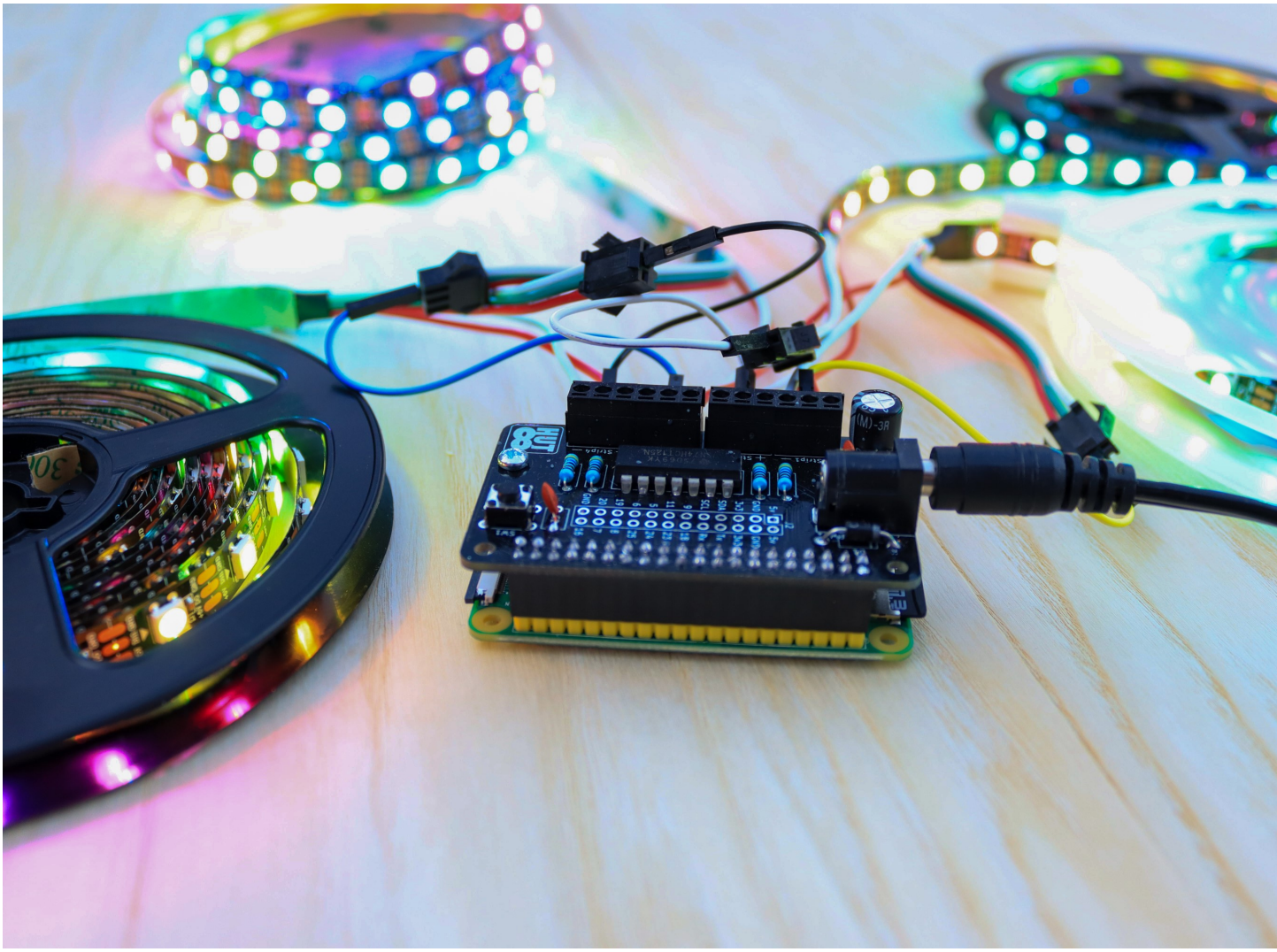
PixelPi Library

Sep 10, 2020

Contents

The PixelPi Library is intended to be used with the PixelPi PCB from Hut 8 Designs.

