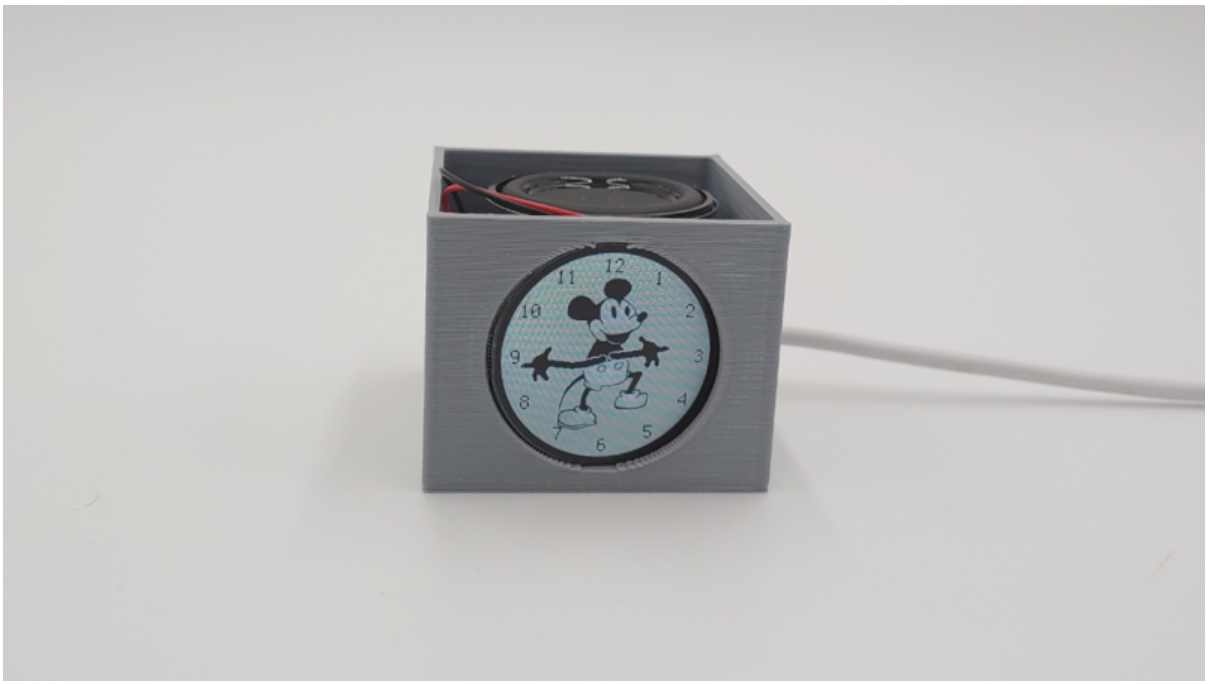
Cartoon Character Clock

Created by Tim C

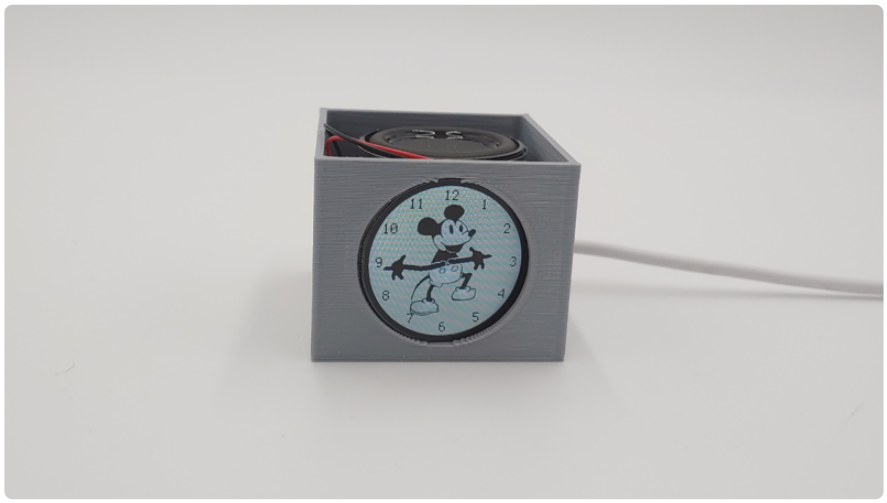


https://learn.adafruit.com/cartoon-character-clock

Last updated on 2025-02-25 06:23:26 PM EST

Table of Contents

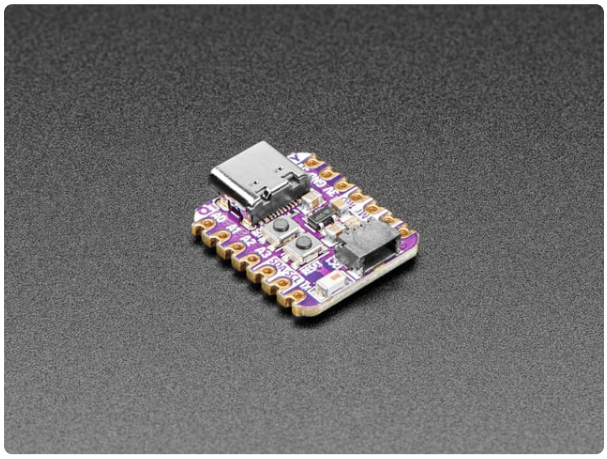
Overview



The GC9A01A round displays (http://adafru.it/6178) are perfect for displaying a clock face. This fun clock uses the Display Analog Clock library (https://adafru.it/1aeV) along with some customized assets inspired by an early 20th-century public domain cartoon (https://adafru.it/1aeA). The arms act as the hour and minute hands of the clock, rotating around to point at the different numbers.

