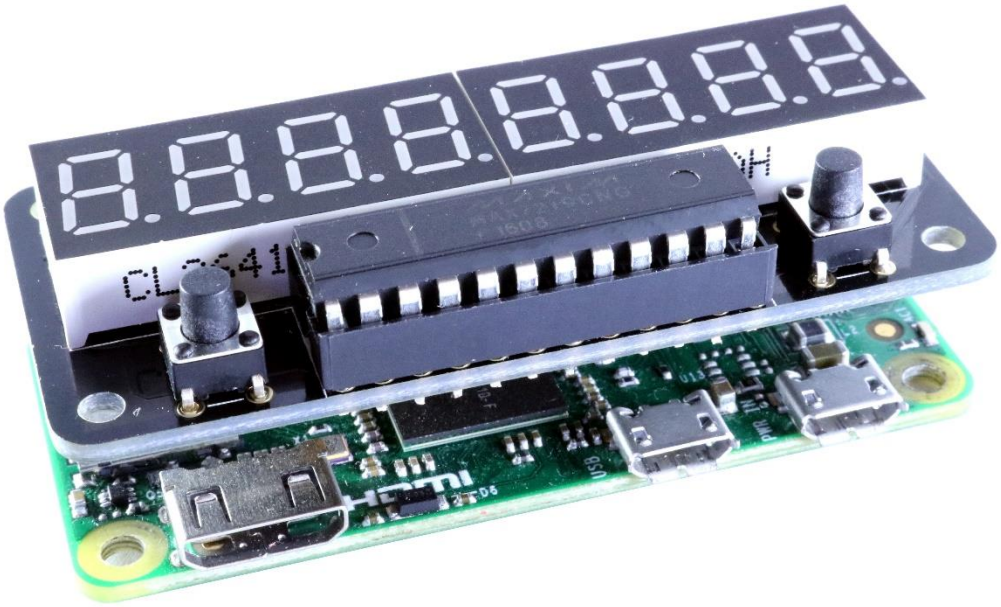
ZEROSEG

USER GUIDE



Raspberry Pi Seven-Segment Display Add-on Board

User Guide and Information

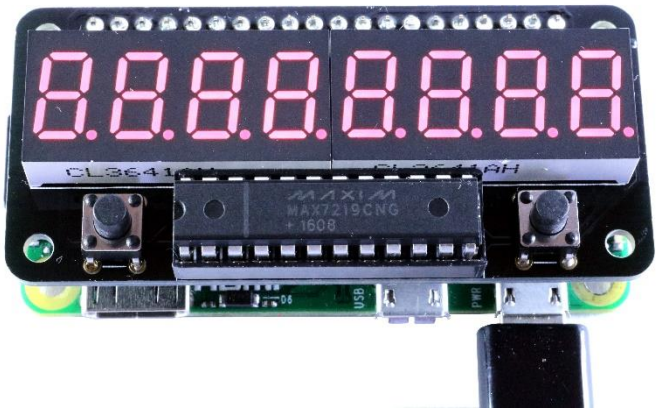
Product Page: ThePiHut.com/ZeroSeg

GitHub Library: https://github.com/AverageMaker/ZeroSeg

GUIDE CONTENTS

INTRODUCTION

Congratulations on your purchase of the ZeroSeg 8-character 7-segment display add-on board for the Raspberry Pi!



The ZeroSeg contains two (4-character) 7-segment displays, giving you the ability to display 8-digit data on a tiny Pi Zero sized add-on board. It also holds 2 tactile buttons for controlling data, brightness or any other element of your project.

The ZeroSeg works with any 40 GPIO pin Raspberry Pi – not just the Pi Zero - and is controlled by a MAX7219CNG integrated circuit, which manages the display of each LED segment, requiring very few GPIO pins to run the board.

This board's circuit is wired in the exact same way as generic 7-segment modules, allowing the use of existing code and libraries to easily create Pi Zero projects with 8-character displays.

The code and library provided in this guide (via GitHub) was originally cloned from Richard Hull's excellent open source MAX7219 library. This library, and the provided example scripts, makes it easy to display data along the ZeroSeg's twin 7-segment displays with just a few simple lines of code *(Note: It has since been replaced with the luma.led_matrix library, however our clone repository continues to work well thanks to the fantastic Raspberry Pi community).*

This user guide will take you through each step to get your ZeroSeg up and running, from initial assembly & soldering, to setup and library install, example scripts, code and frequently asked questions.

DESIGN FEATURES

- Eight character 7-segment display for displaying data on your Pi
- 2 tactile buttons to include in your projects and control the display
- Easy-to-learn programming with lots of code examples available to download via a GitHub library
- The board has been kept as symmetrical as possible to make sure your project looks great
- Small components are fitted to the underside of the board, with solder joints hidden under the displays, to give the board a clean look
- Tactile switches are the same height as the displays and IC ensuring ease of use
- Mounting holes have been included to attach to HATs or other add-on boards
- We've used ENIG (gold) plating on the PCB to protect against oxidation and offer a good soldered contact.
- The MAX7219CNG chip can be replaced in the unlikely event that it is damaged
- The circuit is designed in the same way as generic units, allowing existing code and libraries to be used
- Kit form – enjoy soldering and assembling the board yourself. Yes, **YOU** made it!

KIT CONTENTS

Like many Raspberry Pi add-on boards, the ZeroSeg is sold as a kit which requires soldering. The following parts should be inside your kit bag:

1x ZeroSeg PCB

1x 40-pin GPIO header

1x MAX7219CNG IC

1x 24-pin IC socket

2x 4-character 7-segment displays

2x Tactile switches

1x 100nF capacitor (labelled '104')

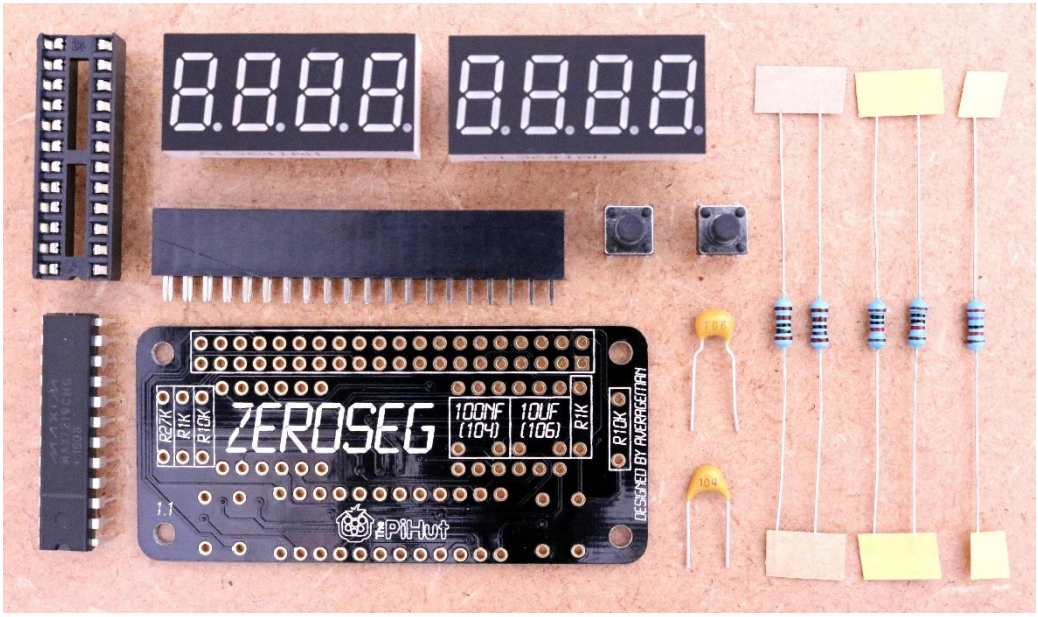
1x 10uF capacitor (labelled '106')

2x 10k resistors (colour bands: brown/black/black/red/brown)

2x 1k resistors (colour bands: brown/black/black/brown/brown)

1x 27k resistor (colour bands: red/violet/black/red/brown)

NOTE: *On the 27k resistor, the violet band can be hard to distinguish from brown, however it's the only resistor in the kit to start with a red band, making it easy to identify.*



ASSEMBLY

Assembling the ZeroSeg involves soldering the provided components to the PCB.

