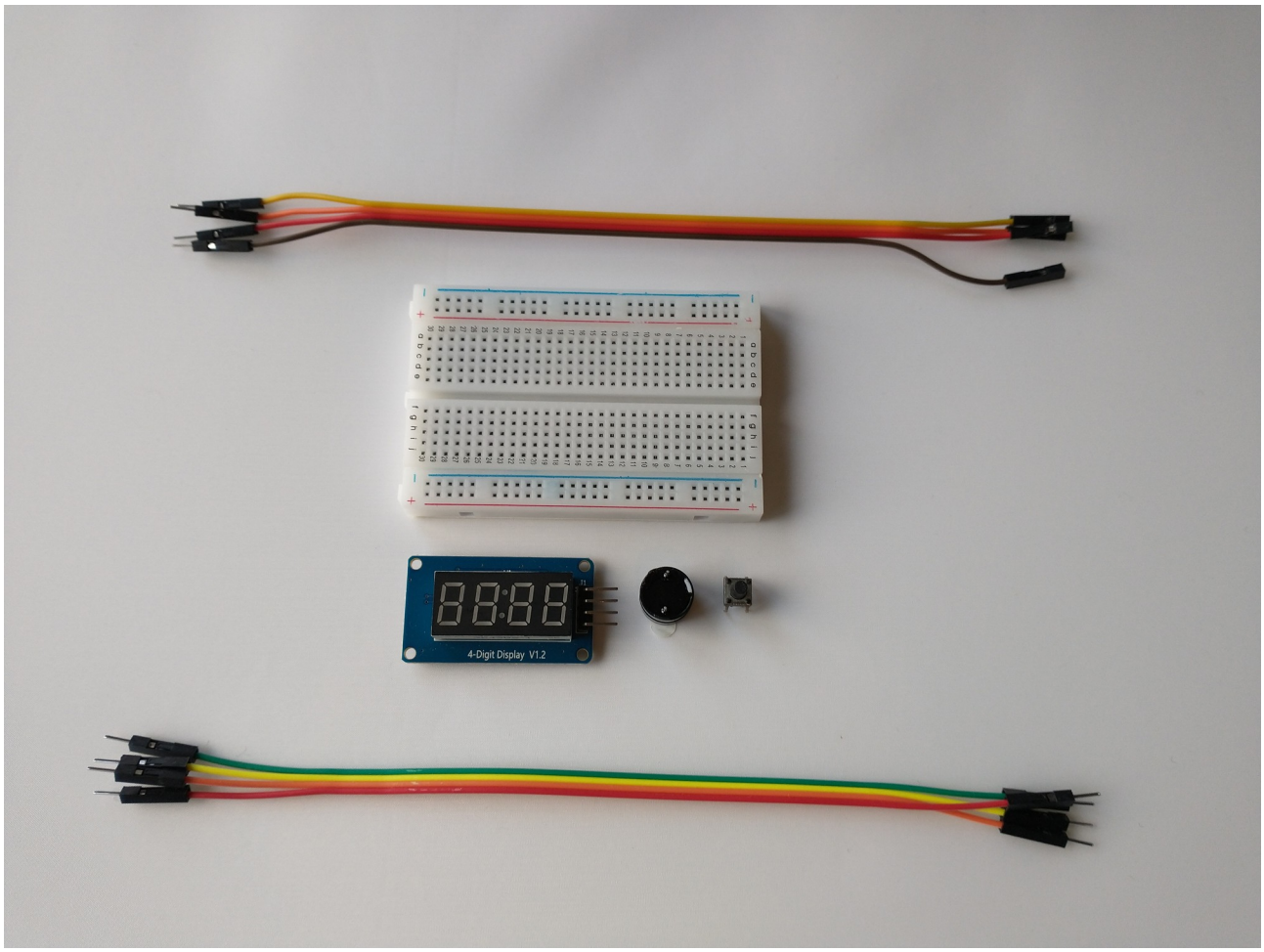
Digital Alarm Clock Project Kit for Raspberry Pi Pico.

This kit contains everything you need to drive a TM1637 4 digit 7 segment display module from your Raspberry Pi Pico microcontroller board and make it behave like a digital alarm clock.

