## Guide Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview</strong></td>
<td>3</td>
</tr>
<tr>
<td><strong>Parts</strong></td>
<td>4</td>
</tr>
<tr>
<td>- Raspberry Pi 3 - Model B - ARMv8 with 1G RAM</td>
<td>4</td>
</tr>
<tr>
<td>- Adafruit 1.2&quot; 4-Digit 7-Segment Display w/I2C Backpack - Red</td>
<td>4</td>
</tr>
<tr>
<td>- Adafruit DS1307 Real Time Clock Assembled Breakout Board</td>
<td>4</td>
</tr>
<tr>
<td>- DS18B20 Digital temperature sensor + extras</td>
<td>5</td>
</tr>
<tr>
<td>- Adafruit Pi Cobbler + Kit- Breakout Cable for Pi B+/A+/Pi 2/Pi 3</td>
<td>5</td>
</tr>
<tr>
<td>- Premium Male/Male Jumper Wires - 20 x 3&quot; (75mm)</td>
<td>5</td>
</tr>
<tr>
<td>- Half-size breadboard</td>
<td>5</td>
</tr>
<tr>
<td><strong>Hardware</strong></td>
<td>7</td>
</tr>
<tr>
<td><strong>Software</strong></td>
<td>11</td>
</tr>
<tr>
<td>- Add 1-Wire to your Pi</td>
<td>11</td>
</tr>
<tr>
<td>- Install LED Backpack software</td>
<td>11</td>
</tr>
<tr>
<td><strong>Configure and Test</strong></td>
<td>14</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.
Parts

Raspberry Pi 3 - Model B - ARMv8 with 1G RAM

$35.00
IN STOCK
ADD TO CART

Adafruit 1.2" 4-Digit 7-Segment Display w/I2C Backpack - Red

$17.50
IN STOCK
ADD TO CART

Adafruit DS1307 Real Time Clock Assembled Breakout Board

$7.50
IN STOCK
ADD TO CART
DS18B20 Digital temperature sensor + extras

$3.95
IN STOCK
ADD TO CART

Adafruit Pi Cobbler + Kit- Breakout Cable for Pi B+/A+/Pi 2/Pi 3

$6.50
OUT OF STOCK
OUT OF STOCK

Premium Male/Male Jumper Wires - 20 x 3" (75mm)

$1.95
IN STOCK
ADD TO CART

Half-size breadboard

$5.00
OUT OF STOCK
OUT OF STOCK
Hardware

This project uses a lot of modules and there are a lot of connections to be made. The Pi Cobbler, LED Display and RTC are all supplied as kits that must be assembled.

You can follow the instructions on the product pages for the components, with one exception, when you come to assemble the real-time clock module, do not solder the two resistors R1 and R2. (Or if using the assembled version, cut the two jumper traces on the bottom) They are not needed when the module is used with a Raspberry Pi, as it has its own pull-up resistors.

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.

Having assembled the modules, you can then fit them onto the breadboard and wire them up.

You can just about cram all the components onto a single half-sized breadboard, but it is easier if you use two clipped together side to side.

The first step is to place all the components as shown below. Make sure everything is the right way around, especially the temperature sensor.
The diagram above shows the layout for a smaller 0.56 inch display which will work just as well, however, in the final design, you will notice that just underneath the + connection on the display, there is an orange jumper wire connected to +3V. The larger display needs this extra connection to set the logic level to use, whereas the smaller 0.56 inch module does not. Otherwise, the connections are identical and either module can be used.

The next step is to make all the power connections. The diagram show GND connections in blue, +5V in red and +3.3V in purple.
Now let's add the leads for the I2C bus connections. I have used orange wires for SDA and yellow for SCA.
Finally we need the connections to the switch and the temperature sensor output, which are made using green wires.