The speaker on top plays jingles from the cartoon. It can be configured to play hourly like a chime, or at a specific time as an alarm. The QTPY ESP32-S2 (http://adafru.it/5325)'s WiFi connection allows the clock to sync with online NTP servers.

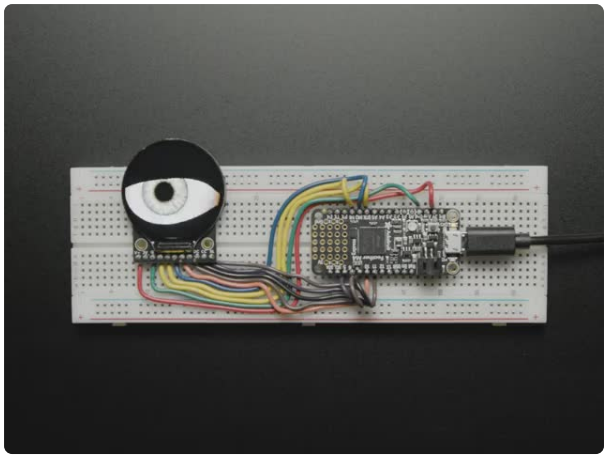
Parts



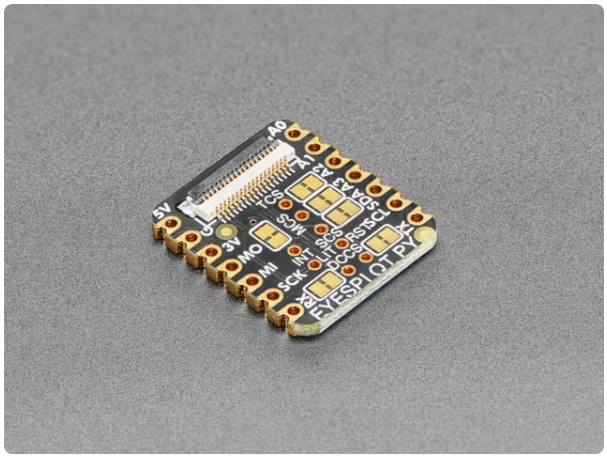
Adafruit QT Py ESP32-S2 WiFi Dev Board with STEMMA QT

What has your favorite Espressif WiFi microcontroller, comes with our favorite connector - the STEMMA QT, a chainable I2C port, and has...

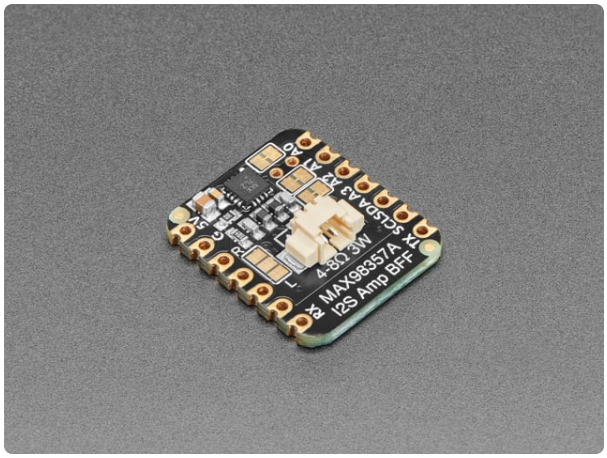
https://www.adafruit.com/product/5325



[Adafruit 1.28" 240x240 Round TFT LCD Display with MicroSD]
'Round these parts we enjoy unusually-shaped displays. And this one certainly fits the description - it's a 1.28" diagonal TFT that comes in a round shape and contains a...
https://www.adafruit.com/product/6178