The kit includes:

TM1637 4 digit 7 segment SPI display module
400 point solderless half breadboard
Push button
Buzzer
4 male to female breadboard cables
4 male to male breadboard cables

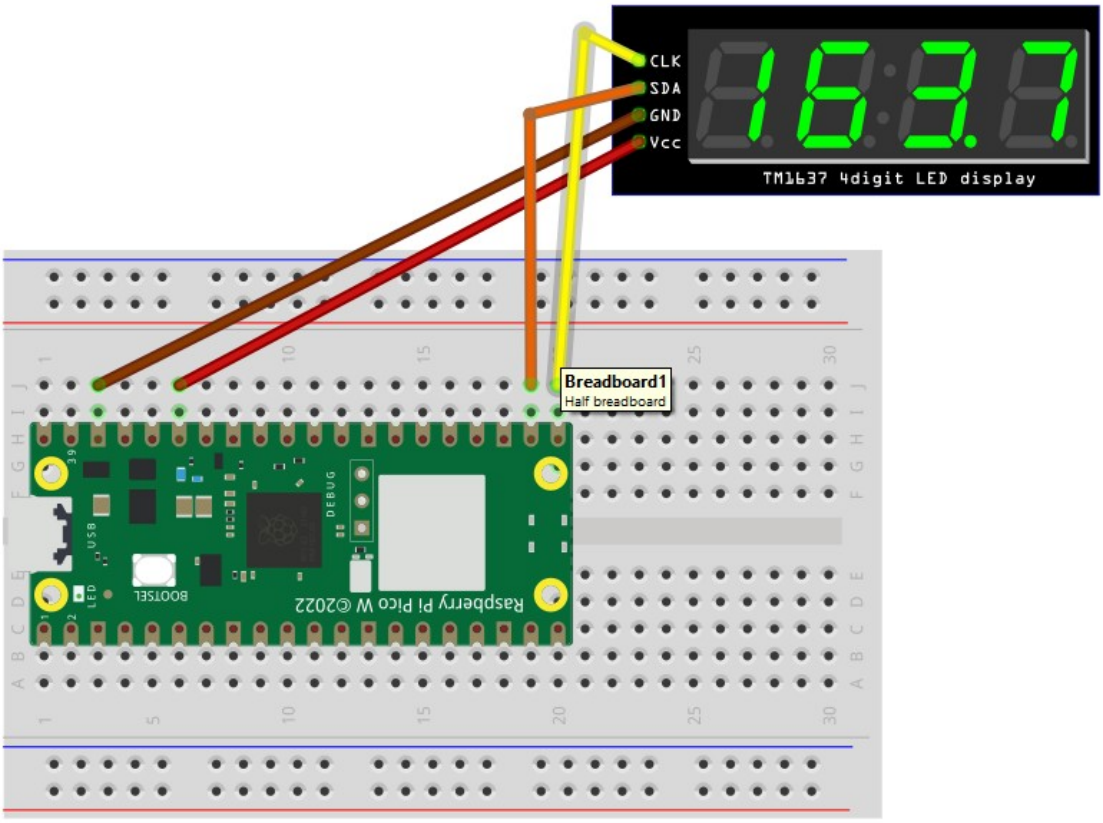


If this is the first time you've used your Raspberry Pi Pico you'll need to install the Thonny MicroPython development software on your Mac or PC and then upload the standard firmware for MicroPython to the board.

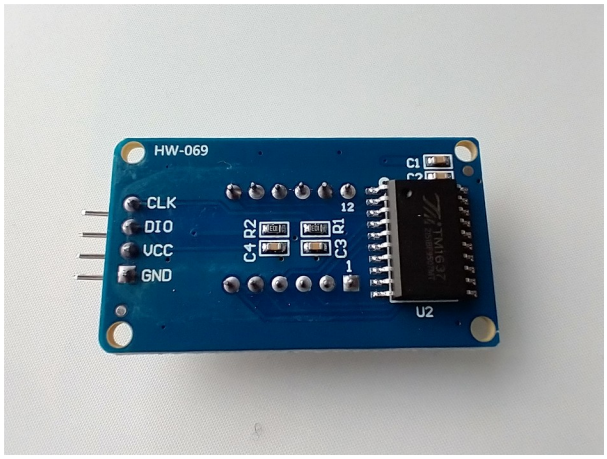
Thonny can be downloaded from https://thonny.org/

Then follow this Raspberry Pi guide: https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/3 to get the latest firmware on your Pi Pico board.