The PixelPi is a HAT-like add on board from Hut 8 Designs for the Raspberry Pi computer (40-pin versions) and enables you to control up to four (4) WS281x type RGB LED strips or matrices independently with one Raspberry Pi.



The PixelPi in action!

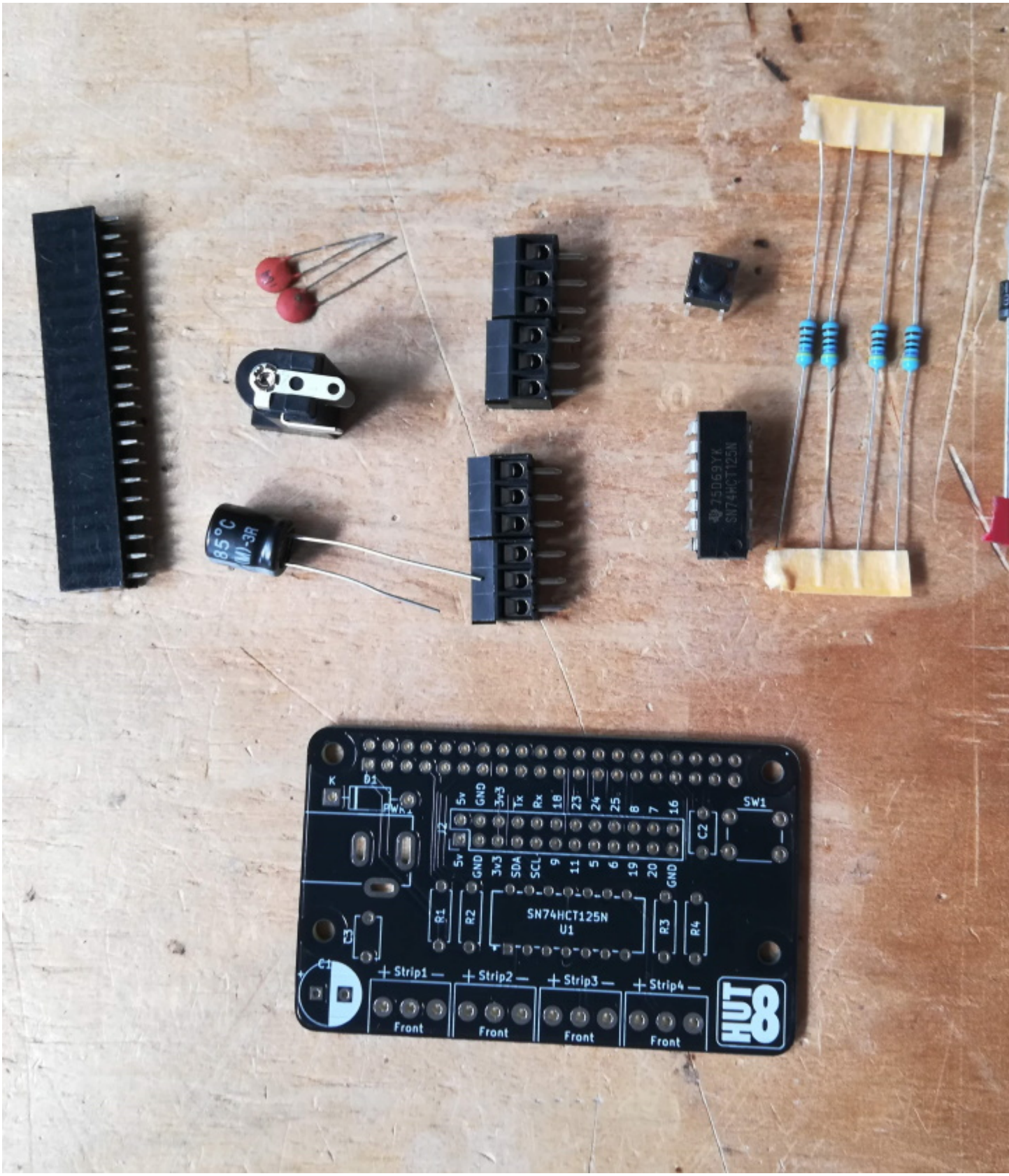
CHAPTER 1

Building the PixelPi Board

1.1 Components

The PixelPi consists of the following components:

- Four 4.7k Resistors (R1. R2. R3, R4)
- Four screw terminal blocks (Strip1, Strip2, Strip3, Strip4)
- One 1N5817 Schottky Rectifier (D1)
- One 2.1mm barrel jack (PWR1)
- Two 1pF capacitors (C2, C3)
- One 470uF capacitor (C1)
- One button (SW1)
- One 2x40 pin header (J1)
- Optional GPIO Pin breakout pins (J2)
- One PixelPi PCB
- One SN74HTC125N Level Shifter

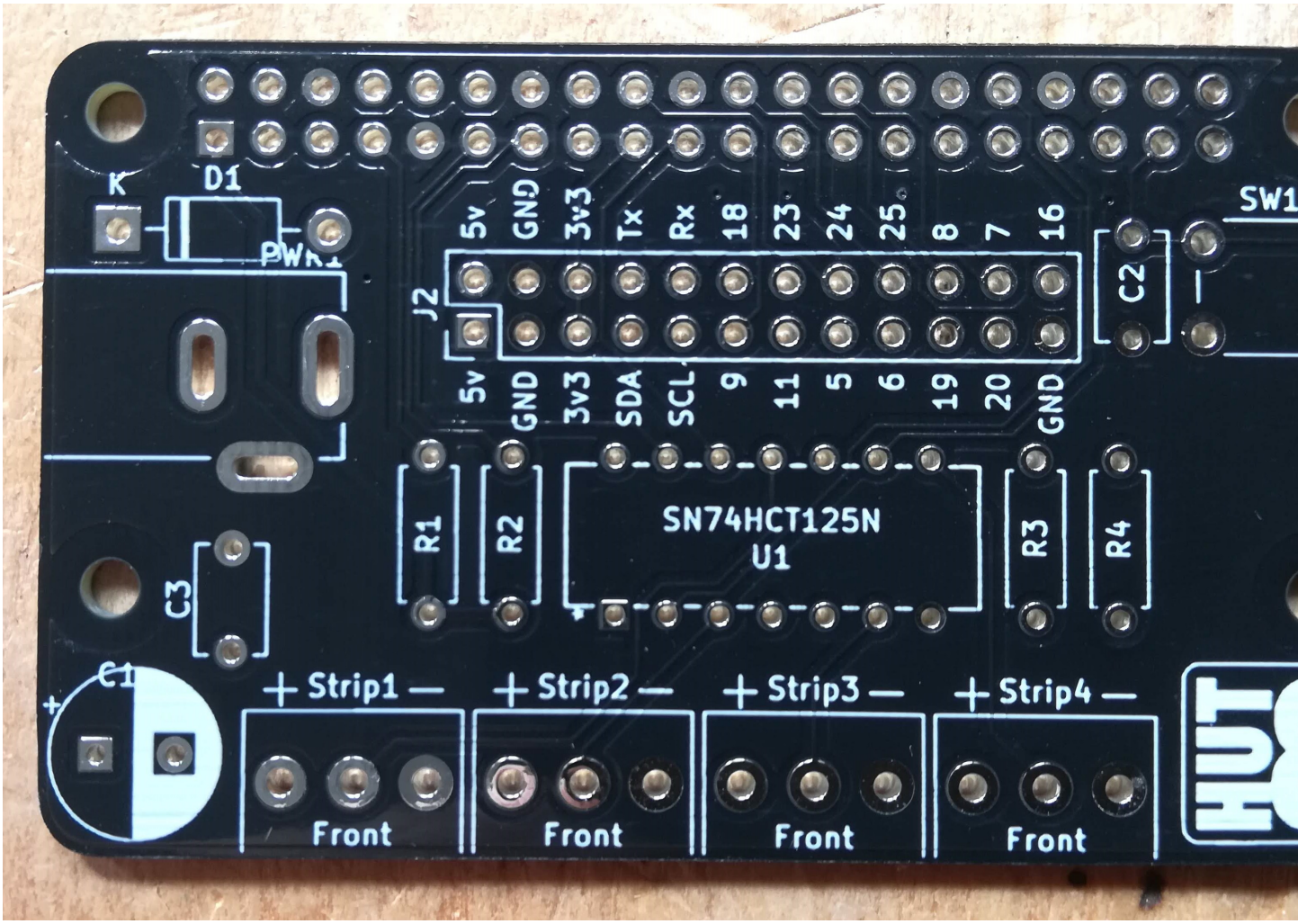

SN74HCT125N
U1
Strip1
Strip2
Strip3
Strip4
Front
SW1
HUT 8

1.2 Soldering Order

The order you solder the board together does not matter, but doing it in the following order will make it easier.