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://adafruit.it/aPm) and temperature sensor [http://learn.adafruit.com/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing](https://adafruit.it/aWY)

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

Add 1-Wire to your Pi

You'll need to start by installing support for the DS18B20 1-Wire subsystem. Visit this page to do so, you'll have to add an overlay to config.txt (https://adafru.it/sd4)

Install LED Backpack software

The software for this project uses the Adafruit code for driving the 7 segment display. So, you should start by downloading this onto your Raspberry Pi.

```bash
sudo apt-get update
sudo apt-get install -y git build-essential python-dev python-smbus python-imaging python-pip python-pil
git clone https://github.com/adafruit/Adafruit_Python_LED_Backpack.git
```

We are going to place the program for this into a file inside the Adafruit code, so issue the following commands to get to the right directory.

```bash
cd Adafruit_Python_LED_Backpack
sudo python setup.py install
```

Now create a new file to edit, by typing the command:

```
nano thermo_clock.py
```

Then paste the program below into the editor.

```python
import os
import glob
import time
import datetime
from Adafruit_LED_Backpack import SevenSegment
import RPi.GPIO as io
import subprocess

io.setmode(io.BCM)
switch_pin = 18
io.setup(switch_pin, io.IN)

segment = SevenSegment.SevenSegment(address=0x70)
# Initialize the display. Must be called once before using the display.
segment.begin()

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

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

def read_temp_raw():
def read_temp_raw():
    catdata = subprocess.Popen(['cat', device_file], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    out, err = catdata.communicate()
    out_decode = out.decode('utf-8')
    lines = out_decode.split('
')
    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():
    segment.set_colon(False)
    temp = int(read_temp()[1]) # F
    # temp = int(read_temp()[0]) # C
    sign = (temp < 0)
    temp = abs(temp)
    digit_1 = temp % 10
    temp = temp / 10
    digit_2 = temp % 10
    temp = temp / 10
    digit_3 = temp % 10
    if sign:
        segment.set_digit_raw(0, 0x40)  # - sign
        if digit_3 > 0:
            segment.set_digit(0, digit_3)  # Hundreds
        else:
            segment.set_digit_raw(0, 0)
    if digit_2 > 0:
        segment.set_digit(1, digit_2)  # Tens
    else:
        segment.set_digit_raw(1, 0)
    segment.set_digit(2, digit_1)  # Ones
    segment.set_digit_raw(3, 0x71)  # Temp units letter
    # segment.set_digit_raw(3, 0x39) # C

def display_time():
    now = datetime.datetime.now()
    hour = now.hour
    minute = now.minute
    second = now.second
    # Set hours
    segment.set_digit(0, int(hour / 10))  # Tens
    segment.set_digit(1, hour % 10)  # Ones
    # Set minutes
    segment.set_digit(2, int(minute / 10))  # Tens
    segment.set_digit(3, minute % 10)  # Ones
    # Toggle colon
    segment.set_colon(second % 2)  # Toggle colon at 1Hz

while True:
segment.clear()
if io.input(switch_pin):
    display_temp()
else:
    display_time()
segment.write_display()
time.sleep(0.5)

Python is indent/whitespace sensitive! So make sure you get all the tabs/indents as shown above. Press CTRL-X and then Y to save the file.

The code is really a merger of code from a number of other projects. The code for displaying the time comes from the file ex_7segment_clock.py in the same directory as this program. The code for reading the temperature is taken from this tutorial http://learn.adafruit.com/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing (https://adafru.it/aWY)

The main loop simply checks the position of the switch and then either displays the temperature or the time. You can also find information on using switches with the GPIO library here: http://learn.adafruit.com/adafruits-raspberry-pi-lesson-12-sensing-movement (https://adafru.it/c67)
Configure and Test

We are now ready to run the program, but first if you are using the RTC, you need to set it. You can find instructions to do this here: [http://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi](http://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi)

To run the program type the following command:

```
sudo python thermo_clock.py
```

The temperature display can easily be changed from displaying degrees F to degrees C by swapping over a few lines in the code.

```
segment.set_digit_raw(3, 0x71) #F        # Temp units letter
#segment.set_digit_raw(3, 0x39) #C
```  

and

```
 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.