Digital Maker CIC

We make makers...

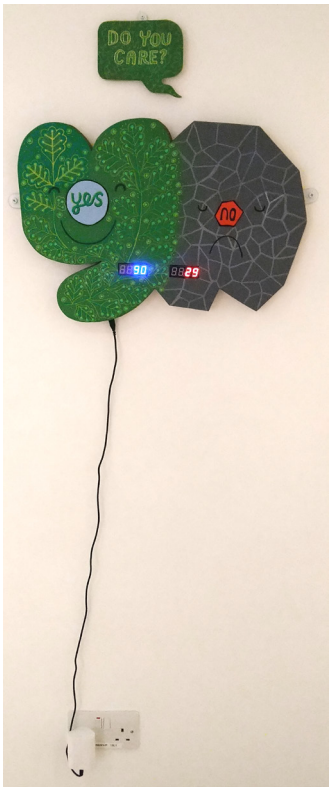
CircuitPython Build #2 for "Oor Monsters / Oor Future" by Gabrielle Reith & Philip Thompson

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

#YesNo ("do you care?")



This installation used a Raspberry Pi "Pico" ("a low-cost, high-performance microcontroller board with flexible digital interfaces.") 2x4 digit (7 segment) led displays, and two "home made" buttons (made with copper tape, wire, foam sponge & wood). This was created to see (in a fun & very unscientific way) if people cared about climate change issues or not, with a concious (tongue in cheek) bias, as the "yes" button was far bigger than the "no"... we assumed you'd have to be extra mean to hit the no button... We also 3d printed a "Pico housing" to make consealing the pico inside the artwork neater, we have uploaded the 3d file for that on our website too.

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

Learn.adafruit.com/category/circuitpython

Parts

Raspberry Pi Pico

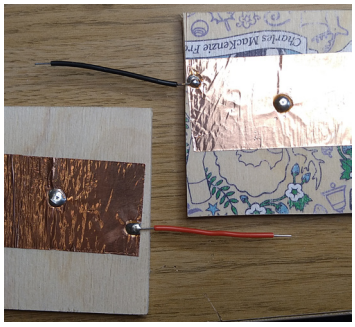
2xAdafruit 0.56" 4-Digit 7-Segment Display

Copper Tape

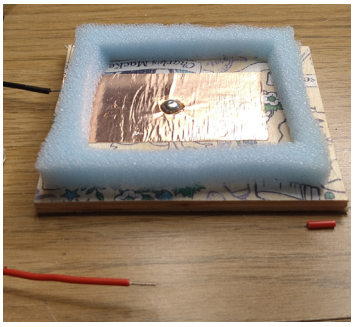
Electrical Wire

Assembly

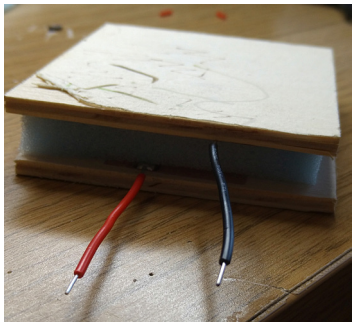
Gabrielle & Philip used recycled / found wood from old furniture & offcuts to make the "Oor Monsters / oor Future" exhibition. The "Do you care? monster" has 2 layers of 10mm ply (one with "tracks" routed into the back of the ply for wiring / component placement with a holes cut for the buttons & LED displays). The Pico had to be accessible from the bottom of the artwork (for ease of power supply). We made "Cutom Buttons" by sandwiching 2 thin ply squares around a thin foam square (with a hole cut out the middle). We followed the Instructables "Simple-Electronic-Button" how to by gitterbug23 with some tweaks, and glued the "button shape" on top of the thin ply button. We also added in "common + & -" copper tape areas to make wiring easier. Here is a step by step "build" we followed.



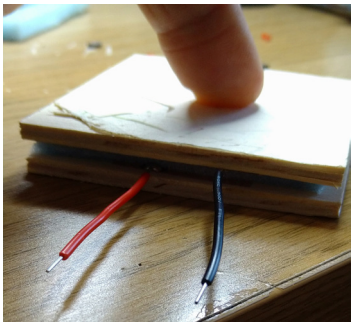
two thin ply squares with opper tape & soldered wire onto surface. We added a solder "bump" to the middle for extra "contact opportunity" when pressed.



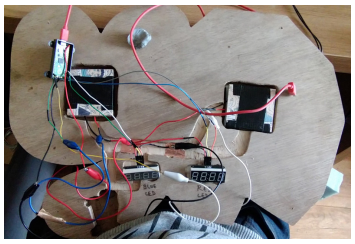
We added the foam square (with hole cut so the contacts can be made!)



We sandwiched foam between the two sides of the button & taped them lightly with duct tape (not shown)