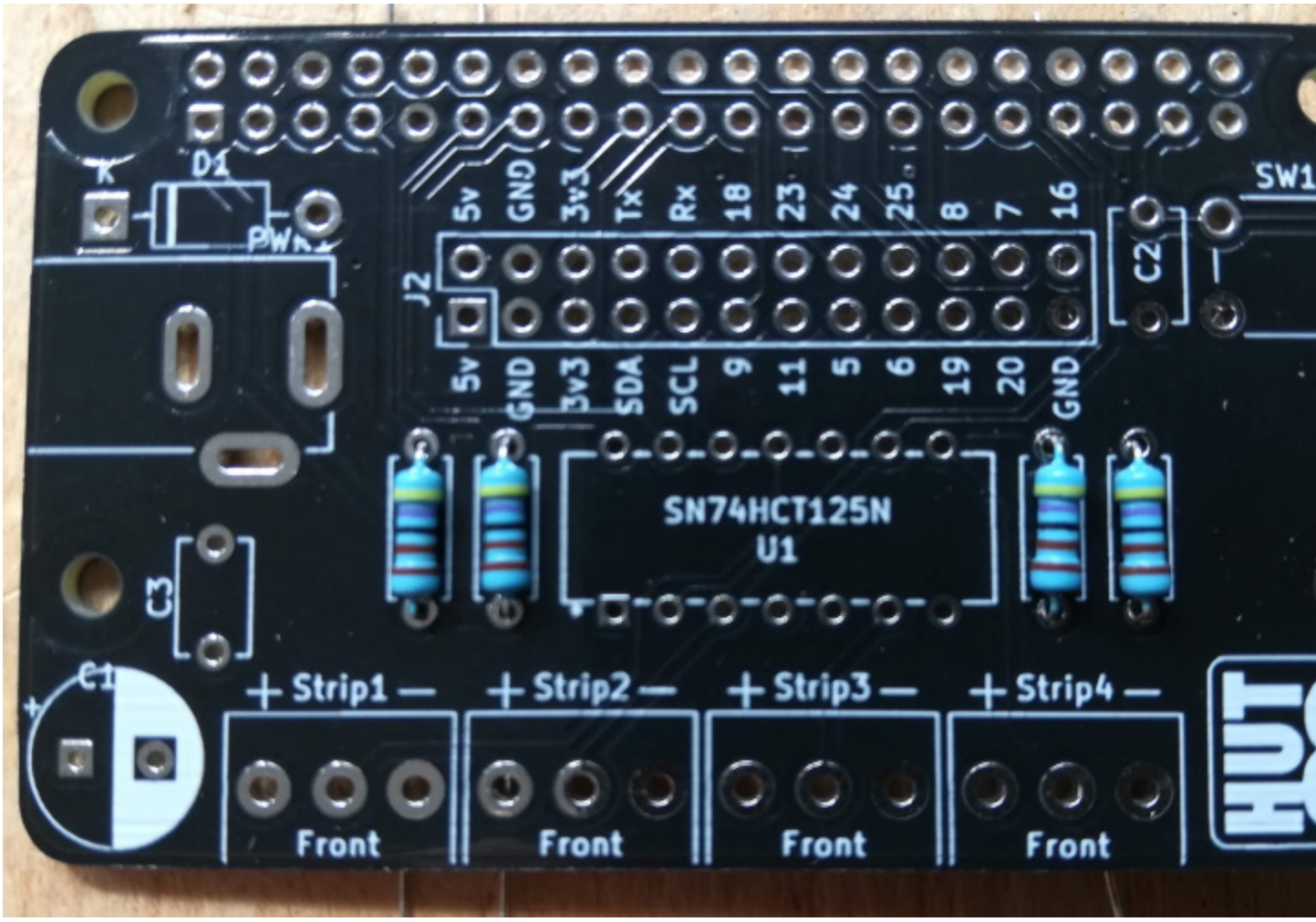
1.2.1 The PCB

View the PCB from the top and locate the positions of each component.