TOOLS REQUIRED

- Soldering Iron
- Solder
- Wire/Side cutters
- Protective eyewear
- Optional: Blu-Tack

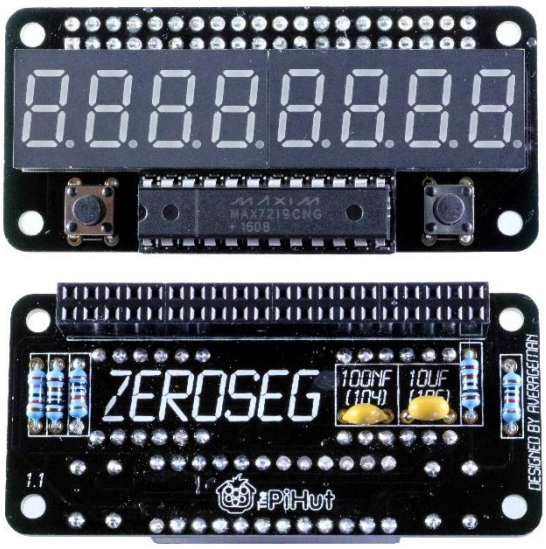
BEFORE YOU START

- Make sure you have all the parts required
- Read all steps through before starting, and take a good look at each image to confirm component positions
- Follow each step exactly as documented
- Always wear protective eyewear when soldering, and keep your room well ventilated

ASSEMBLY STEPS

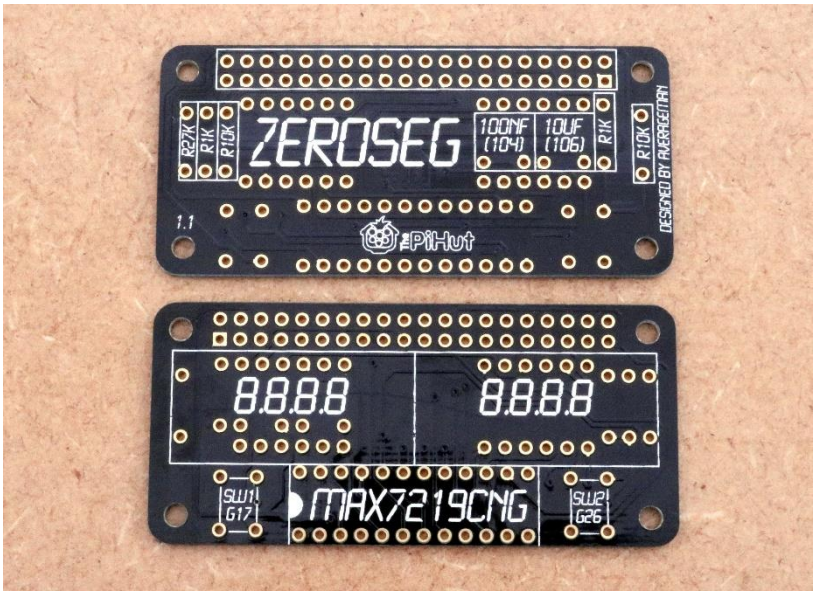
1. **Familiarise Yourself with the Completed Board**

   It helps to see the completed board, front and back, before you start – so that you get an initial idea of where each component fits:



2. **Identify PCB Sides**

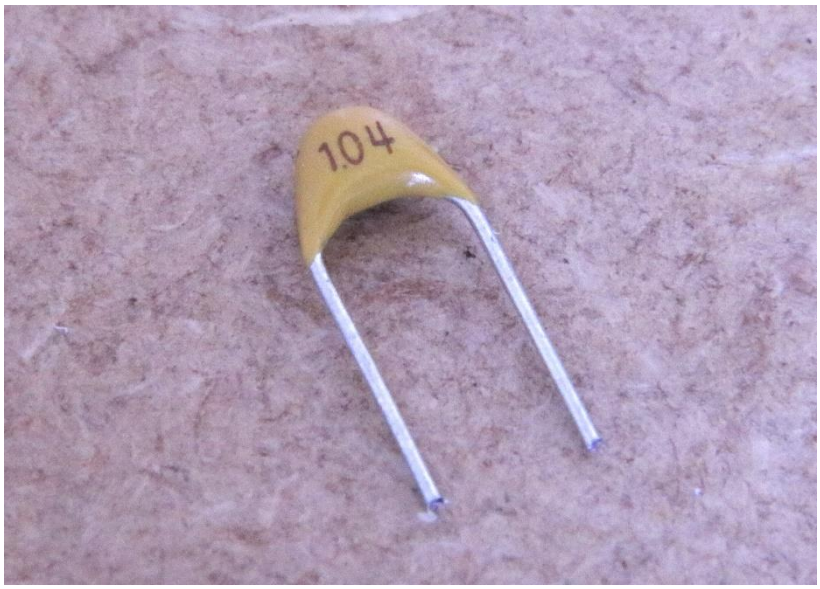
   First, let's identify the front and rear of the board. The front (top) is the side with 'MAX7219CNG' printed on it. The rear (underside) has the Pi Hut logo printed.



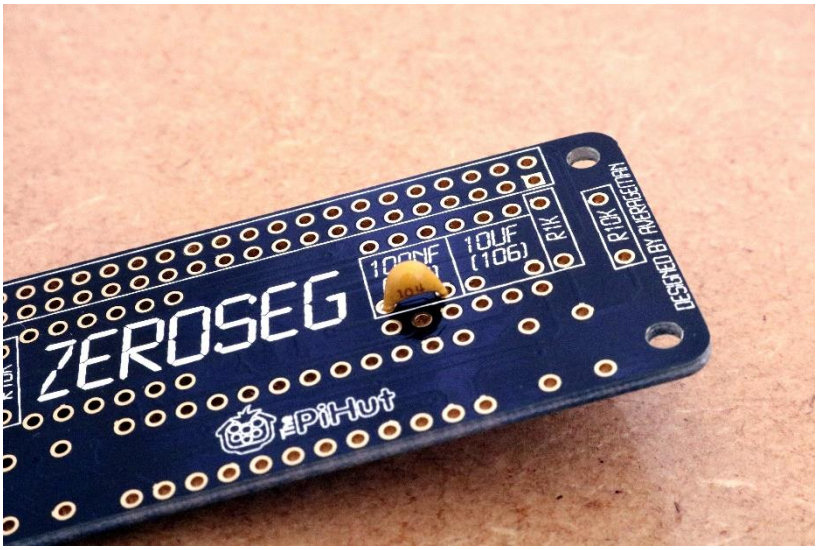
   We need to solder the parts that are fitted to the underside of the board first, as the displays and chips will cover those soldering points once fitted.