Insert your Pi Pico onto the breadboard and wire up the TM1637 module. The pins on the module are screen printed on the back of the PCB and are CLK, DIO, VCC, GND – they are in different positions to those shown in the graphic below and SDA becomes DIO on the real board.



So, wire CLK to the yellow wire position shown on the Pico. Wire SDA/DIO to the orange wire position on the Pi, VCC to the red wire position, and GND to the brown wire position shown. Note: If we've supplied different coloured wires, that doesn't matter, just make the correct connections.

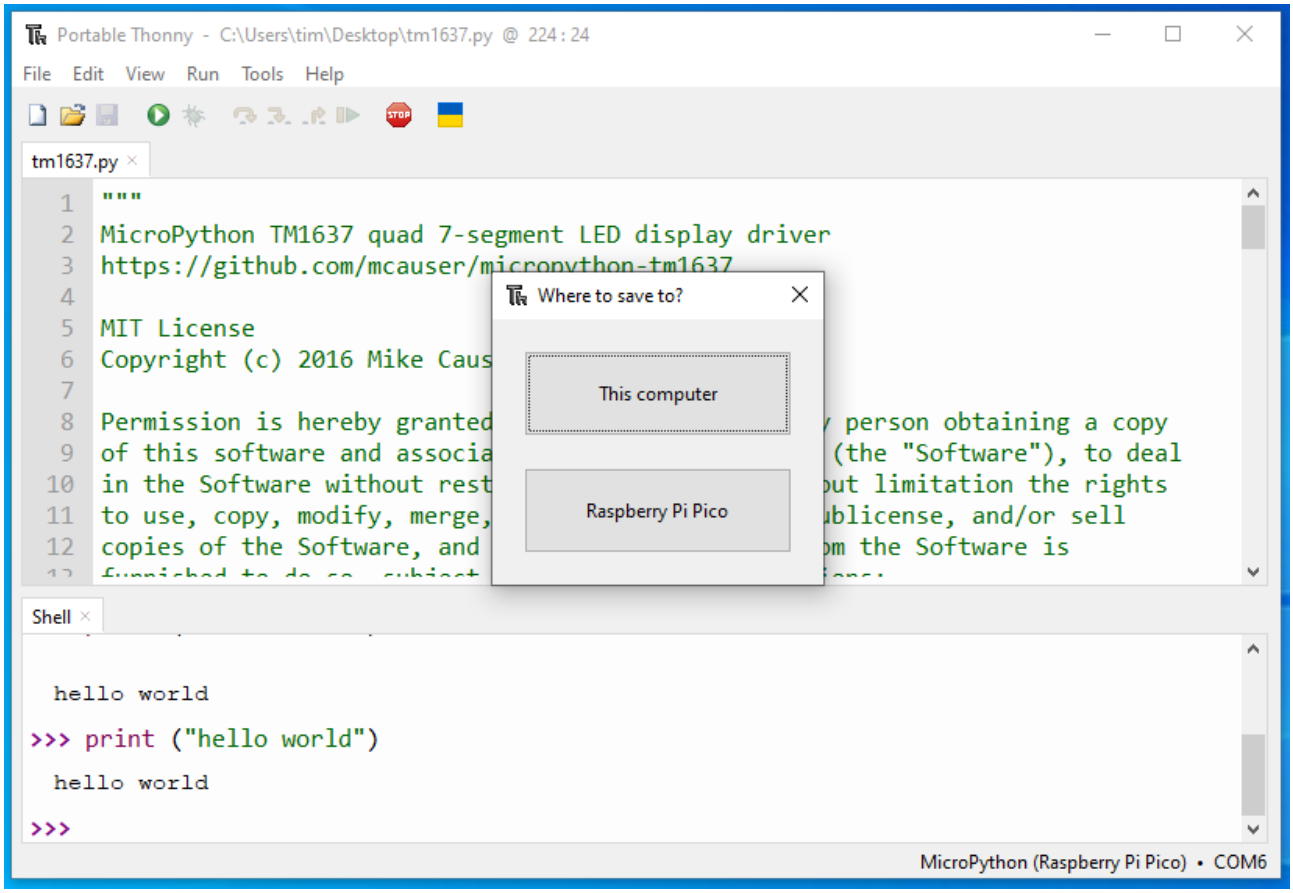


Once you've got the TM1637 LED module wired up correctly, we need to download the accompanying project files from https://www.securipi.co.uk/pi-pico-clock-kit.zip