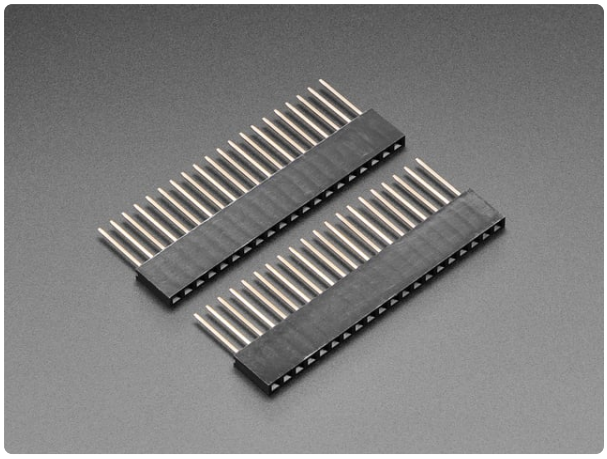


[Adafruit EYESPI BFF for QT Py or Xiao - 18 Pin FPC Connector]
Our QT Py boards are a great way to make very small microcontroller projects that pack a ton of power - and now we have a way for you to add a small, colorful, and bright display to...
https://www.adafruit.com/product/5772



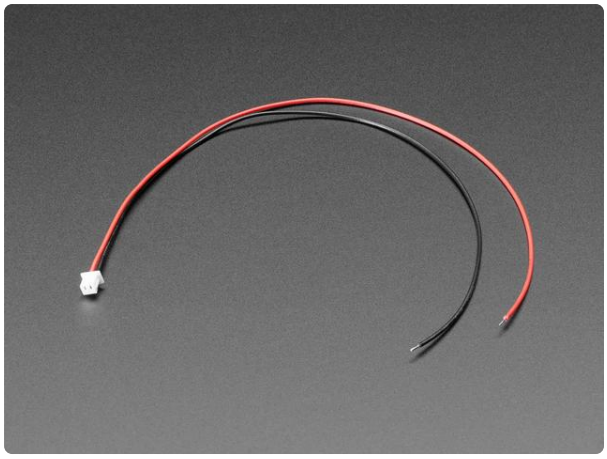
[Adafruit I2S Amplifier BFF Add-On for QT Py and Xiao]
Our QT Py boards are a great way to make very small microcontroller projects that pack a ton of power - and now we have a way for you to add an I2S 3 Watt amplifier, for high quality...
https://www.adafruit.com/product/5770

[Speaker - 40mm Diameter - 4 Ohm 5 Watt]
Hear the good news! This speaker is a great addition to any audio project where you need a 4 Ohm impedance and 3W or less of power.At 40mm diameter it...
https://www.adafruit.com/product/3968



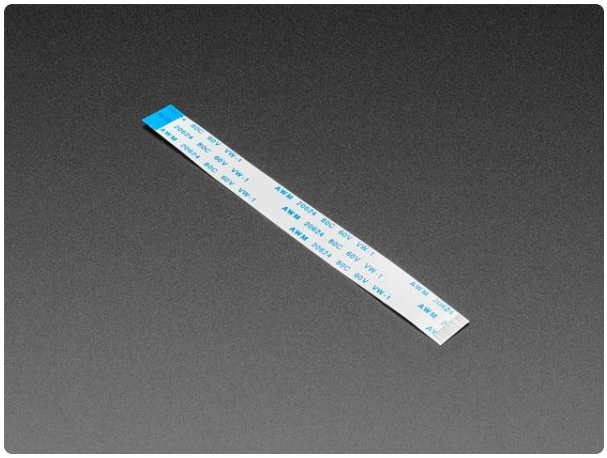
Stacking Headers for Raspberry Pi Pico - 2 x 20 Pin

These two Stacking Headers alone are, well, lonely. But pair them with the https://www.adafruit.com/product/5582



Molex PicoBlade 2-pin Cable - 200mm

When 0.1" is too big, and JST PH's too chunky, the ultra-slim "PicoBlade" is a reliable alternative. These are only 1.25mm pitch, but have a nice clicky... https://www.adafruit.com/product/3922



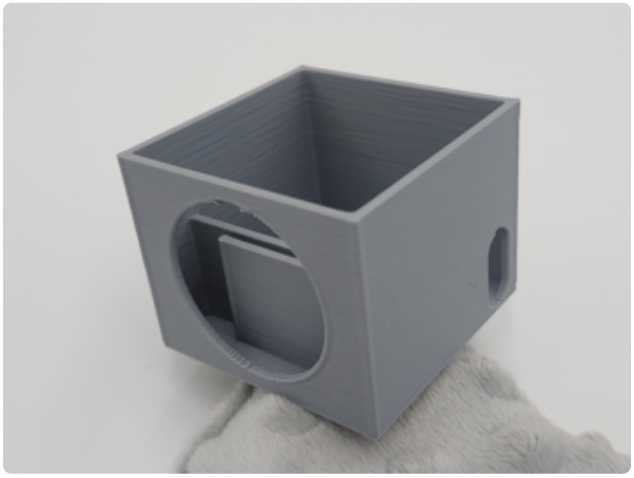
EYESPI Cable - 18 Pin 100mm long Flex PCB (FPC) A-B type

Connect this to that when a 18-pin FPC connector is needed. This 25 cm long cable is made of a flexible PCB. It's A-B style which means that pin one on one side will match... https://www.adafruit.com/product/5239

1 x USB Type A to Type C Cable https://www.adafruit.com/product/4474

Approx 1 meter / 3 ft long

3D Printing



3D Printed Box
STL files for 3D printing will need to be oriented for print using either FDM or SLS machines. Parts were built with PLA filament.

