Digital Maker CIC

We make makers...

CircuitPython Build #1 for "Oor Monsters / Oor Future" by Gabrielle Reith & Philip Thompson

For an overview of the Oor Monsters exhibition, see the Aberdeen Performing Arts website: https://www.aberdeenperformingarts.com/whats-on/oor-monsters-oor-future

---

Overview

#OptimisticOracle (Sustainable Sam)

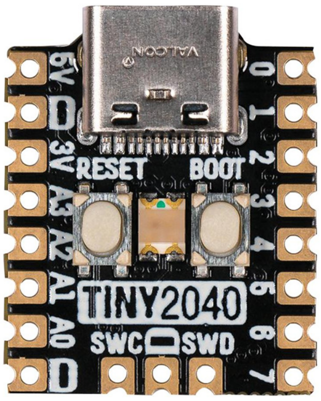
FORTUNE Teller

This installation used a Pimoroni "Tiny2040" (8mb) (" A postage stamp sized RP2040 development board with a USB-C connection, perfect for portable projects, wearables, and embedding into stuff...") a Speaker and a PIR (HC-SR501). Using the fantastic & easily implemented CircuitPython MP3 Audio utilities – we stitch together some randomly chosen mp3 phrases to make an "Optimistic Oracle" inspired by the Zoltar unit in Big! (and named after the lovely Sam Bentley, who delivers round ups of progressive Green / Eco technology developments & good news stories from around the world.)

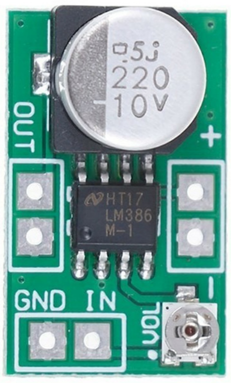
For more info on CircuitPython & specfic versions for a Tiny2040, visit