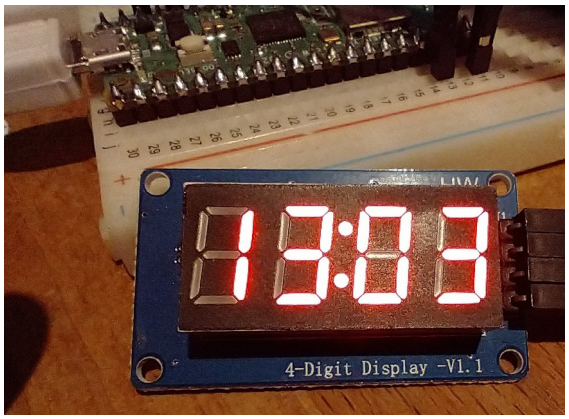
Extract the zip file and you'll see a number of Python project files with the file extension .py

We need to save the tm1637.py library file onto the Pi Pico's internal storage, so the other scripts can access it.

With Thonny running go to File → Open → This Computer → and locate the tm1637 python file. You should see it open in the project window. Next go to File → Save As → Raspberry Pi Pico.



Once that's done, unplug the Raspberry Pi Pico from the USB port and reinsert it. You should now be able to File → Open the clock.py script in Thonny, click the round green run button, and see the current time displayed on the LED module.



The contents of the clock.py file

```
import tm1637
import machine
import utime
mydisplay = tm1637.TM1637(clk=machine.Pin(16), dio=machine.Pin(17))

rtc=machine.RTC()

while True:
    timestamp=rtc.datetime()
    timestring="%02d%02d"%(timestamp[4:6])
    print(timestring + "\n")
    mydisplay.show(timestring, True)
    utime.sleep(60)
```

How the script works:

The first three lines import libraries that the script will use.

1- The tm1637 library controls the LED panel, it's the file we saved onto the Pi Pico's memory earlier. It allows the Pico to display numbers and some alphabet characters on the 4 digit display. It can adjust the brightness of the elements and it's possible to scroll text and numbers across the panel.

2- import machine. This library allows us to get the system time and date from the PC or Mac that the Pi Pico is attached to via the USB cable. The lines in the script where you see rtc stand for real time clock. We strip out the date and seconds values, so we're just left with the time in hours and minutes, and these get sent to the LED module with the mydisplay.show command, the True option makes the colon dots light up between the hours and minutes value on the LED module.

3- import utime. When you want a script to pause before moving on you use utime.sleep command with the number of seconds to pause for in brackets. In our example we pause for 60 seconds. If you wanted to pause for fractions of one second, you'd use the utime.sleep_ms command instead.

4- mydisplay – tells the tm1637 library which pins on the Pi Pico the TM1637 module is physically connected to.

5- rtc this is the real time clock value gleaned from the host PC or Mac.