STL file for printing may be downloaded using the link below.

The box was sliced with CURA using the settings below

PLA filament 205c extruder
0.35 layer height
No supports

round_eyespi_display_box.stl

https://adafru.it/1aev

Original Design File

The box was designed using OpenScad (https://adafru.it/XD4), find the source code for it below.

```
wall_thickness = 1.6;
short_wall_height = 25;

difference(){

    cube([50+(wall_thickness*2),50+(wall_thickness*2),42+1], center=true);
    translate([0,0,1]){
        cube([50,50,42+1], center=true);
    }

    translate([0,20,0])
    rotate([90,0,0]){
        cylinder(r=36/2, h=30, center=true, $fn=80);
    }

    translate([
```

```
        -10,
        -(50+(wall_thickness*2))/2 + 9/2 + 4,
        -(42+1)/2 + 18/2 + 1]){

        hull(){
            translate([0,0,-4.5])
            rotate([0,90,])
            cylinder(r=9/2, h=40, center=true, , $fn=30);

            translate([0,0,4.5])
            rotate([0,90,0])
            cylinder(r=9/2, h=40, center=true, $fn=30);
        }
        //cube([40,9,18], center=true);
    }
}


translate([-(50+(wall_thickness*2))/2 + 40/2,
    ((50+(wall_thickness*2))/2) - wall_thickness/2 - 8,
    -(43/2)+(short_wall_height/2)])
cube([40,wall_thickness,short_wall_height], center=true);


translate([(50+(wall_thickness*2))/2 - 40/2,
    ((50+(wall_thickness*2))/2) - wall_thickness/2 - 8 - 4,
    -(43/2)+(short_wall_height/2)])
cube([40,wall_thickness,short_wall_height], center=true);


/*
translate([0,-5,0])
cube([22, 32, 18], center=true);
*/
```

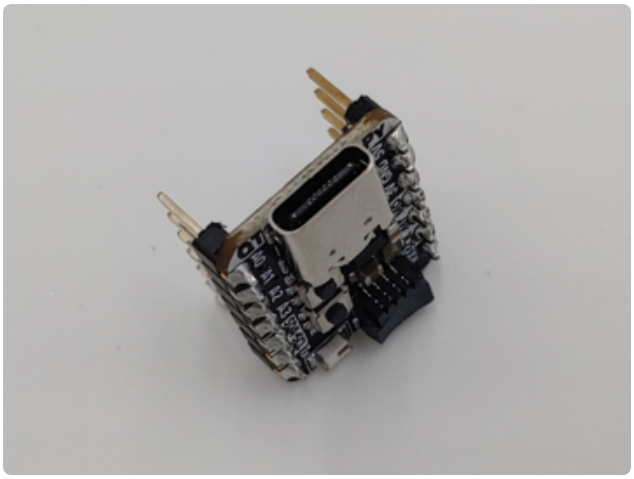
Assembly

The QT Py, BFFs, and EYESPI display make this project easy to assemble. The only soldering required is the headers on the BFFs and QT Py, and the molex picoblade connector wires to the speaker.

Solder Headers

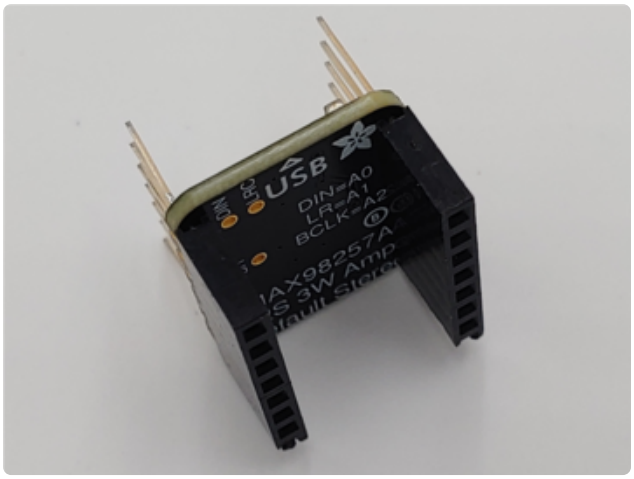
First solder the following header types to the BFFs and QT Py. Make sure that you have the appropriate header paired with the correct board and that the header is facing the proper direction relative to the board. We have a learn guide (https://adafru.it/1a1h) that covers soldering headers if you're new to the process or want a refresher.