3. **Solder Capacitor 104**

Start with the capacitor labelled '104' - this is printed on the yellow body:



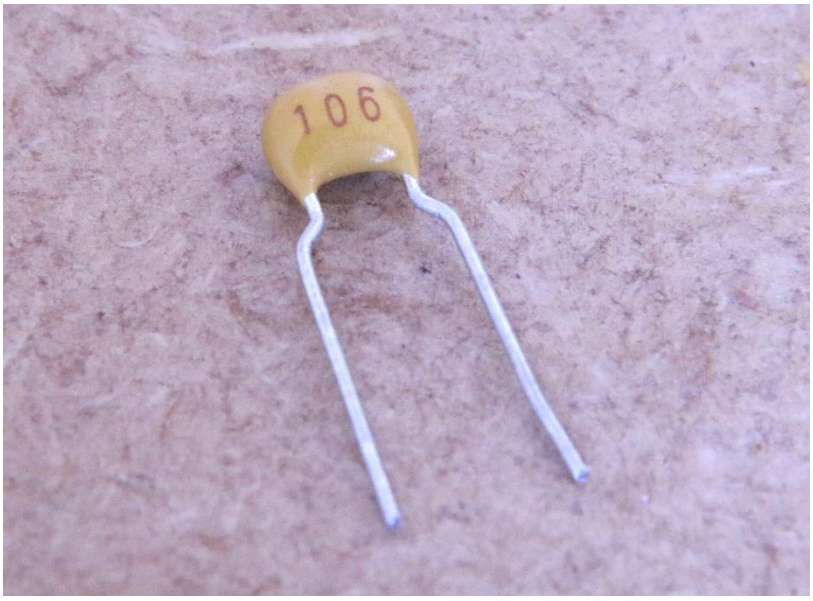
This capacitor fits inside the box with '100NF (104)' printed on it, with the yellow 'head' of the capacitor showing on the underside of the PCB:



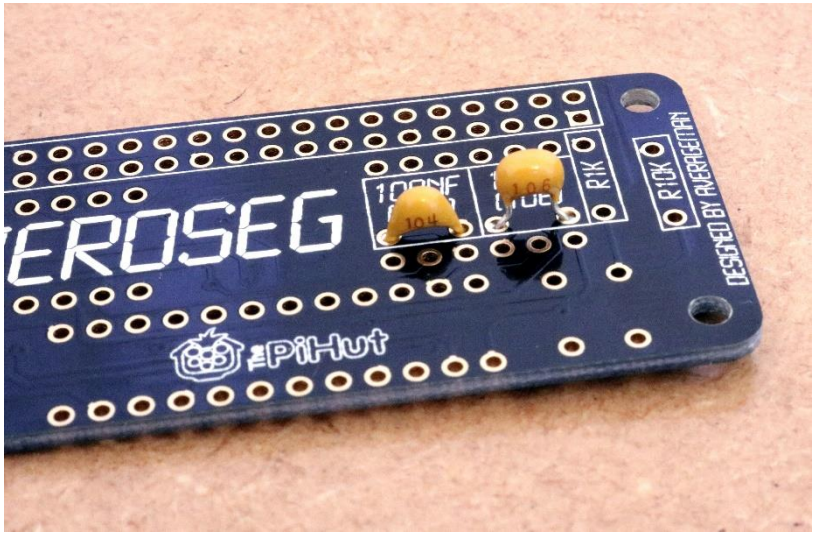
Push the legs through the holes within the printed box – it doesn't matter which way you fit this as these capacitors have no set polarity. Solder the legs from the other side and snip away any remaining leg.

4. **Solder Capacitor 106**

Next, solder the other capacitor labelled '106'- this is also printed to the yellow body:



This capacitor needs to be fitted inside the box with '10UF (106)' printed on it, with the yellow 'head' of the capacitor showing on the underside of the board:



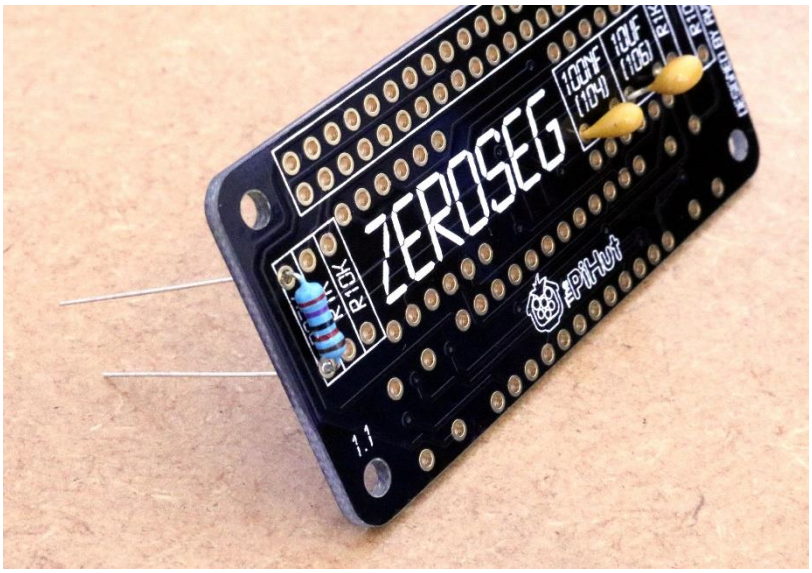
Push the legs through the holes within the printed box, then solder the legs from the other side and snip away any remaining leg.

5. Solder the 27K Resistor

Next fit the 27K resistor. This resistor should be in a single strip in the kit, and has the colours **red**/**violet**/**black**/**red**/**brown** on the body.

The violet band looks brown in certain lights – just remember that this is the only resistor that starts with a red colour band.

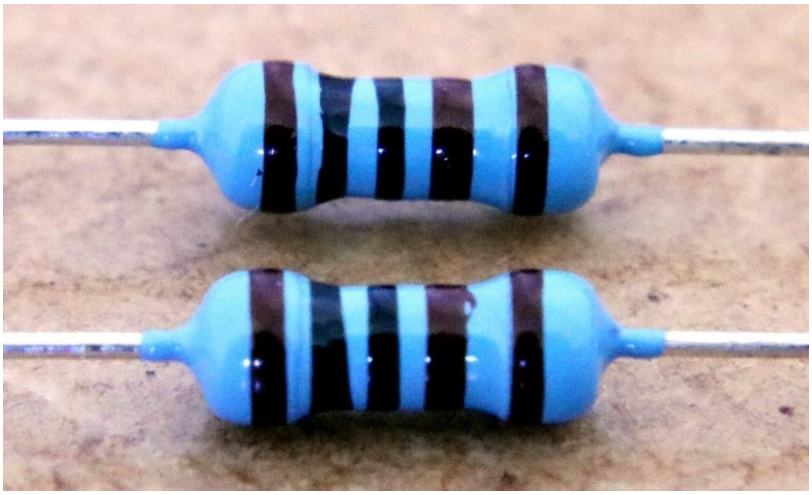
Find the box on the underside of the board with 'R27K' printed inside it. Bend the resistor legs and fit them through the holes inside this box. Make sure the body of the resistor is on the underside of the PCB, and solder in place.



Snip off any remaining leg after soldering and checking the fit.

