
wave_sine = np.sin(np.linspace(0, 2*np.pi, SAMPLE_SIZE, endpoint=False)) * VOLUME,
            dtype=np.int16
wave_saw = np.linspace(VOLUME, -VOLUME, num=SAMPLE_SIZE, dtype=np.int16)
wave_weird1 = np.array((198,2776,5441,8031,10454,12653,14609,16333,17824,19130,20260,21227,22243,
                        22721,23699,24019,24243,24385,24461,18630,-26956,-28048,-29175,-30249,
                        -31227,-32073,-32631,-32359,-31817,-30941,-29663,-27900,-25596,-22591,
                        -18834,-14291,-9016,-3212,2794,8624,13943,18544,22353,25408,27780,29553,
                        30855,31751,32315,32687,32593,32351,31983,31491,30871,30097,28895,
                        -28240,-30499,-31343,-31975,-32431,-32697,-32615,-32217,-31525,
                        -30489,-29035,-27090,-24519,-21237,-17178,-12339,-6829,-902,5081,10748,
                        15805,20102,23615,26396,28510,30109,31245,31995,31437,30729,29887,
                        28943,27908,26784,25560,24077,22781,-22027,-22735,-22079,-22471,-22065,
                        -21497,-20773,-19896,-18872,-16361,-14857,-13141,-11206,-9054,-6717,
                        -4259,-1796,522,2548,4167,5339,6079,6445,6503,6319,5949,5449,4847,4183,
                        3480,2756,2028,1304,590,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                        -478,-1168,-1882,-2596,
                        -3336,-4074,-4795,-5487,-6119,-6669,-7095,-7357,-7399,-7157,-6559,-5543,
                        -4876,-2132,), dtype=np.int16)
wave_noise = np.array([random.randint(-VOLUME, VOLUME) for i in range(SAMPLE_SIZE)], dtype=np.int16)
waveform[:] = wave_saw
wave_mix = 0.0
```
Low Frequency Oscillator

A low frequency oscillator, or LFO, is a waveform with a frequency below the audible threshold. LFOs are often used to modulate other synthesizer parameters, such as the pitch or amplitude of an audio oscillator's waveform.

We'll run an LFO at different rates by pressing the SET button to cycle among them in the `lfo_rates` list.

The `lfo1` object is created with a rate and waveshape, in this case a `sine`, but this could be any shape you like, such as a triangle or saw.

```python
lfo_rates = (0.1, 0.5, 0.8, 1.5, 3.0, 6.0, 7.0, 8.0)
lfo_index = 0
lfo1 = synthio.LFO(rate=(lfo_rates[lfo_index]), waveform=wave_sine)  # rate is in Hz
synth.lfos.append(lfo1)
```

Helper Functions

We create a number of helper functions to control NeoPixels when buttons are pressed, remap and clamp values, and provide linear interpolation (lerp) between values.

```python
# map s range a1-a2 to b1-b2
def map_range(s, a1, a2, b1, b2):
    return b1 + ((s - a1) * (b2 - b1) / (a2 - a1))

# mix between values a and b, works with numpy arrays too, t ranges 0-1
def lerp(a, b, t):
    return (1-t)*a + t*b

def light_button_pixels(button_number):
    pixels[pix_map[button_number]+1] = 0xFF0000
    pixels[pix_map[button_number]-1] = 0xFF0000
    pixels.show()

def reset_button_pixels(button_number):
    pixels[pix_map[button_number]+1] = 0x000000
    pixels[pix_map[button_number]-1] = 0x000000
    pixels.show()

def clamp(v, low, high):
    return min(max(v, low), high)
```

Main Loop

The main loop of the program checks for button presses and switch events.
When a note button is pressed a corresponding synth note is played (and the NeoPixels surrounding the button position light up). You'll notice the notes have a short attack and a long release, thanks to the ADSR envelope values we created initially.

If the SCORE button is held, the note buttons will also play a unison note one octave down.

Pressing the SET button increments the LFO rate, while a long press decrements it.

Flipping the MODE switch changes the waveform pair that's mixed between.

When the SKILL switch is in the center position, all pressed notes will sustain indefinitely -- perfect for nice drone chords! In the non-center positions the notes will release when the note buttons are released.

```python
# watch for mod buttons to be pressed
mod_button_event = mod_buttons.events.get()
if mod_button_event:
    mod_key = mod_button_event.key_number
    if mod_button_event.pressed:
        if mod_key == 0:  # SET switch
            last_mod_button_event_time = time.monotonic()
        if mod_key == 1:  # enable octaves
            octaves = True
        if mod_button_event.released:
            if last_mod_button_event_time and mod_key == 0:  # short press-release
                increase LFO rate
                lfo_index = clamp(lfo_index+1, 0, len(lfo_rates)-1)
                print(lfo_index)
                lfo_rate = lfo_rates[lfo_index]
                lfo1.rate = lfo_rate
                last_mod_button_event_time = 0
            if mod_key == 1:  # disable octaves
                octaves = False
            # long press slows the LFO rate
            if last_mod_button_event_time != 0 and time.monotonic() -
                last_mod_button_event_time > 1.0:
                last_mod_button_event_time = 0
                lfo_index = clamp(lfo_index-1, 0, len(lfo_rates)-1)
                lfo_rate = lfo_rates[lfo_index]
                lfo1.rate = lfo_rate

    # watch for note buttons to be pressed
    note_button_event = note_buttons.events.get()
    if note_button_event:
        i = note_button_event.key_number
        if note_button_event.pressed:
            if octaves:
                synth.press((note_list[i]+(octave*12), note_list[i]+(octave*12)-12))
            else:
                synth.press((note_list[i]+(octave*12),))
            light_button_pixels(i)
        if note_button_event.released:
            if not hold:
                reset_button_pixels(i)
```
```python
synth.release((note_list[i]+(octave*12), note_list[i]+(octave*12)-12))
reset_button_pixels(i)