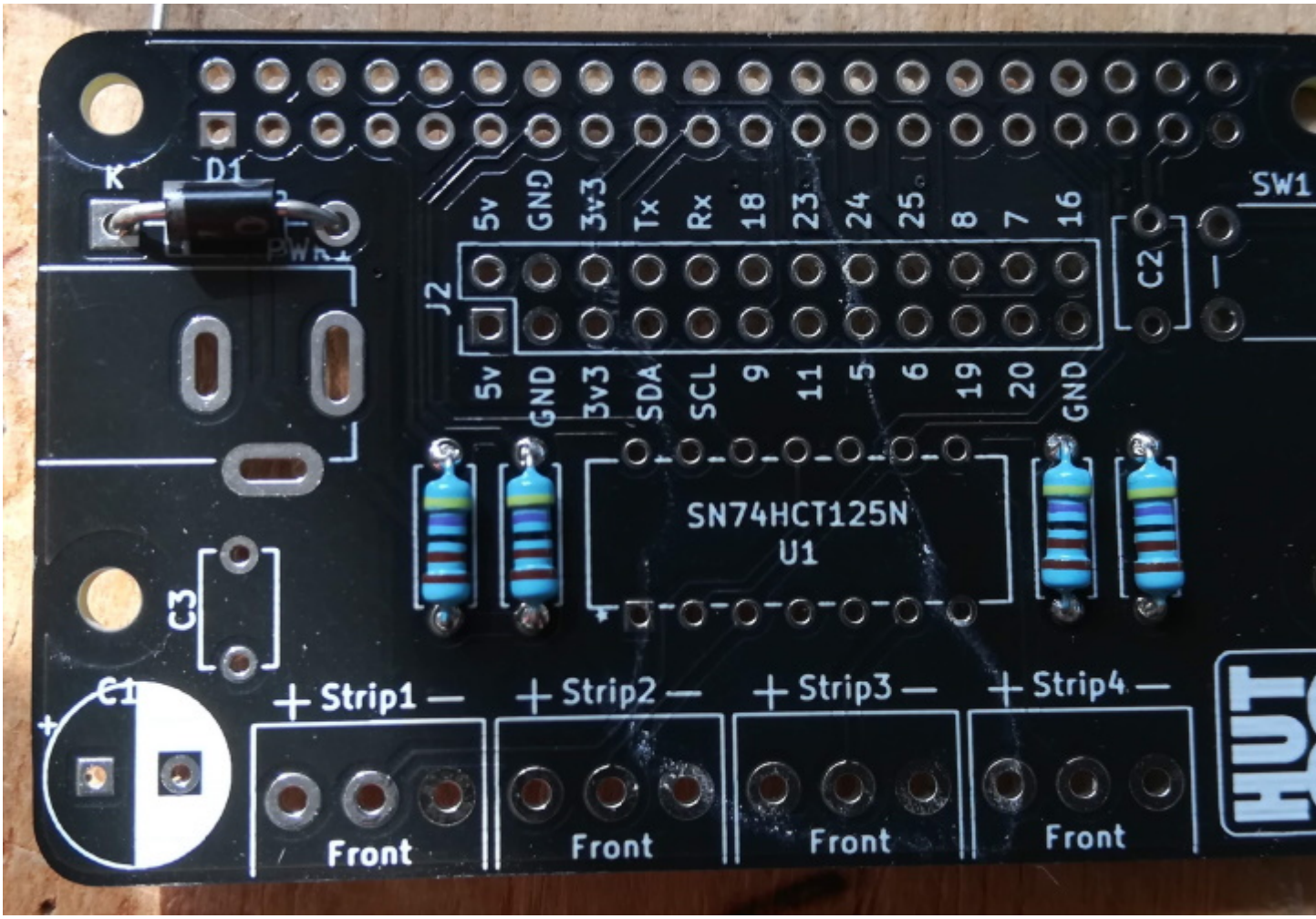
1.2.2 Resistors (R1. R2. R3, R4)

The resistors can be soldered either way round, but they look better if they are all the same way round.