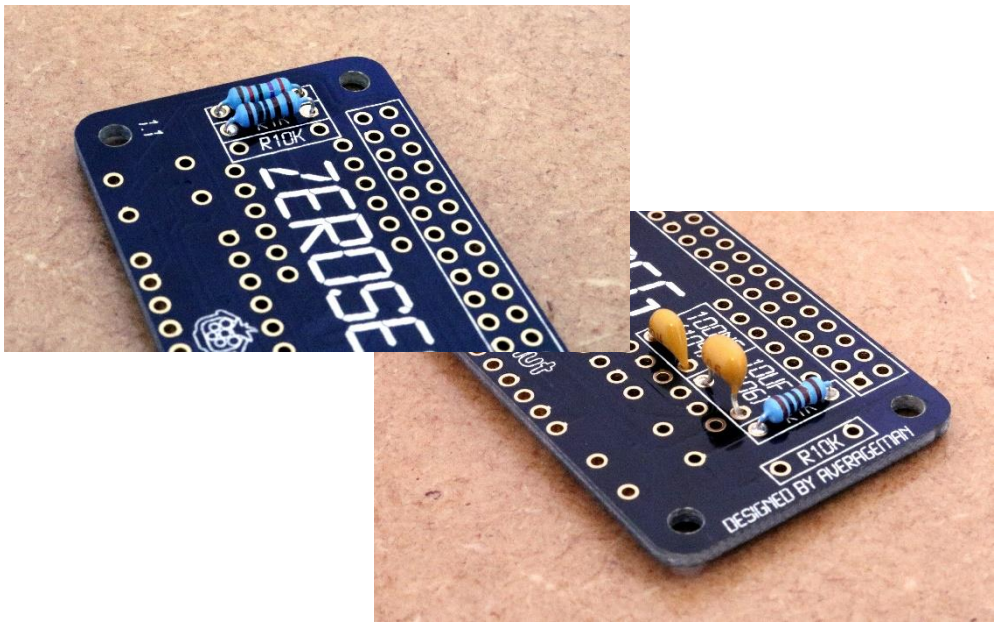
6. Solder 1K Resistors

Now fit the 1k resistors. These resistors have the colour bands **brown/black/black/brown/brown**, and are easy to identify as these resistors are the only ones in the kit with no red colour bands:



There are 2 printed boxes for the 1k resistors, one each side of the PCB.

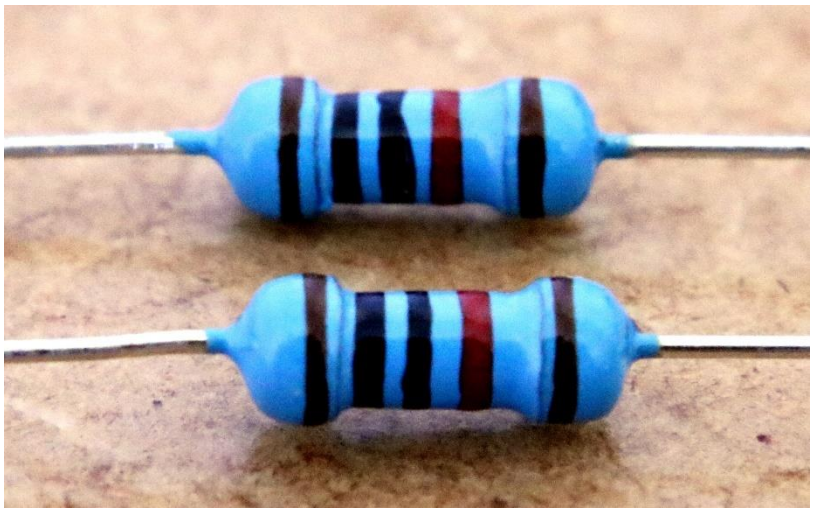
As before, bend the resistor legs and fit them through the holes inside each box. Make sure the body of the resistors are on the underside of the PCB, and solder them in place.



Snip off any remaining leg after soldering and checking the fit.

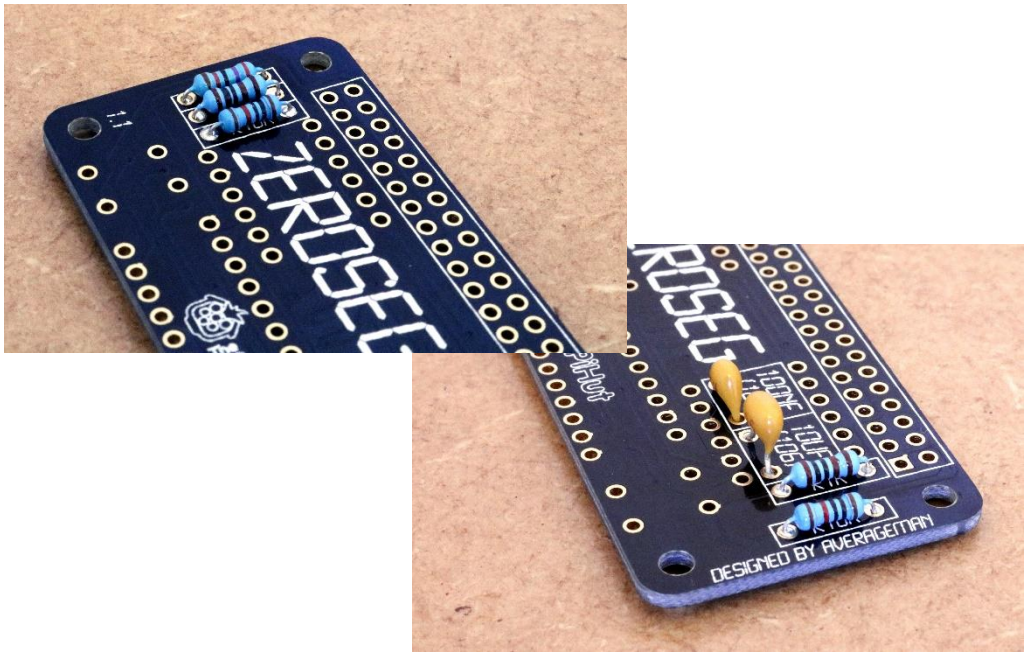
7. **Solder 10K Resistors**

The last resistors to fit are the 10k resistors, which have the colour bands **brown/black/black/red/brown**.



There are 2 printed boxes for the 10k resistors, one each side of the PCB.

Bend the resistor legs and fit them through the holes inside each box. Make sure the body of the resistors are on the underside of the PCB, and solder them in place.

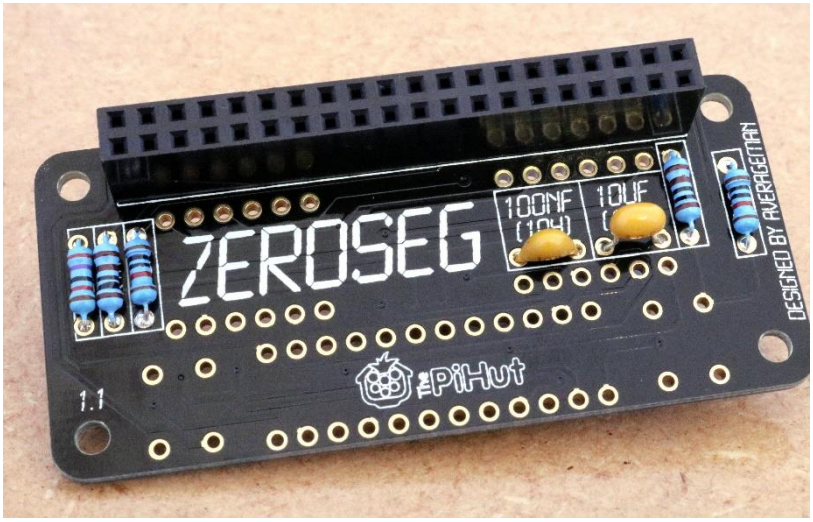


Snip off any remaining leg after soldering and checking the fit.