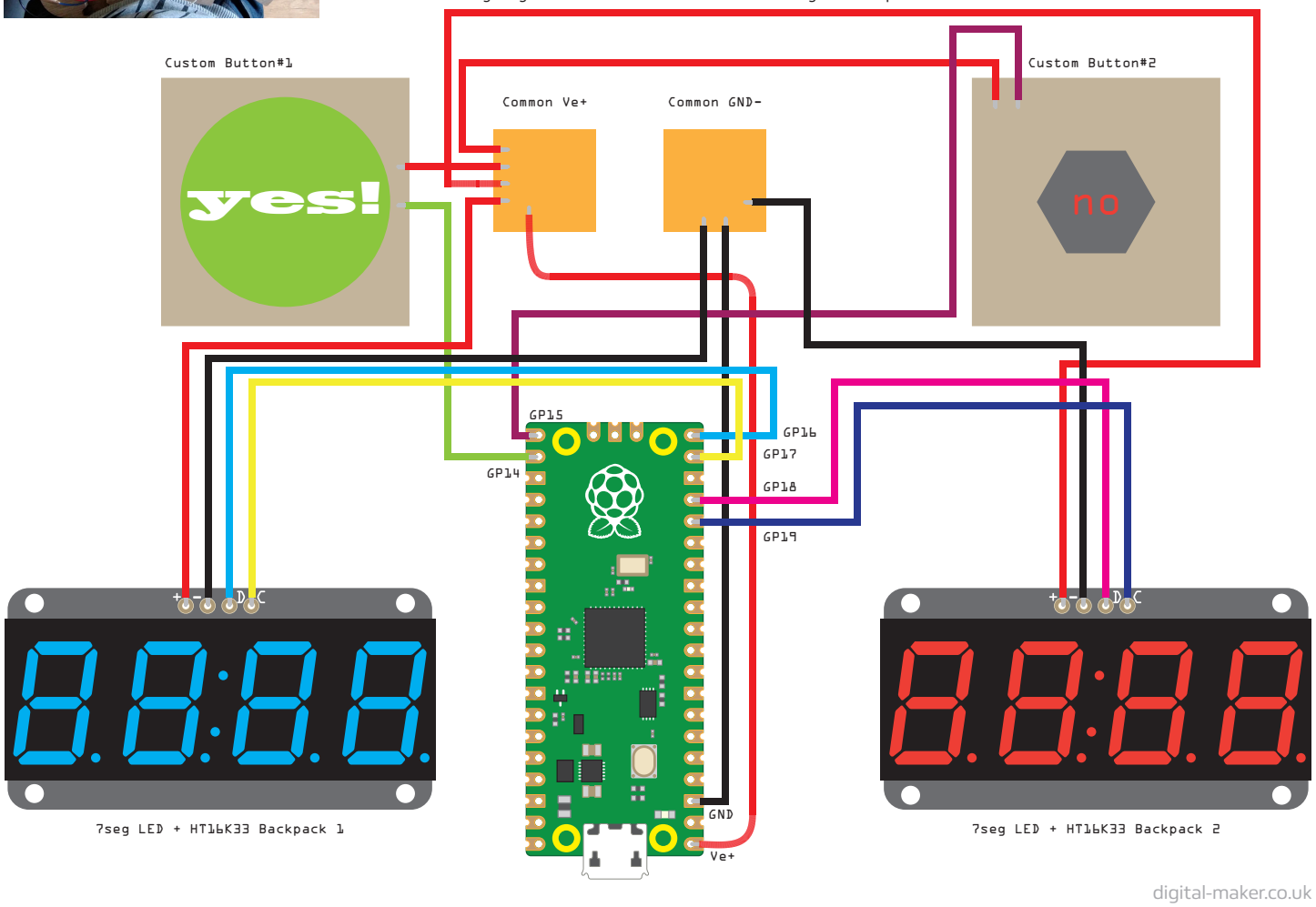


we tested the press / contact & found that solder alone oxidised over a few hours, so we added in copper tape to the "lump" to ensure clean contacts per press.



Routed ply for components & wiring to sit in (Crocodile clips for testing purposes) as the final wiring was "tight". Phil used colour coded wiring to ensure the red/blue LED display was affected by the correct button (YES/NO).

Wiring diagram for the Pico + 2xButtons & 2x7SegLED Backpacks



Code

The LED “Backpacks” have a great Library that makes using them relitively straightforward. Just make sure you have the SDA & SCL (‘D’ & ‘C’ on the unit) in the right pins! We created two variables to keep track of counters (C1 & C2) & each button press adds to the relevant counter. We also added in a small pause to avoid any “bounces” (button presses too quickly)

```
import board
import busio
import digitalio
from adafruit_ht16k33.segments import Seg7x4
import time

#set up 2 buttons (+Ve and a "ground" that leads into a GP pin, "listening" for a pulse!)
b1 = digitalio.DigitalInOut(board.GP14)
b1.switch_to_input(pull=digitalio.Pull.DOWN)
b2 = digitalio.DigitalInOut(board.GP15)
b2.switch_to_input(pull=digitalio.Pull.DOWN)

# i2c on the backpack makes wiring easier & use fewer wires!
# sda1 = board.GP16 = #pin 21 & scl1 = board.GP17 = #pin 22
# sda2 = board.GP18 = #pin 24 & scl2 = board.GP19 = #pin 25
i2cA = busio.I2C(board.GP17, board.GP16)
i2cB = busio.I2C(board.GP19, board.GP18)

#create 2 displayObjects with the ht16k33 library for Seg7x4
display1 = Seg7x4(i2cA)
display2 = Seg7x4(i2cB)
#keep the display on the duller side, as they are very bright!
display1.brightness = 0.1
display2.brightness = 0.1

#function to set the correct display unit's counter
def updateNum(disp, count) -> None:
    disp.fill(0)
    disp.print(count)

# start the counter variables at 0
c1 = 0
c2 = 0
# blank the counter LEDs
updateNum(display1, c1)
updateNum(display2, c2)

while True:
    if b1.value:
        c1 +=1
        # check to see the number can be displayed on the 4 digit display!
        if c1 > 9999:
            # if not set this counter to 1
            c1 = 1
        updateNum(display1, c1)
        while b1.value:
        #pause for a little bit to stop multi touches being counted
            time.sleep(0.1)

    if b2.value:
        c2 += 1
        if c2 > 9999:
            c2 = 1
        updateNum(display2, c2)
        while b2.value:
            time.sleep(0.1)
```

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