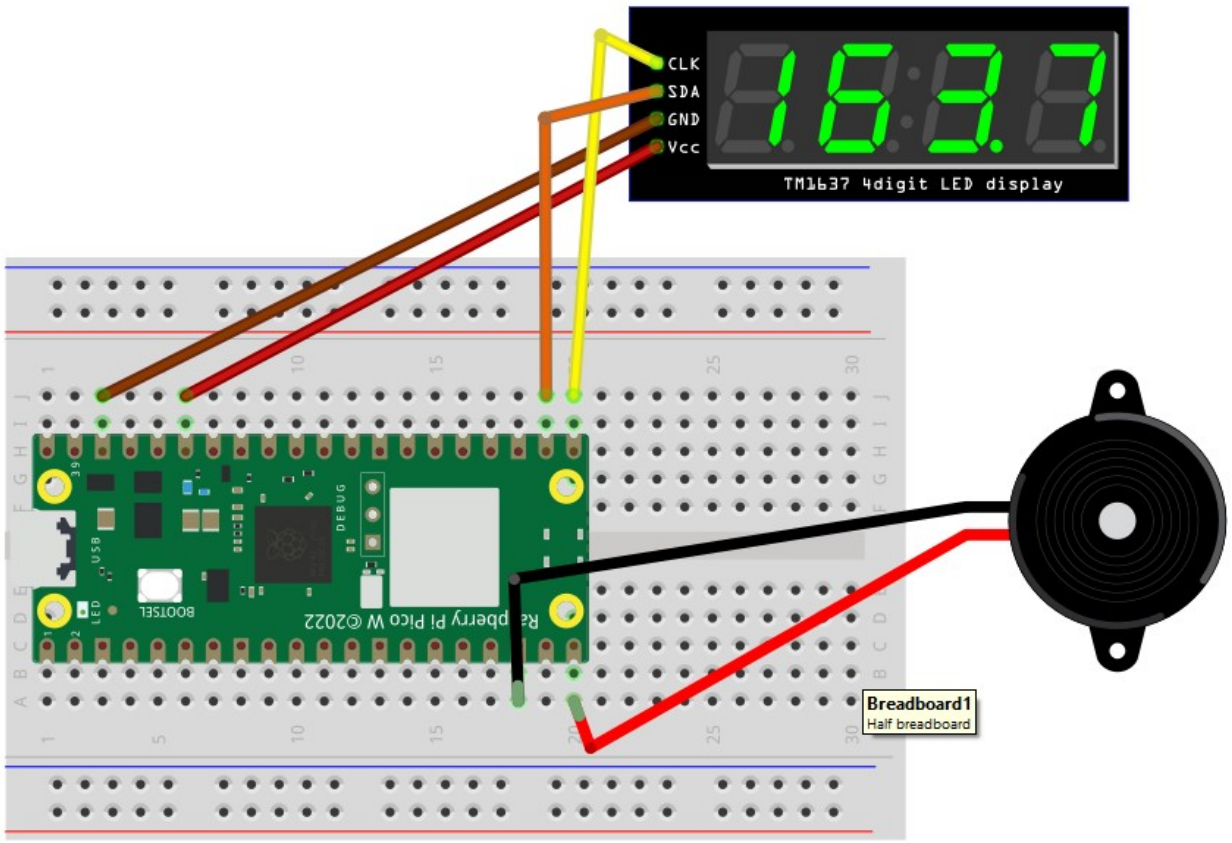
6- while loop. This goes around and around in a loop. It extracts the hour and minutes value from timestamp down into our timestring. It print this value to the console and also sends it to the LED module. The script then pauses for 60 seconds, and then does it all again.

This script scrolls a message across the display:

```
import tm1637
import utime
mydisplay = tm1637.TM1637(clk=machine.Pin(16), dio=machine.Pin(17))

while True:
    mydisplay.scroll("Hello Raspberry Pi Pico    ", delay=200)
```

Turn the clock into an alarm clock.

This wiring diagram and code add a buzzer circuit so you can set the Pi Pico to sound the buzzer when it reaches a certain time.