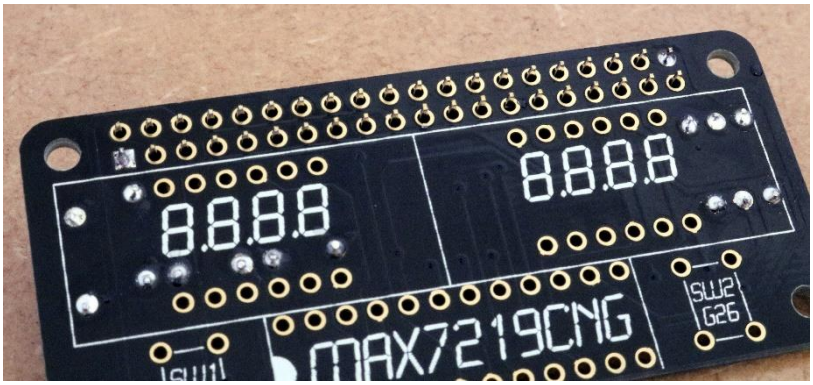
8. **Solder the 40-pin Header**

Lastly on this side of the board, we will fit the 40-pin GPIO header, as it's easier to solder this now rather than once the displays are fitted, and avoids any potential damage to the visible elements of the board from soldering.

Push the GPIO header pins through the large rectangular printed section at the top of the board, so that the plastic body is the same side as the underside of the board.



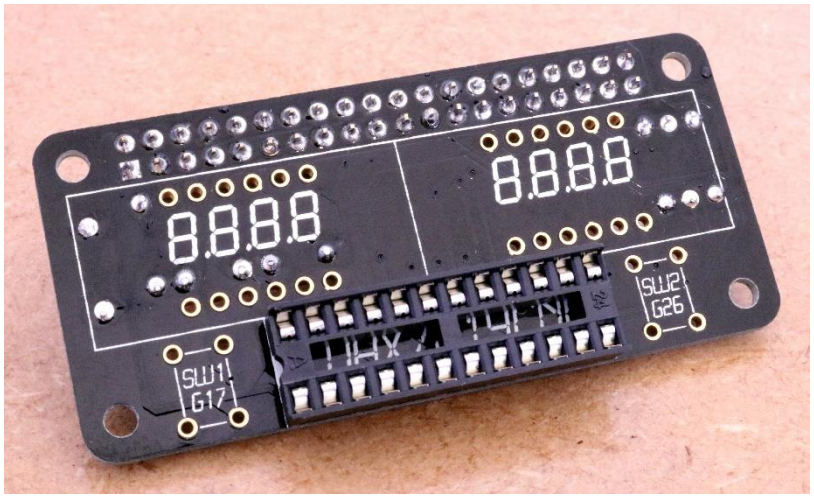
Solder a corner pin, and then an opposite corner, then check the fit. If the header isn't flat or straight, re-heat the joints and carefully move the header until it's in the right place.



Once happy with the header's position, solder the remaining pins.

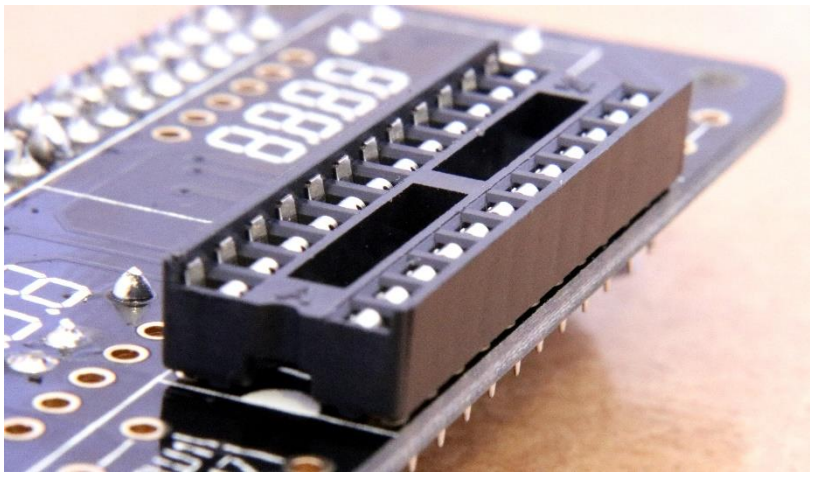
9. **Solder the IC Socket**

Now we will fit the IC socket. Looking at the front of the board, push the socket in to the holes inside the box with 'MAX7219CNG' printed inside. The plastic body of the socket should be on the top (front) of the board.



Make sure the small notch in the socket is to the left of the board, as this helps remind us which way around to fit the IC later on.

IMPORTANT: The bottom of the socket must be flush with the edge of the PCB to give the displays room to be fitted next.



Solder a corner pin, and then an opposite corner, then check that the bottom of the socket is flat and flush with the PCB. If not, re-heat the corners where appropriate and carefully push into the correct position.

Once happy with the position, solder the remaining pins.

10. Solder the Displays

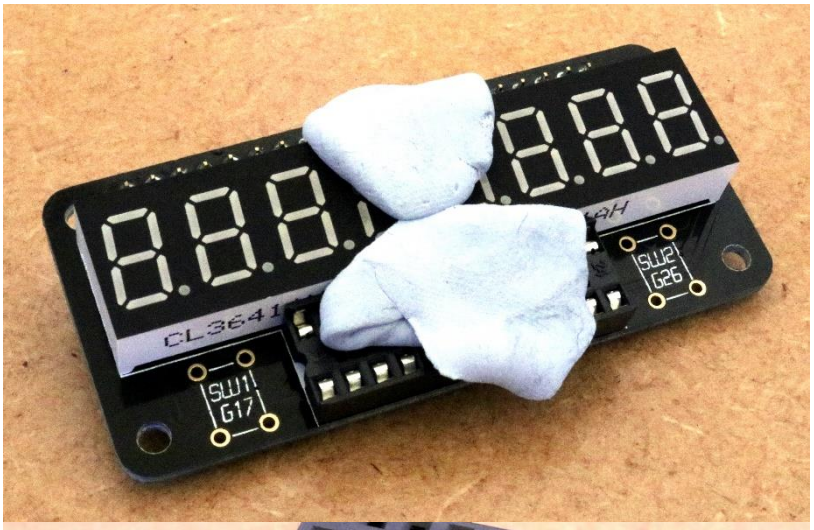
Next fit the segment displays. The 2 boxes on the front of the board with '8.8.8.8' printed inside them are where the displays fit to the board.

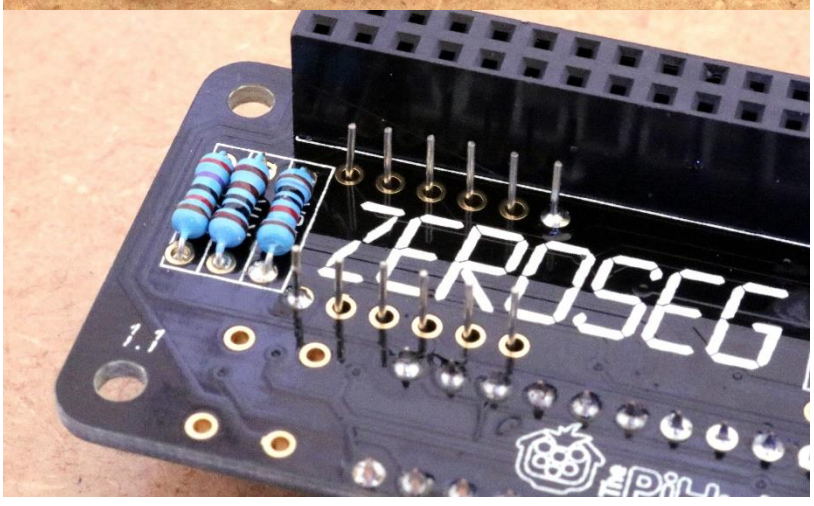
The displays fit the same way as the printed '8.8.8.8' text, with the decimal points next to the IC socket.

IMPORTANT: Fit the displays as close together as possible, with no gap, whilst keeping them flush against the IC socket – again, with no gap.