Learn.adafruit.com/category/circuitpython

Parts

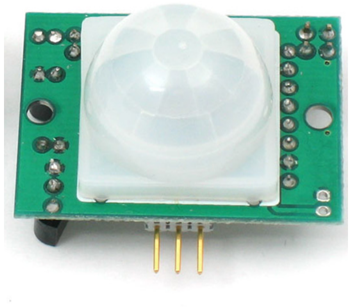


Pimoroni Tiny2040



LM386 Mini Micro Amp

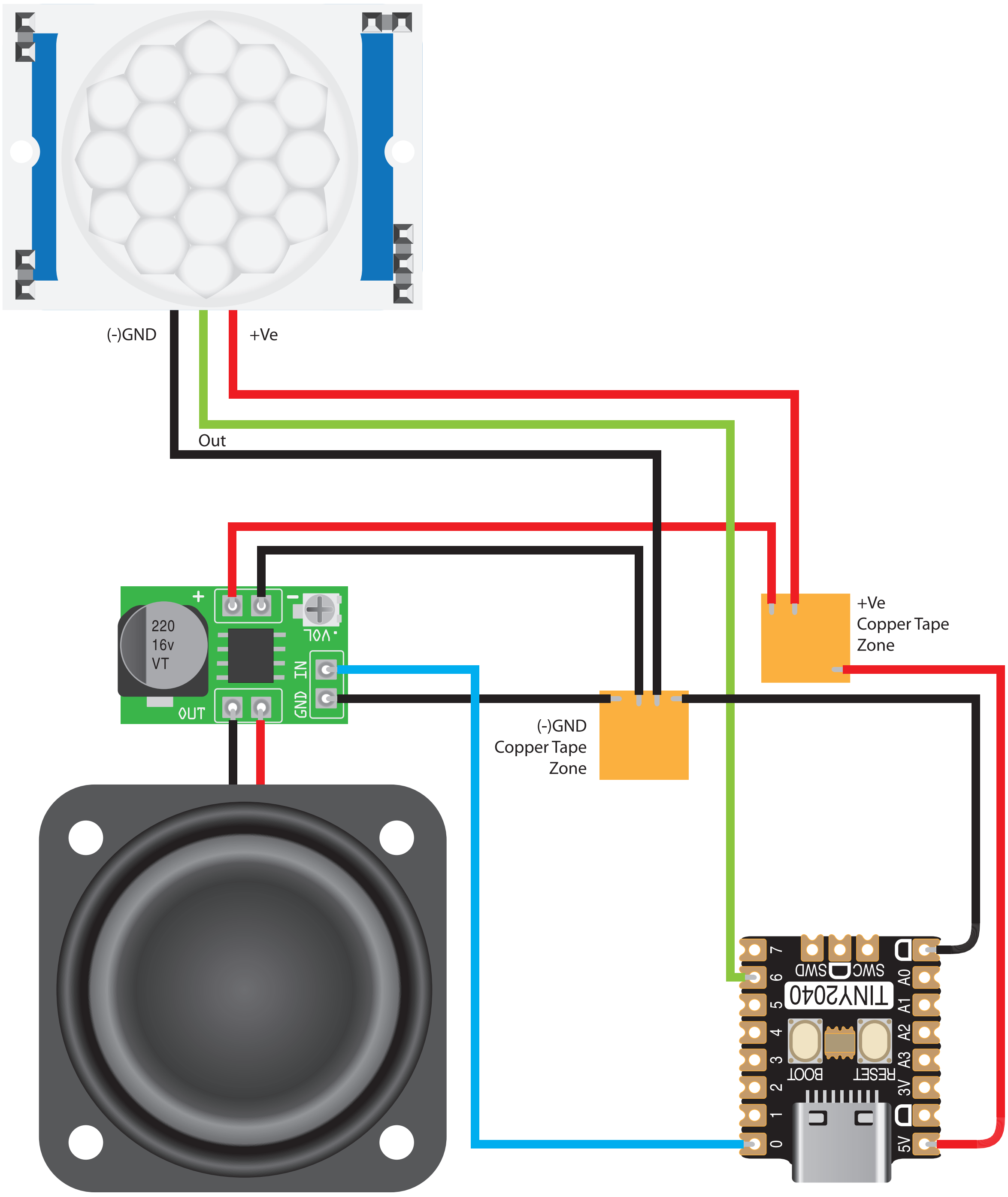
Mini Speaker 4Ω 3w



PIR sensor

Assembly

Gabrielle & Philip used recycled / found wood from old furniture & offcuts to make the "Oor Monsters / oor Future" exhibition. The Optimistic Oracle has a 10mm ply layer which had "tracks" routed into the back of the ply for wiring / component placement with a hole drilled, allowing the PIR sensor to sit flush in the routed recess. The Tiny2040 had to be accessible from the bottom of the artwork (for ease of power supply) - and the placment of the speaker & amplifier were calculated accordingly from the routed recesses. The wiring was quite minimal, the PIR was connected directly to the Tiny2040 on pin GP6, and the Amplifier was connected to pin GP0. The speaker was connected to the amplifier. We also created a "Shared Ground" and "Shared +Ve" by adding copper tape "zones" & soldering a wire from the Vbus+ (5V) pin to one copper tape patch & a GND pin to the other zone (Kept far apart & labeled with a pen for ease). The +Ve & GND wires for each component were then linked to the zones via a soldered wire too (PIR & Amp). Once all wires were connected / soldered, we tested the unit before glueing / screweing the artwork together. Below is a diagram of the wiring.



Code

Because we use MP3s saved to the Tiny2040, we need to create a folder on the Tiny2040 called “Mp3”. We’ve also saved the MP3s with standardised naming, so it’s easier to pull them into the code (using random number generation, instead of having to collate a big array of names to pull from.) The way we created a “prediction” was to pull in one of nine “I predict” type phrases, then a static “you will” mp3, and finally one of 29 “action” mp3s. The prediction is started when the PIR detects movement & only checks when the MP3 is finished playing.

```python
# PIR / MP3 circuitPython code on a Tiny2040 for "Optimistic Oracle" in the Oor Monsters / Oor Future exhibition
# by Gabrielle Reith (Small Stories) & Philip Thompson (Digital Maker cic)
# import the neccesary libraries (thankfully all part of the base libraries so no need to physically install them onto your Tiny2040)

import board
import audiopwmio
import audiomp3
import digitalio
from random import choice

PIR_PIN = board.GP6   # Pin number connected to PIR sensor output wire.
AUDIO_PIN = board.GP0  # Pin connected to Amp "IN" terminal.

# ada fruit PIR & MP3 circuitPython help:
# https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/circuitpython-code
# https://learn.adafruit.com/mp3-playback-rp2040

pir = digitalio.DigitalInOut(PIR_PIN)
pir.direction = digitalio.Direction.INPUT
audio = audiopwmio.PWMAudioOut(AUDIO_PIN)

old_value = pir.value

# function to pull together the random "prediction"


def talk() -> None:
    # all "I foresee" type intros begin with "p_" e.g. "p_4.mp3" (we made 9 mp3s for these )
    predict = "p_{}".format(choice(range(1, 9)))
    # set up the MP3 to play via the MP3Decoder library
    v1 = audiomp3.MP3Decoder(open("mp3/{}.mp3".format(predict), "rb"))
    #play the file
    audio.play(v1)
    while audio.playing:
        pass
    # the audio has now completed playing, we put in a single mp3 "you will"
    v2 = audiomp3.MP3Decoder(open("mp3/you_will.mp3", "rb"))
    audio.play(v2)
    while audio.playing:
        pass
    # "you will" has completed playing...
    # generate a random "action" files beginning with "a_" e.g. a_10.mp3 ( (there are 29 action mp3 files )
    action = "a_{}".format(choice(range(1, 29)))
    v3 = audiomp3.MP3Decoder(open("mp3/{}.mp3".format(action), "rb"))
    audio.play(v3)
    while audio.playing:
        pass
    # action mp3 has finished playing, return back to the main loop
    return


while True:
    # check the pir.value (if True (movement detected) fire up the mp3 player)
    pir_value = pir.value
    if pir_value:
        # Check if this is the first time movement was
        # detected and speak a message!
        if not old_value:
            talk()
            while audio.playing:
                pass
    else:
        print("done")

    old_value = pir_value
```

Download the code & files here: digital-maker.co.uk/wp-content/uploads/2023/04/optimisticOracle.zip