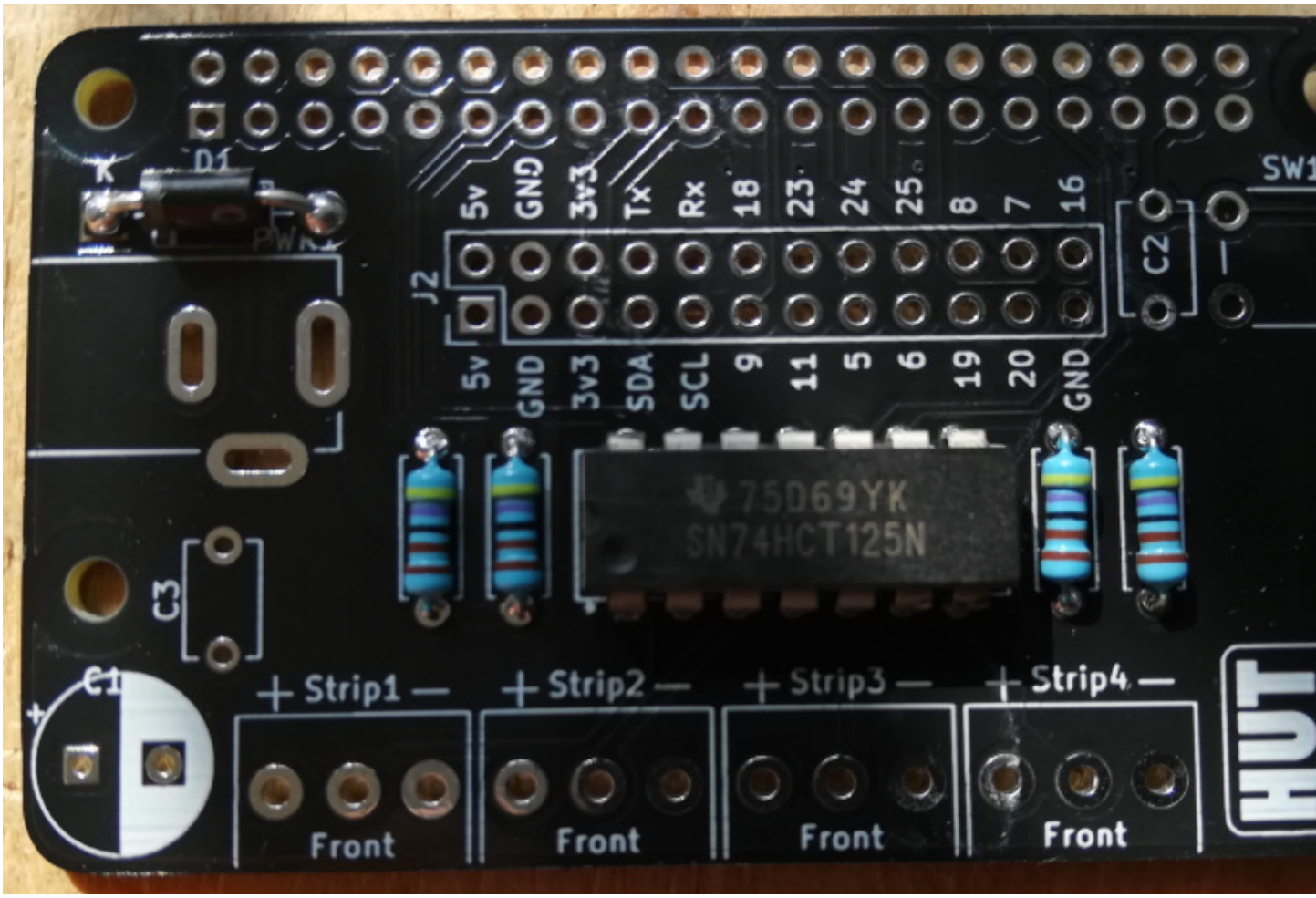


1.2.3 Schottky Rectifier

The rectifier (diode) has to be soldered the correct way around. Locate the white band on one end of the diode and line it up with the white line on the location D1.