IMPORTANT: Take time when soldering the display pins, taking care not to touch the rear components with your soldering iron.

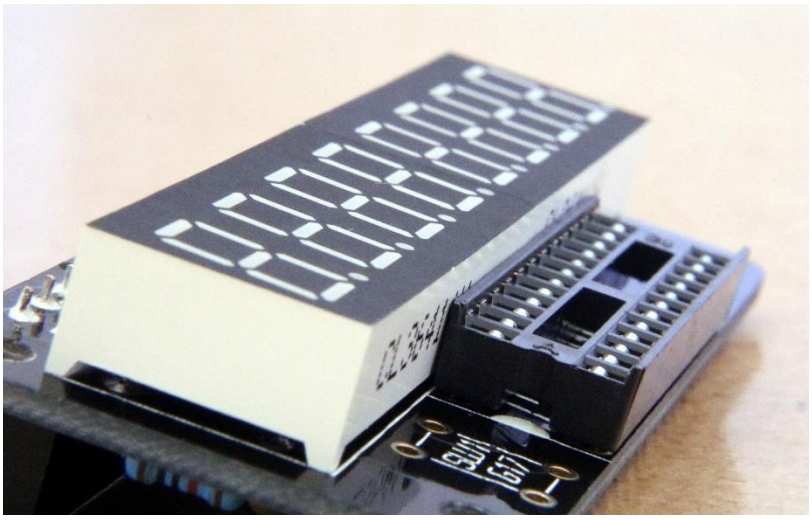
With the displays in place, solder 2 opposite corners of each display. It may be helpful to use Blu-Tack or similar to hold the displays in place whilst you solder the corners.



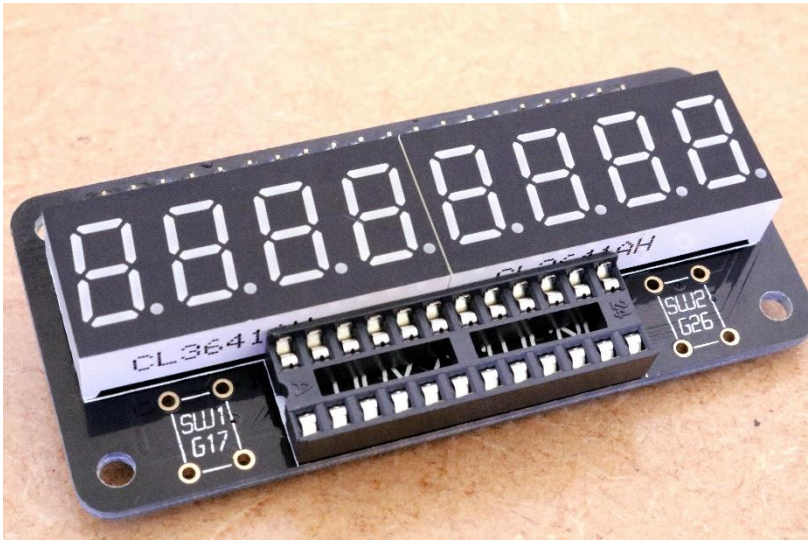


Turn the board over and check the fit.

If the displays aren't straight or there is a gap between them, or they aren't flush with the IC socket, re-heat the necessary corner joints and move into place.



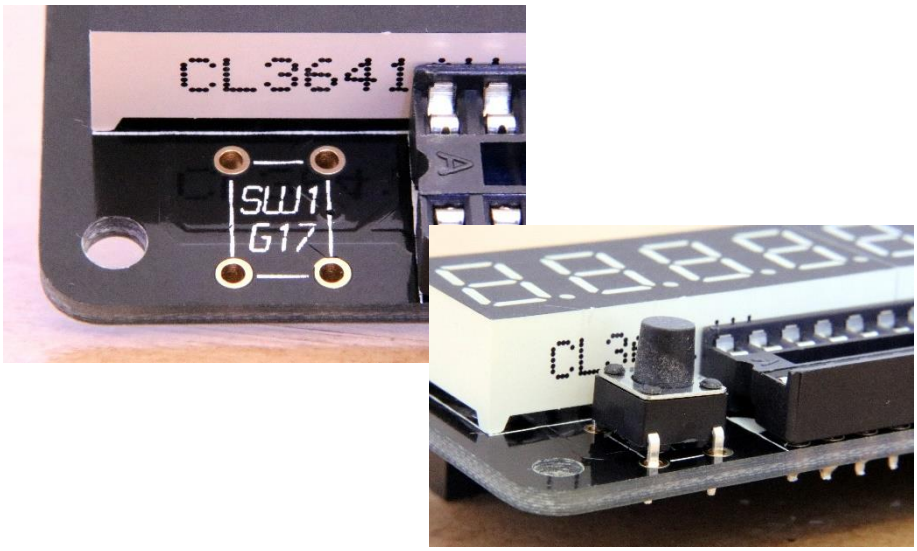
Once you're happy that the displays are straight and close together, solder the remaining legs, and then snip off any excess leg.



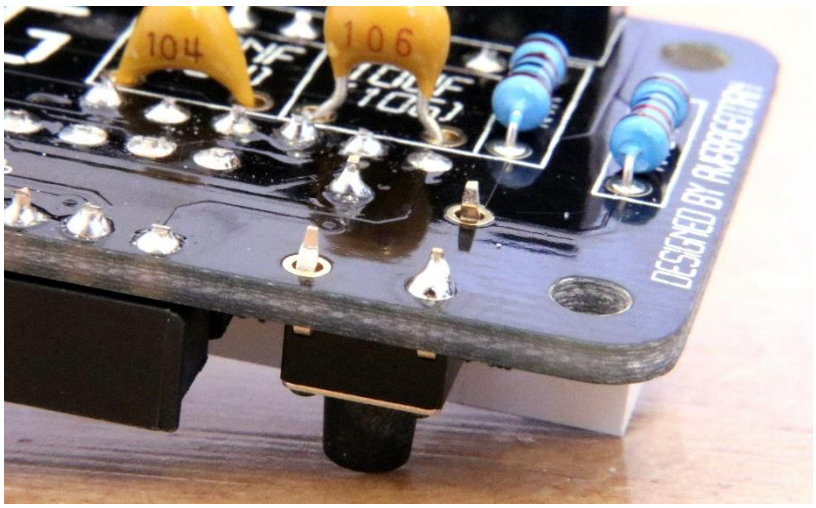
11. Solder the Switches

The next components to fit are the two tactile switches.

The switches of course need to be on the front of the board, and are positioned inside the boxes labelled ‘SW1’ and ‘SW2’.



Push each switch into place, with the 4 legs going through the holes. As always, solder 2 opposing corners first and then check the fit:

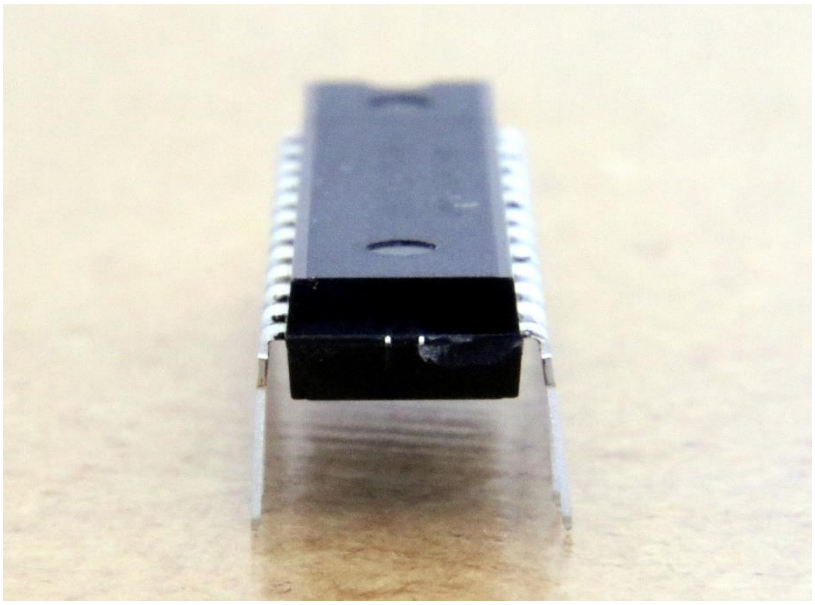


Once you’re happy that each switch is straight and sitting on the PCB correctly, solder the remaining legs.