- QT Py ESP32-S2 - Standard male header pins
- I2S Audio BFF - Stacking female header pins
- EYESPI BFF - Standard female header pins



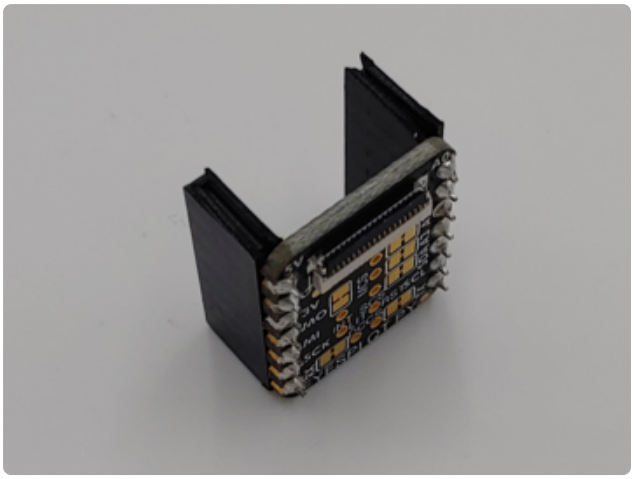
QT Py ESP32-S2

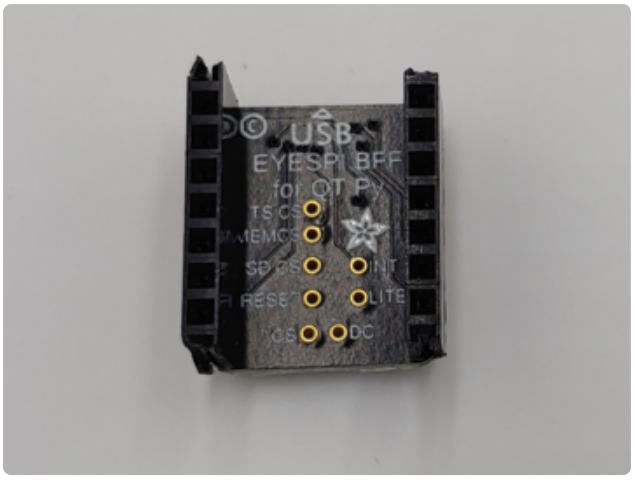
Standard male header with the pins going down on the side opposite the USB-C connector.



I2S Audio BFF

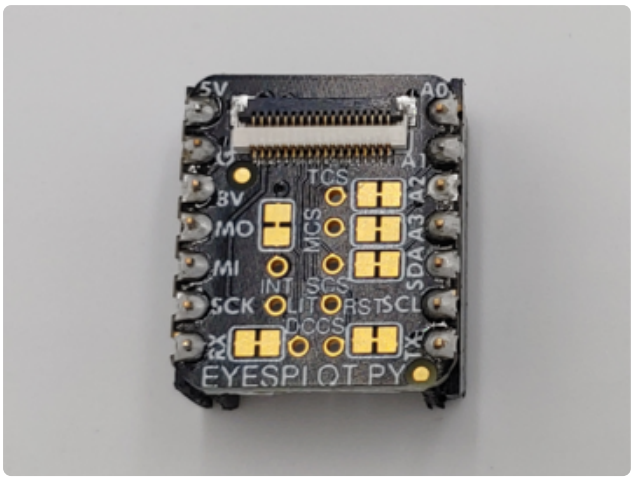
Stacking female header with the female header on the opposite side of the board from the speaker connection.





EYESPI BFF

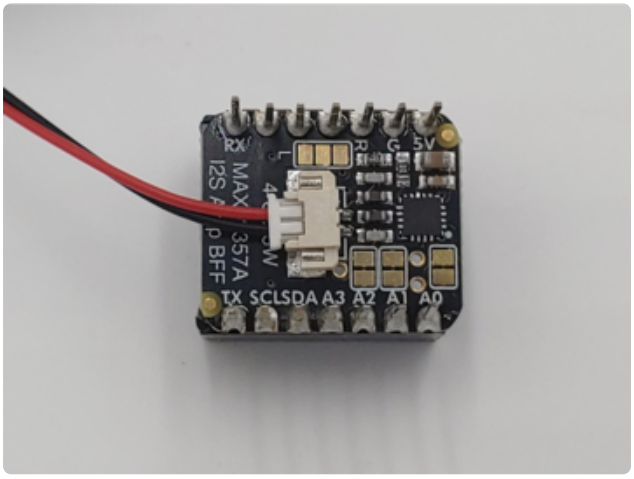
Standard female header with the female header on the opposite side of the board from the EYESPI ribbon cable connector.



