# Table of Contents

## Overview

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
</tbody>
</table>

## Hardware

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 40-Pin (A, B, B+ and Zero) Cobbler Plus Schematic</td>
<td>4</td>
</tr>
<tr>
<td>• 26-Pin (Raspberry Pi Rev 1 and Rev 2) Cobbler Schematic</td>
<td>5</td>
</tr>
</tbody>
</table>

## Pi Prep

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Update Your Pi to the Latest Raspbian</td>
<td>5</td>
</tr>
<tr>
<td>• Install adafruit-blinka</td>
<td>6</td>
</tr>
<tr>
<td>• Enable 1-Wire Interface</td>
<td>6</td>
</tr>
<tr>
<td>• Enable I2C Interface</td>
<td>7</td>
</tr>
<tr>
<td>• Test the I2C Bus</td>
<td>7</td>
</tr>
<tr>
<td>• Install the HT16K33 Library</td>
<td>8</td>
</tr>
</tbody>
</table>

## Software

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>• CircuitPython Code</td>
<td>8</td>
</tr>
<tr>
<td>• Download the Code</td>
<td>10</td>
</tr>
<tr>
<td>• Run the Code</td>
<td>10</td>
</tr>
<tr>
<td>• Changing Temperature Units</td>
<td>10</td>
</tr>
</tbody>
</table>
Overview

This project combines a whole heap of modules to enable a Raspberry Pi to power a large 1.2 inch 4 digit 7 segment display. A small switch switches the display between showing the temperature and the current time. The project uses a real-time clock (RTC) to ensure that the Pi always has the correct time, even if it is not connected to the Internet.

Hardware

This guide works with all versions of Raspberry Pi except the compute modules which do not have a header. Older Raspberry Pi revisions 1 and 2 will require a 26-pin Cobbler and newer versions make use of the Cobbler Plus 40-pin GPIO connector. We provide wiring diagrams for both 40-pin and 26-pin below.
The RTC will allow a Raspberry Pi to know the time, even when not connected to the Internet. As such it is not essential to this project if your Raspberry Pi is going to have an Internet connection.

The diagrams show GND connections in blue, +5V in red and +3.3V in purple.

The I2C bus connections use orange wire for SDA and yellow for SCA.

40-Pin (A, B, B+ and Zero) Cobbler Plus Schematic
You may also like to look at separate tutorials for the RTC [http://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi](https://adafru.it/aPm) and temperature sensor [http://learn.adafruit.com/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing](https://adafru.it/aWY).

When everything is assembled, you can connect the ribbon cable to the GPIO connector. Remember to have the red or white band of the ribbon cable towards the SD card on the Raspberry Pi.

---

**Pi Prep**

**Update Your Pi to the Latest Raspbian**

Your Pi will need to be running the latest version of Raspbian. This tutorial was written using Raspbian Buster (July 2019). Checkout our guide for [Preparing an SD Card for your Raspberry Pi](https://adafru.it/dDL) if you have not done so already. After the installation is complete be sure and run the following commands to make confirm your installation packages are up to date.

```
sudo apt-get update -y
sudo apt-get upgrade -y
sudo apt-get dist-upgrade -y
```
Install adafruit-blinka

Run the following command to install the adafruit_blinka CircuitPython Libraries.

```
pip3 install adafruit-blinka
```

Enable 1-Wire Interface

You'll need to start by enabling support for the DS18B20 1-Wire subsystem.

Visit this page for instructions on using raspi-config to enable 1-Wire support. (https://adafruit.io/sd4)

```
sudo raspi-config
&lt;Interfacing Options&gt;
&lt;1-Wire&gt; Enable/Disable
&lt;Back&gt;
&lt;Finish&gt;
```
Enable I2C Interface

Our clock and 7-segment display work using I2C bus which the Raspberry Pi supports, but it needs to be enabled. This page has step by step instructions (https://adafru.it/dEO). Enabling I2C is almost an identical process as enabling 1-Wire except that a reboot is required and we want to make sure to install the optional packages to scan the bus.

```
sudo raspi-config
<Interfacing Options>
<I2C> Enable/Disable
<Enable>
<Back>
<Finish>
sudo reboot
```

Test the I2C Bus

```
lsmod | grep -i i2c
```

Now that our Pi has rebooted we can confirm that the correct I2C modules have loaded with the following command.

```
$ lsmod | grep -i i2c
i2c_bcm2835  16384  0
i2c_dev     16384  0
$ i2cdetect -y 1
```

```
sudo i2cdetect -y 1
```
This shows that two I2C addresses are in use – 0x68 and 0x70.

Note that if you are using one of the very first Raspberry Pis (a 256MB Raspberry Pi Model B) then you will need to change the last digit of the command port 1 to port 0.

Install the HT16K33 Library

The software for this project uses the Adafruit code for driving the 7 segment display.

```
$ pip3 install adafruit-circuitpython-ht16k33
```

Software

CircuitPython Code

The main loop simply checks the position of the switch and then either displays the temperature or the time.

```
import glob
import time
import datetime
from adafruit_ht16k33 import segments
import board
import busio
import digitalio
```
switch_pin = digitalio.DigitalInOut(board.D18)
switch_pin.direction = digitalio.Direction.INPUT
switch_pin.pull = digitalio.Pull.UP

# Create the I2C interface.
i2c = busio.I2C(board.SCL, board.SDA)

# Create the LED segment class.
# This creates a 7 segment 4 character display:
display = segments.Seg7x4(i2c)

base_dir = '/sys/bus/w1/devices/
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c, temp_f

def display_temp():
    temp = read_temp()[1]  # F
    # temp = read_temp()[0]  # C
    display.print(int(temp))

def display_time():
    now = datetime.datetime.now()
    hour = now.hour
    minute = now.minute
    second = now.second
    clock = int('%i%i' % (hour,minute))  # concat hour + minute
    display.print(clock)

    # Toggle colon when displaying time
    if second % 2 == 0:
        display.print(':')  # Enable colon every other second
    else:
        display.print(';')  # Turn off colon

display.fill(0)

while True:
    if not switch_pin.value:
        display_temp()
    else:
        display_time()
    time.sleep(0.5)

We can easily copy this code onto our Pi’s home directory using the ‘wget’ command and run it using the following python syntax.
Download the Code

```bash
cd
wget https://raw.githubusercontent.com/adafruit/Adafruit_Learning_System_Guides/master/Large-Pi-Based-Thermometer-Clock/thermo_clock.py
```

Run the Code

The `thermo_clock.py` script will not output anything when run, but what you should see is the time or temperature on the large 7-segment display. Using the slide switch should switch modes between time and temperature.

```
python3 ./thermo_clock.py
```

Changing Temperature Units

The temperature display can easily be changed from displaying degrees F to degrees C by swapping comment line in the code.

```python
#temp = int(read_temp()[1]) # F
#temp = int(read_temp()[0]) # C
```

To swap, just move the `#` in front of the line that does not apply.