# watch for switches to be changed
switch_event = switches.events.get()
if switch_event:
    sw = switch_event.key_number
    if switch_event.pressed:
        if sw == 0:  # MODE toggle right
            mixer.voice[0].level = 0.45
            # wave_mix = 0.5
            waveset = 0
        if sw == 1:  # SKILL toggle center
            hold = True

    if switch_event.released:
        if sw == 0:  # MODE toggle center
            mixer.voice[0].level = 0.95
            waveset = 1
        if sw == 1:  # SKILL toggle right or left
            hold = False
        for r in range(len(note_list)):  # turn off all notes
            # if octaves:
                synth.release((note_list[r]+(octave*12), note_list[r]+(octave*12)-12))
        for h in range(len(pix_map)):  # turn off held pixels
            reset_button_pixels(h)

lfo_val_for_lerp = map_range(lfo1.value, -1, 1, 0, 1)
if waveset == 0:
    waveform[:] = lerp(wave_sine, wave_weird1, lfo_val_for_lerp)
else:
    waveform[:] = lerp(wave_saw, wave_noise, lfo_val_for_lerp)
```
Assemble the Computer Perfection Synth

Open the Panel
Remove the two screws at the bottom of the top panel, then pivot the panel open.
Remove PCB
Unscrew the four screws holding the game PCB in place.
Desolder Wires
Desolder the two lid switch wires, as well as the two piezo buzzer wires.

Remove Microcontroller
The Computer Perfection game ran on a 4-bit microcontroller, the Matsushita MN1400ML.

Carefully pry it up to remove it from the DIP socket. We won't be re-using the chip, but we will use the socket to connect all of the buttons and switches to our Metro M7. A rather large upgrade!
Wiring Diagram

The Fritzing diagram and schematic below show the connections we'll make to use the Computer Perfection game's buttons and switches as GPIO inputs on the Metro M7.

You can also see the connections to the I2S amplifier and NeoPixels.
DIP Header Wiring
Rather than tediously solder wires to the underside of the board, we'll use this super cool DIP header that Jan Goolsbey () suggested.

These are designed to clamp down on flat ribbon wires, but in this case we'll use silicone sheathed wire for the necessary flexibility and low profile that'll allow the wire bundle to be routed out and under the board.

A pair of tweezers works well for pressing each wire into its pin slot -- no need to remove insulation, the pin does the work for you.

Using the Fritzing diagram as a guide, run each wire to its respective pad on the Proto-Screwshield.
Cap
Press the connector's cap into place.

Socket Connection
Paying attention to the orientation, place the DIN connector in the DIP socket, being careful to seat each leg so that nothing is misaligned or bent.

Use tape to neatly dress the wiring off to the left side of the board.
Wiring Length

Remount the PCB to the Computer Perfection top panel so you can pull the wires to even length in the Proto-Screwshield.

Remove a bit of the insulation and solder each wire.
To prep the I2S amp, first solder the header strip and terminal block as shown here.

Solder a 7-pin section of socket header pins to the proto area of the Proto-Screwshield.

Run wires to pins D9, D10, D12, GND, and 3.3V as shown to correspond with the pins of the amplifier, connecting them to their respective pins on the headers.

Feed the speaker wires through the base and screw them into the terminal block.

Plug the amplifier into the board.
Mount Speaker

Use a piece of double-stick foam tape to mount the speaker in the base as shown.
NeoPixel Wiring

We can't re-use the existing LEDs on the board easily because they are wired to the same pins as the input buttons -- the original microcontroller pins toggled quickly between input and output to perform both functions -- so we'll upgrade the Computer Perfection with a strip of NeoPixels!

Solder terminal blocks to the Proto-Screwshield to accommodate the NeoPixel wiring to pin D11, GND, and 3.3V.
LED Placement
Remove the PCB and test fit the NeoPixel strip, then cut off the excess as shown.

Use the uGlu dashes or other adhesive to affix the NeoPixels, running the wiring to the side as shown.

Put the PCB back onto the top panel.
Screw the NeoPixel wires into their blocks and then you can plug the Proto Screwshield into the Metro M7.
USB Cable

Assemble the DIY USB cable USB-C end to the ribbon cable and plug it into the Metro M7.

Run the ribbon through the base as shown, then attach the micro-B socket.
Metro Placement

Use double-stick foam tape to adhere the Metro M7 to the inner base, then screw the top panel into place.
Magnetic USB Cable
You can optionally use a magnetic tipped USB cable to make the connections a bit easier.
Playing the Computer Perfection synthesizer is satisfying and simple! There are just a few options to know about:

- Press any blue button to play a note. You can play multiple notes at once to create chords.
- If the Sustain (SKILL) switch is in position 3 or 1, the note will only sustain as long as you hold the button or buttons.
- Move the Sustain (SKILL) switch to the center position marked 2 and any new notes will drone indefinitely until you move the switch to the left or right.
- Hold the Sub-Octave (SCORE) button to add an additional note one octave below any new notes you play.
- Press and release the green LFO (SET) button to increase the LFO rate by one step in the rate list. Long-press (about a second) it to decrease the LFO rate by one step.
- Set the Waveform (MODE) switch to the center for one waveset or to the right for the other -- the wavesets contain two mixed waveforms that the LFO modulates between.
- Leave the GAME switch in the left-most position otherwise the other switches and buttons will misbehave.