12. Fit the IC

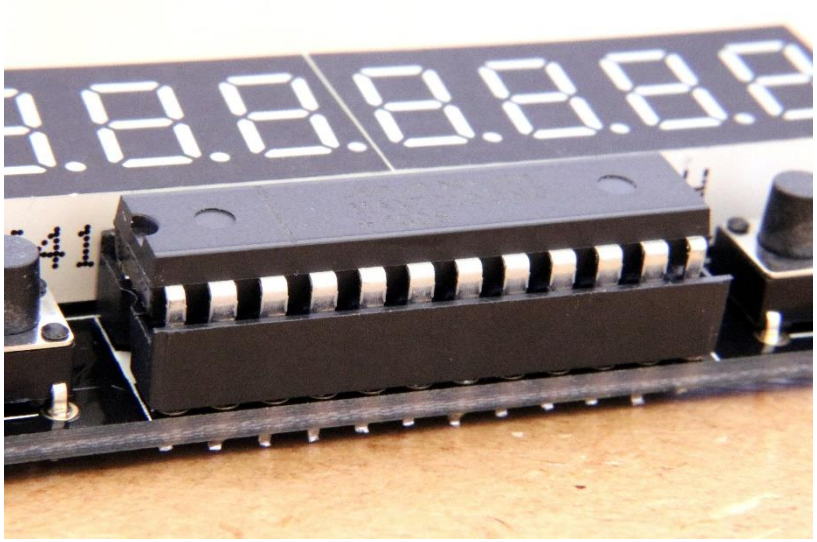
The last step in the assembly is to fit the MAX7219CNG IC to the socket.

You will find that the legs of the chip are sitting too wide to fit into the socket, which is common and how they are delivered from the factory.



Check how far out the legs are, then gently bend them in slightly.

Fit the chip in to the socket, ensuring the 'D' notch on the chip is to the left of the board (also indicated by the socket and the print on the PCB).



Remove the protective film from the displays, wipe off any excess glue, and you're finished!

SETUP AND INSTALL

INTRODUCTION

The ZeroSeg comes with a code library which makes it easy to start programming with the board. This library also includes a number of example scripts to help get you started.

The library is a clone of the excellent work of Richard Hull, who shared his MAX7219 code via GitHub for use with generic display modules.

We've cloned this library in GitHub, made a few changes to make it work with the ZeroSeg, and then added some more specific and simplified example Python scripts to help you get started.

Note: the MAX7219 library has since been replaced with the luma.led_matrix library, which also works with the ZeroSeg.

You will require an internet connection on your Raspberry Pi to install the provided library, as the setup process downloads the files from GitHub. Don't worry – it's a simple step by step process with no GitHub experience/account required.

Alternatively, if you're a *master programmer*, perhaps you'll make your own library and share it with the world?

INSTALL PROCESS

We recommend using a fresh Raspbian image to avoid any conflicts.

1. Download and install the latest copy of Raspbian from here: https://www.raspberrypi.org/downloads/raspbian/

2. Boot (power on) your Raspberry Pi either directly to terminal, or open a terminal window from the desktop. The displays *may* light up, this is perfectly normal.

3. In the terminal window, update Raspbian by typing the following commands and then hitting enter after each one:

   ```
   sudo apt-get update
   ```

   ```
   sudo apt-get upgrade
   ```

4. Next we need to enable SPI. Run the configuration tool by entering the following command and hitting enter:

   ```
   sudo raspi-config
   ```

   When the tool opens, use the arrow keys to go down to option 5 'Interfacing Options' and press enter.

   In the next menu, use the arrow keys to highlight option 4 'SPI' and press enter.

   Select *Yes* to enable the SPI interface and hit enter.

5. Exit the config tool by selecting ***Finish*** and hitting enter again.

6. Reboot your Raspberry Pi by entering `sudo reboot` in the terminal, and hitting enter.

7. Once rebooted, enter the following command and press enter:

   ```
   sudo apt-get install git build-essential python-dev python-pip
   ```

   Enter `Y` when prompted, and hit enter. Let the install run.

8. Enter `cd` in the terminal and hit enter, to ensure you're in the home directory

9. Next enter the following command and press enter:

   ```
   git clone https://github.com/AverageMaker/ZeroSeg.git
   ```

   This downloads the ZeroSeg code library to your Pi.

10. Enter the following command and press enter, to enter your new ZeroSeg directory (case sensitive):

    ```
    cd ZeroSeg
    ```

11. Next run the following command and hit enter:

    ```
    sudo python setup.py install
    ```

12. With the files now downloaded, complete the SPI setup. Whilst still in the ZeroSeg directory, run the following command and hit enter `sudo pip install spidev`

13. **Optional**: To delete the files you probably won't need (and reduce clutter) whilst in the ZeroSeg directory run the following commands, hitting enter after each:

    ```
    rm LICENSE.md
    ```

    ```
    rm README.md
    ```

    ```
    rm setup.py.
    ```

TEST THE DISPLAY

A main example Python script (*zeroseg_example.py*) has been provided, which showcases a number of different display examples. A great way to fire up your ZeroSeg for the first time!

To run this script:

1. Enter `cd` in the terminal and hit enter, to make sure we're starting from the same place (the home directory).
2. Now enter `cd ZeroSeg` and press enter - this will take you to the main ZeroSeg directory.
3. Next enter `cd examples` and hit enter, to go to the example script directory.
4. To run the test script, enter `sudo python zeroseg_example.py` and press enter.
5. The displays should show the date, then the brightness should fade in and out. The date will then scroll left, followed by the time being displayed. Next the display will count up from a negative number, followed by hex numbers and finally random number count.

EXAMPLE SCRIPTS

We have provided a number of additional Python scripts in the 'examples' directory of the ZeroSeg library to help get you started:

button_example.py - Gives basic button control code. Press the buttons to see printed text in your terminal window.

time_example.py - Constantly shows the time on the display until the script is manually closed.

scrolling_example.py - Scrolls the number '1234' on to the display, then scrolls away to the left, and repeats until the script is manually closed.

date_example.py - Shows the date on the display, and updates every 15 minutes until the script is manually closed.

date_time_example - Shows the date for 5 seconds, then the time for 5 seconds, and repeats until the script is manually closed.

text_example.py - Displays a static message up to 8 characters for 3 seconds until the script is manually closed.

rotating_example.py - Scrolls text until the script is manually closed.

CODING BASICS

This section expands on the code found within the example Python scripts.

Extracting code from the main example script (*zeroseg_example.py*) is a great way to learn how to use the library and expand on it further for your own projects:

IMPORTS AND BASIC DATA DISPLAY

Start your file by importing and initialising the *sevensegment* class:

```
1. import ZeroSeg.led as led
2. device = led.sevensegment()
```

You can now start with something simple like displaying the number '1234' for 5 seconds on one of the displays:

```
1. import ZeroSeg.led as led
2. import time
3. device = led.sevensegment()
4.
5. device.write_number(deviceId=0, value=1234)
6. time.sleep(5)
```

SHOW THE DATE

A common project feature for displays like this, is to show the date.