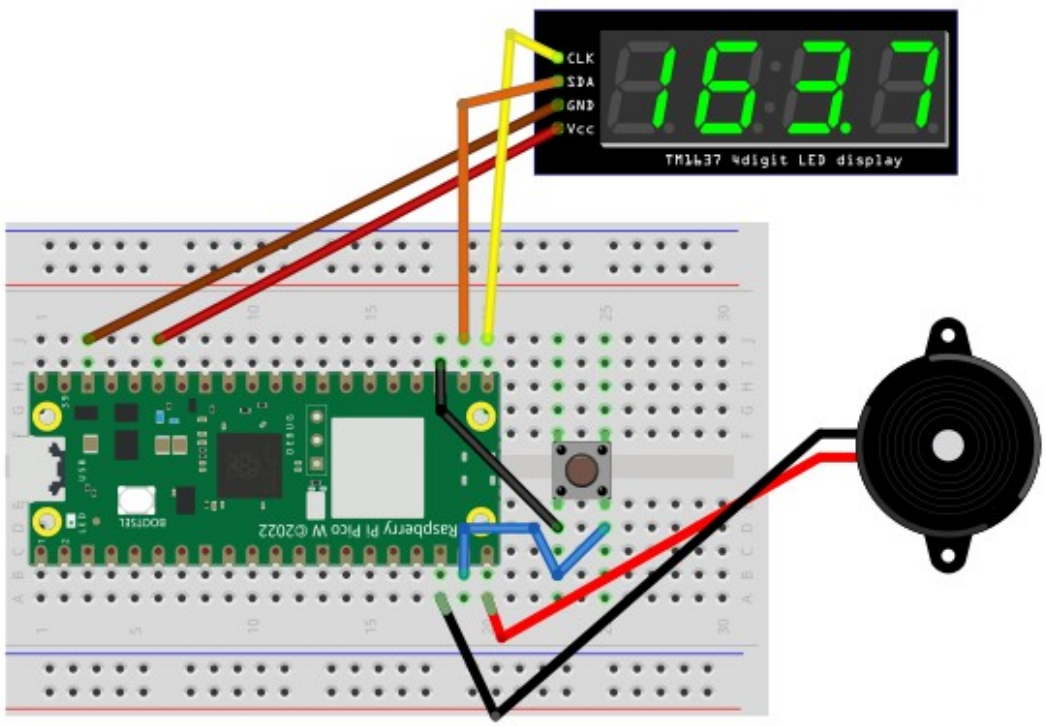
Just change the variable alarmtime to the time you want the buzzer to sound.

```
import tm1637
import machine
import utime
buzzer = machine.Pin(15, machine.Pin.OUT)
mydisplay = tm1637.TM1637(clk=machine.Pin(16), dio=machine.Pin(17))

rtc=machine.RTC()
alarmtime='1714'
while True:
    timestamp=rtc.datetime()
    timestring="%02d%02d"%(timestamp[4:6])
    print(timestring + "\n")
    mydisplay.show(timestring, True)
    if timestring == alarmtime:
       buzzer.high()
       utime.sleep(5)
       buzzer.low()
    utime.sleep(60)
```

This script does the same thing, but the display LEDs are dimmed, until the buzzer sounds:

```
import tm1637
import machine
import utime
buzzer = machine.Pin(15, machine.Pin.OUT)
mydisplay = tm1637.TM1637(clk=machine.Pin(16), dio=machine.Pin(17))

rtc=machine.RTC()
alarmtime='1714'
mydisplay.brightness(1)

while True:
    timestamp=rtc.datetime()
    timestring="%02d%02d"%(timestamp[4:6])
    print(timestring + "\n")
    mydisplay.show(timestring, True)
    if timestring == alarmtime:
       buzzer.high()
       mydisplay.brightness(7)
       utime.sleep(5)
       buzzer.low()
       mydisplay.brightness(1)
    utime.sleep(60)
```

Now setup the breadboard with the push button attached too. Next we'll make an alarm that stops when we push the button.



If we use the push button we can make an alarm that sounds for up to 60 seconds, unless we press