Solder Speaker Connector

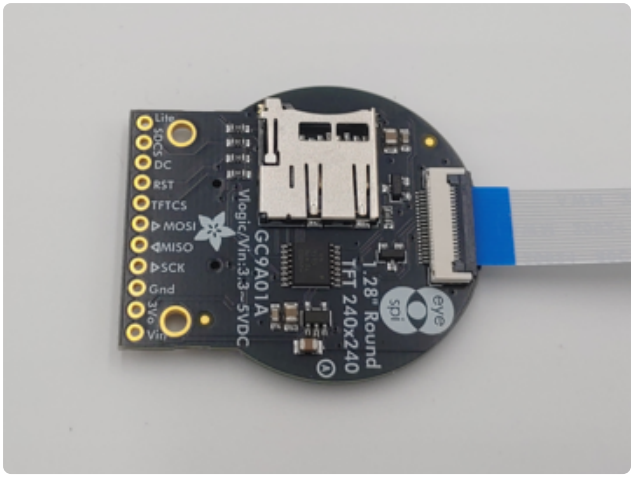
Strip an additional 1/4" or so of the insulation off the ends of the picoblade molex connector wires. Then loop the stripped wire sections through the holes on the speaker, fold it back on itself and solder the connection.

Connect Speaker & Display



Speaker

Connect the picoblade molex connector to the Audio BFF.



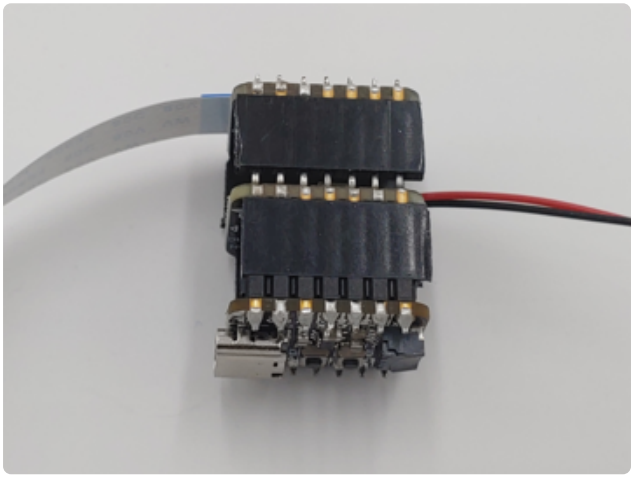
Display

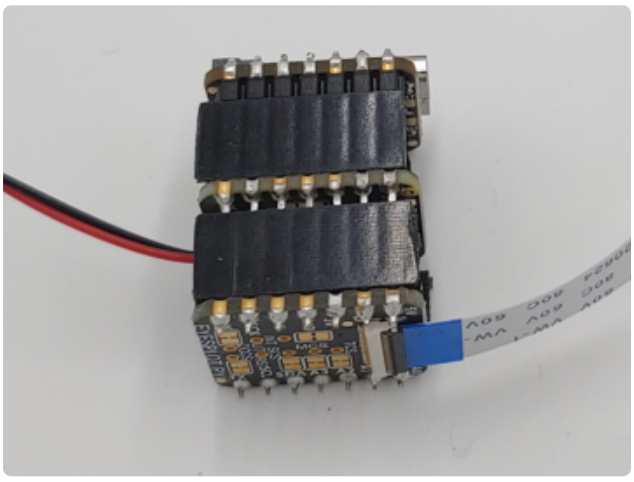
First you must carefully flip up the dark grey flap that secures the ribbon. Gently slip your fingernail or a small flat head screw driver under it, then carefully pry upward. The flap is delicate so be careful, it should not require much force once you've got something up underneath it.

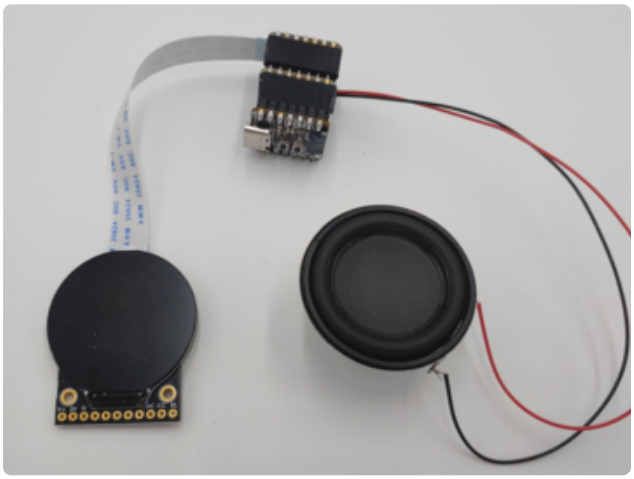
Connect one end of the EYESPI ribbon cable to the EYESPI BFF.
Connect the other end of the EYESPI ribbon cable to the GC9A01A breakout.

Both connections should have the blue tab facing upward. Insert the ribbon cable, then push down the flap to secure it.