Using code found in the example script, we can extract what we need to create a simple date display that updates every 15 minutes:

```
#!/usr/bin/env python

import ZeroSeg.led as led
import time
from datetime import datetime

def date(device, deviceId):

    now = datetime.now()
    day = now.day
    month = now.month
    year = now.year - 2000

    # Set day
    device.letter(deviceId, 8, int(day / 10))     # Tens
    device.letter(deviceId, 7, day % 10)          # Ones
    device.letter(deviceId, 6, '-')               # dash
    # Set day
    device.letter(deviceId, 5, int(month / 10))   # Tens
    device.letter(deviceId, 4, month % 10)        # Ones
    device.letter(deviceId, 3, '-')               # dash
    # Set day
    device.letter(deviceId, 2, int(year / 10))    # Tens
    device.letter(deviceId, 1, year % 10)         # Ones

device = led.sevensegment(cascaded=2)

while True:
    date(device, 1)
    time.sleep(900)                               # Update every 15 minutes
    device.clear()
```

SHOW THE TIME

Again using parts of the example script, we can simply show the time with a blinking 'dot' LED between the hour and minute digits:

```
#!/usr/bin/env python

import ZeroSeg.led as led
import time
from datetime import datetime

def clock(device, deviceId, seconds):

    for _ in xrange(seconds):
        now = datetime.now()
        hour = now.hour
        minute = now.minute
        second = now.second
        dot = second % 2 == 0                          # calculate blinking dot
        # Set hours
        device.letter(deviceId, 4, int(hour / 10))     # Tens
        device.letter(deviceId, 3, hour % 10, dot)     # Ones
        # Set minutes
        device.letter(deviceId, 2, int(minute / 10))   # Tens
        device.letter(deviceId, 1, minute % 10)        # Ones
        time.sleep(1)

device = led.sevensegment(cascaded=2)

while True:
    clock(device, 1, seconds=10)
```

SCROLL NUMBERS

Scrolling numbers is another fun element to add to your project's display.

Once again, we're using code from the example script to simply scroll some fixed numbers ("1234"). Chop, change and add to this for your own project needs:

```
#!/usr/bin/env python

import ZeroSeg.led as led
import time

device = led.sevensegment(cascaded=2)

while True:
    device.write_number(deviceId=0, value=1234)
    for x in xrange(2):
        for _ in xrange(8):
            device.scroll_right()
            time.sleep(0.1)
        time.sleep(1)
    device.clear()
```

DISPLAY STATIC TEXT

You can display static text on the ZeroSeg, however as with all 7-segment displays, not every character will work (think about 'W' for example).

Where a letter isn't able to be displayed, an underscore will take its place.

Tip: you can mix letters and numbers with this function:

```
1.  import ZeroSeg.led as led
2.  import time
3.
4.  device = led.sevensegment(cascaded=2)
5.
6.  device.write_text(1,"HELLO")
7.
8.  time.sleep(3)
9.  device.clear()
```

SCROLL ANYTHING!

Scrolling a mix of text and numbers is very easy too. With just a single line of code you can scroll a set message across the display:

```
1.  import ZeroSeg.led as led
2.
3.  device = led.sevensegment(cascaded=2)
4.
5.  device.show_message("HELLO EVERYONE")
```

If you want to speed it up or slow it down, just add the optional delay parameter and have a play with the numbers:

```
1.  import ZeroSeg.led as led
2.
3.  device = led.sevensegment(cascaded=2)
4.
5.  device.show_message("HELLO EVERYONE", delay=0.1)
```

BUTTON CONTROL

We've added a button to each side of the board, which will come in handy for most projects using these displays.

The buttons are hard-wired to GPIO 17 (left button) and GPIO 26 (right button). To use them is just a case of using the RPi.GPIO library which is a very common method.

Within the examples directory is a script called *button_example.py*, which shows some basic control by printing on screen when each button is pressed:

```
#!/usr/bin/env python

import time
import RPi.GPIO as GPIO

switch1 = 17
switch2 = 26

GPIO.setmode(GPIO.BCM)              # Use BCM GPIO numbers

GPIO.setup(switch1, GPIO.IN)
GPIO.setup(switch2, GPIO.IN)

print "start"

while True:
    if not GPIO.input(switch1):
        print "Button 1 pressed"
        time.sleep(0.5)

    elif not GPIO.input(switch2):
        print "Button 2 pressed"
        time.sleep(0.5)

    else:
        pass
```

BRIGHTNESS CONTROL

A great feature of the MAX7219CNG is the ability to change the display brightness using software rather than a physical control.

Even better, we've added a couple of buttons to the board so that you can use them for controlling things like display brightness (or anything you want!).

Brightness can be set from 1 to 15 (low to high), and we've incorporated this in the code below to avoid going out of this range and breaking the library script.

We're using generic GPIO code to control the switches, examples of which can be found in the *button_example.py* script in the examples directory (see the button control section above as well):

```
#!/usr/bin/env python

import ZeroSeg.led as led
import RPi.GPIO as GPIO
import time

device = led.sevensegment()

switch1 = 17
switch2 = 26

GPIO.setmode(GPIO.BCM) # Use BCM GPIO numbers

GPIO.setup(switch1, GPIO.IN)
GPIO.setup(switch2, GPIO.IN)

level = 10

while True:
    print "run - level is at ", level
    device.write_number(deviceId=0, value=12345678)
    device.brightness(level)

    if not GPIO.input(switch1):
        if level == 1:
            print "minimum brightness reached"
            time.sleep(0.5)
        if level >= 2:
            level = level -1
            print "Button 1 pressed - brightness down to ", level
            time.sleep(0.5)

    elif not GPIO.input(switch2):
        if level == 15:
            print "Max brightness reached"
            time.sleep(0.5)
        if level <= 14:
            level = level +1
            print "Button 2 pressed - brightness up to ", level
            time.sleep(0.5)

    time.sleep(0.2)
```

GPIO PINS USED

The following table shows the GPIO pins which have been used in the ZeroSeg board. Power and GND lines have not been documented, however it's worth mentioning that the MAX7219CNG chip uses the 5V line, whilst the switches use 3.3V.

All other remaining GPIO pins are available for use (if breaking out separately).

Name	Description	Physical Pin	RPi Function
DIN	Data In	19	GPIO 10 (MOSI)
CS	Chip Select	24	GPIO 8 (SPI CE0)
CLK	Clock	23	GPIO 11 (SPI CLK)
SW1	Left Switch/Button	11	GPIO 17
SW2	Right Switch/Button	37	GPIO 26

FAQS

Q. Can I use the ZeroSeg with any Raspberry Pi?

A. The ZeroSeg can only be used with 40 GPIO pin Raspberry Pis (B+, A+, Pi2, Pi Zero, Pi Zero W, Pi3 and the Pi3B+).

Q. Is this a HAT?

A. No. HATs have a set specification including EEPROM chips and other standards. We didn't feel this product required HAT features, so we prefer to call it an "add-on board".

Q. Can I stack this with other boards?

A. Yes! This board can be added on top of a stack, using the mounting holes to secure it to other add-on boards or HATs.

Q. How do I programme this board for my project?

A. We have provided example scripts to help get you started. This includes displaying static data, scrolling data, time, date, brightness and button control code. Use these scripts to help you create your specific project.

Q. Do the displays require an additional power supply?

A. No. The standard official Raspberry Pi Power Supply provides enough power.

Q. I made a really cool project with the ZeroSeg and want to show you!

A. Excellent! Tweet us a picture over on Twitter (@ThePiHut)