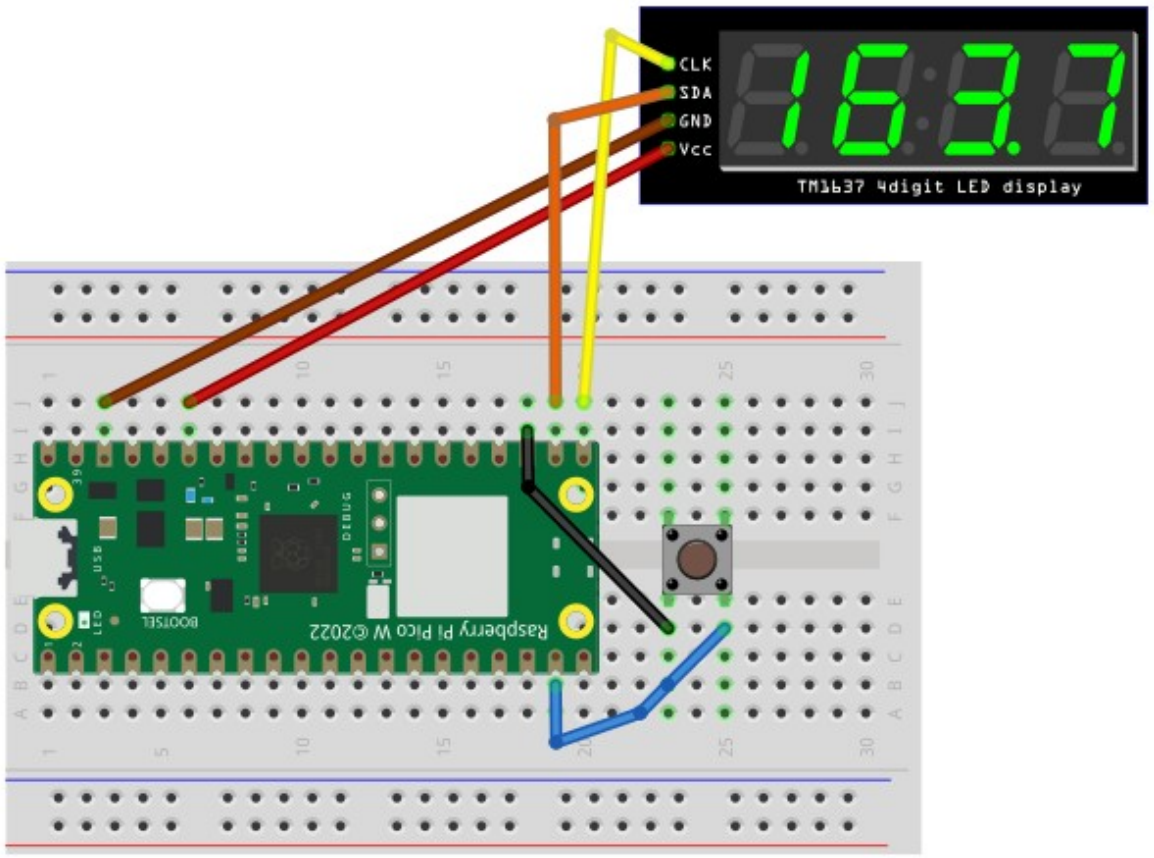
the button to stop it earlier. (clock-alarm-button.py)

```
import tm1637
import machine
import utime
buzzer = machine.Pin(15, machine.Pin.OUT)
button = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL_UP)
mydisplay = tm1637.TM1637(clk=machine.Pin(16), dio=machine.Pin(17))

rtc=machine.RTC()
alarmtime='1807'
mydisplay.brightness(1)

while True:
    timestamp=rtc.datetime()
    timestring="%02d%02d"%(timestamp[4:6])
    print(timestring + "\n")
    mydisplay.show(timestring, True)
    if timestring == alarmtime:
       buzzer.high()
       mydisplay.brightness(7)
       if button.value() == 0:
          buzzer.low()
          mydisplay.brightness(1)
          utime.sleep(61)
    utime.sleep_ms(333)
```

This script will turn the buzzer off, if it gets stuck on, and will also show whether the state of the button is 1 or 0. It should only read zero when pushed. It also dims the LED panel. Good for troubleshooting once you start to modify the scripts. (buzzoff.py)

```
import tm1637
import machine
import utime
buzzer = machine.Pin(15, machine.Pin.OUT)
button = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL_UP)
mydisplay = tm1637.TM1637(clk=machine.Pin(16), dio=machine.Pin(17))

mydisplay.brightness(1)
buzzer.low()
print(button.value())
```

Scrolling Name Badge



This script will print a scrolling message when you press the button (scroll-message-button.py)

```
import tm1637
import machine
import utime

mydisplay = tm1637.TM1637(clk=machine.Pin(16), dio=machine.Pin(17))
button = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL_UP)

while True:
     if button.value() == 0:
        mydisplay.scroll("  Hello Raspberry Pi Pico    ", delay=200)
        utime.sleep(3)
```

Clock3.py

This script scrolls the full date and time across the display with a 5 second break between scrolls.

```
import tm1637
import machine
import utime

mydisplay = tm1637.TM1637(clk=machine.Pin(16), dio=machine.Pin(17))
rtc=machine.RTC()

while True:
    timestamp=rtc.datetime()
    timestring="%04d-%02d-%02d %02d-%02d-%02d"%(timestamp[0:3] +
                                                timestamp[4:7])
    mydisplay.scroll(timestring, delay=200)
    utime.sleep(5)
```

https://www.securipi.co.uk