Final Assembly



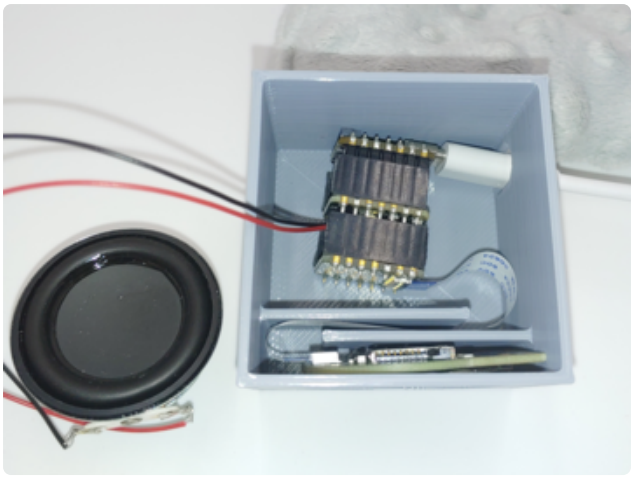




Stack QT Py & BFFs

Connect the QT Py male header pins to the stacking female headers on the Audio BFF. Connect the standard female header pins on the EYESPI BFF to the stacking header pins on the Audio BFF.

The final stack should have the QT Py with its USB C connector face up on one side, and the ribbon cable connector on the EYESPI BFF facing up on the opposite side. The audio BFF with red and black wires coming out is in the middle.



Place into box

Carefully place the display into its slot at the front of the box, feeding the EYESPI ribbon cable down into its channel made by the two offset walls.

Set the QT Py and BFF stack into the open space behind the offset walls with the USB C connector pointing towards the USB Cable hole cut out of the side of the box. Plug in your desired USB C cable to power the project, feeding it through the hole.

Tuck the any extra speaker wire into the box and set the speaker on top of the QT Py and BFF stack.

Project Setup

Are you new to using CircuitPython? No worries, there is a full getting-started guide here (https://adafru.it/cpy-welcome).

Plug the device into your computer with a known good USB cable (not a charge-only cable). The device will appear to your computer in File Explorer or Finder (depending on your operating system) as a flash drive named **CIRCUITPY**. If the drive does not appear, you can install CircuitPython (https://adafru.it/Amd) on your device and then return here.

Download the project files with the Download Project Bundle button below. Unzip the file and copy/paste the **code.py** and other project files to your **CIRCUITPY** drive using File Explorer or Finder (depending on your operating system).

Connect to WiFi

In order to communicate with the NTP servers your CircuitPython device needs to be connected to the internet. The easiest way to do this is to include your WiFi network credentials in a **settings.toml** file. If that file doesn't exist, then create it. Add these

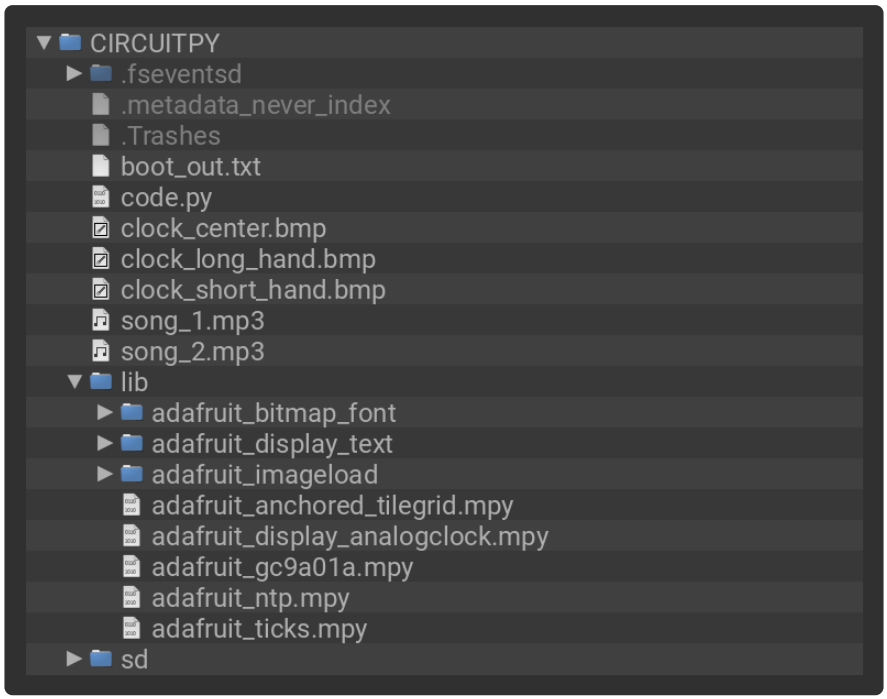
entries to the file and fill in the details for your own WiFi network. See this guide page (https://adafru.it/18f9) for more info about creating the **settings.toml** file

```
CIRCUITPY_WIFI_SSID="network_name"
CIRCUITPY_WIFI_PASSWORD="network_password"
```

Once that file is saved reboot your device and it will automatically connect to WiFi.

Drive Structure

After copying the files, your drive should look like the listing below. It can contain other files as well, but must contain these at a minimum.



Code

The **code.py** for the project is shown below.

There are 3 variables near the top which you can modify to change the clocks behavior.

- `UTC_OFFSET` - Set it to a negative or positive whole number to shift the time by relative to UTC.
- `HOURLY_CHIME` - Set it to `True` to enable the chime and `False` to disable.
- `ALARM_TIME` - Set it to None to disable the alarm, or a tuple containing hour and minute values for the alarm to go off at. These are in the local time after the `UTC_OFFSET` has been applied.

```
# SPDX-FileCopyrightText: 2025 Tim Cocks
#
# SPDX-License-Identifier: MIT
"""
Cartoon Character Clock

This project features an analog clock face rendered on a round display.
The art and songs are inspired by an early 20th-century public domain
cartoon.

"""
import os
import time
import board
from adafruit_display_analogclock import AnalogClock
from adafruit_gc9a01a import GC9A01A
import rtc
import socketpool
import displayio
from fourwire import FourWire
import adafruit_ntp
import wifi
import audiobusio
from audiomp3 import MP3Decoder

# Set the desired offset from UTC time here.
# US Eastern Standard Time is -5
# US Eastern Daylight Time is -4
UTC_OFFSET = -5

# enable or disable the hourly chime jingle
HOURLY_CHIME = True

# Set to a tuple containing hour and minute values to
# enable the alarm e.g. 13, 25 will play the alarm at
# 1:25 pm adjusted for the given UTC_OFFSET.
ALARM_TIME = None
#ALARM_TIME = (13, 25)

# WiFi Setup
# Get wifi AP credentials from a settings.toml file
wifi_ssid = os.getenv("CIRCUITPY_WIFI_SSID")
wifi_password = os.getenv("CIRCUITPY_WIFI_PASSWORD")
if wifi_ssid is None:
    print("WiFi credentials are kept in settings.toml, please add them there!")
    raise ValueError("SSID not found in environment variables")

try:
    wifi.radio.connect(wifi_ssid, wifi_password)
except ConnectionError:
    print("Failed to connect to WiFi with provided credentials")
    raise

# pylint: disable=unsubscriptable-object

if HOURLY_CHIME or ALARM_TIME is not None:
    # Audio Setup
    audio = audiobusio.I2SOut(board.A2, board.A1, board.A0)
    songs = ["song_1.mp3", "song_2.mp3"]
    mp3 = open(songs[0], "rb")
    decoder = MP3Decoder(mp3)

# Display Setup
spi = board.SPI()
tft_cs = board.TX
tft_dc = board.RX

displayio.release_displays()
```

```
display_bus = FourWire(spi, command=tft_dc, chip_select=tft_cs, reset=None)
display = GC9A01A(display_bus, width=240, height=240)
display.rotation = 90

# Sync time from NTP server
pool = socketpool.SocketPool(wifi.radio)
ntp = adafruit_ntp.NTP(pool, server="time.nist.gov", tz_offset=0,
cache_seconds=3600)
rtc.RTC().datetime = ntp.datetime

# Initialize the clock face
clockface = AnalogClock(
    "clock_short_hand.bmp",
    "clock_long_hand.bmp",
    (120, 120), 106, number_label_scale=2,
    background_color=0x989a97,
    background_img_file="clock_center.bmp",
    background_img_anchor_point=(0.5,0.5),
    background_img_anchored_position=(display.width//2, display.height//2 - 2)
)

# set the clockface to show on the display
display.root_group = clockface

print(f"current time {time.localtime().tm_hour + UTC_OFFSET}:
{time.localtime().tm_min}")
cur_hour = time.localtime().tm_hour + UTC_OFFSET
cur_minute = time.localtime().tm_min

clockface.set_time(cur_hour, cur_minute)

while True:
    # If we need to update the clock hands
    if cur_hour != time.localtime().tm_hour + UTC_OFFSET or cur_minute !=
time.localtime().tm_min:
        # store current values to comapre with next iteration
        cur_hour = time.localtime().tm_hour + UTC_OFFSET
        cur_minute = time.localtime().tm_min

        # update the clock face
        clockface.set_time(cur_hour, cur_minute)

        # if the hourly chime is enabled, and it's the top of the hour
        if HOURLY_CHIME and cur_minute == 0:
            # play the hour chime jingle
            audio.play(decoder)
            while audio.playing:
                pass

        # if the alarm is enabled and the current time is what
        # it was set to.
        if ALARM_TIME is not None and \
                cur_hour == ALARM_TIME[0] and cur_minute == ALARM_TIME[1]:

            # open the alarm song file
            decoder.file = open("song_2.mp3", "rb")

            # play the alarm song twice
            for i in range(2):
                audio.play(decoder)
                while audio.playing:
                    pass
                time.sleep(0.5)

            # re-open the hourly chime file
            decoder.file = open("song_1.mp3", "rb")

    time.sleep(3)
```