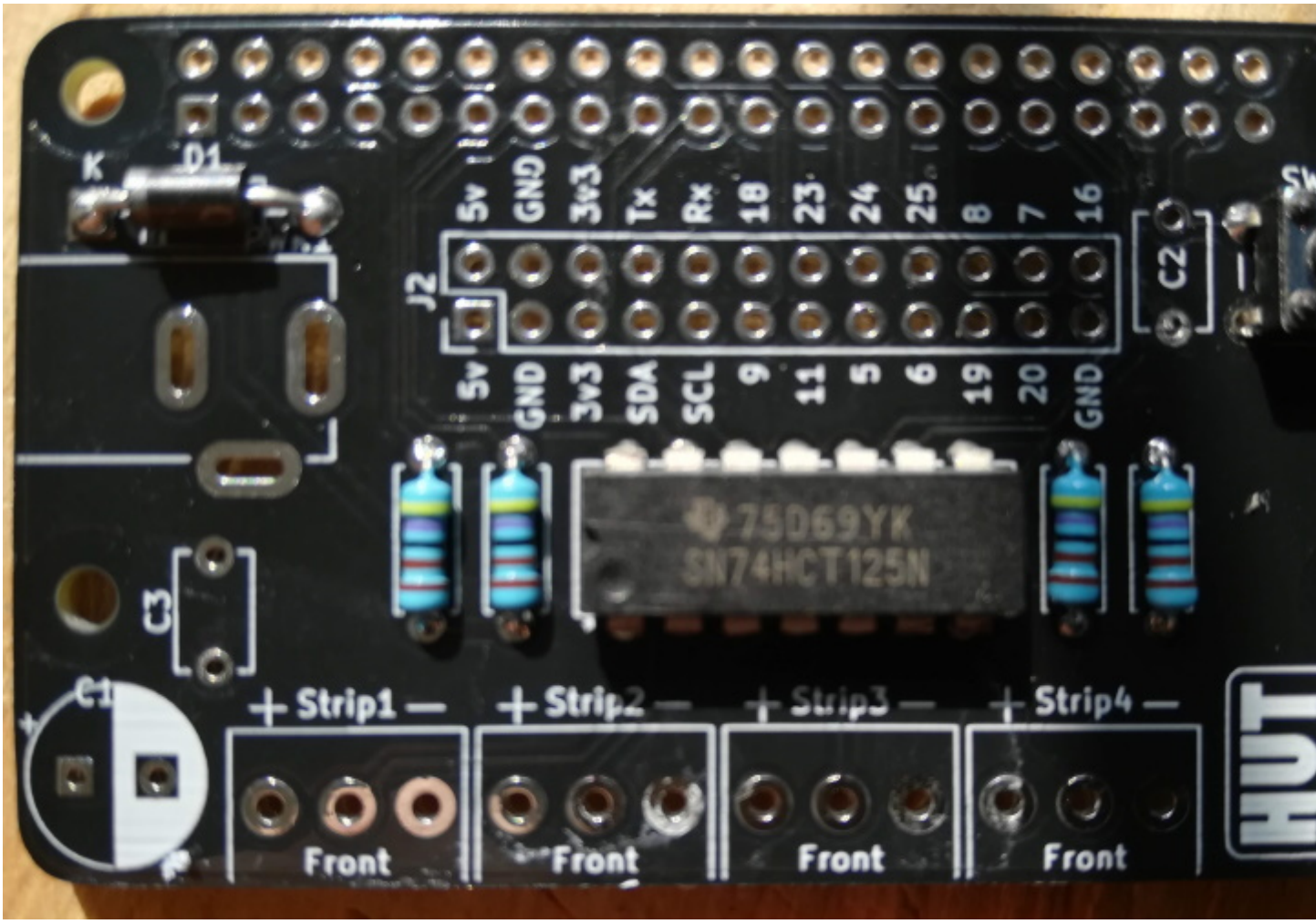

K
D1
SW1
C2
J2
5v
GND
3v3
Tx
Rx
18
23
24
25
8
7
16
5v
GND
3v3
SDA
SCL
9
11
5
6
19
20
GND
SN74HCT125N
U1
C3
C1
Strip1
Strip2
Strip3
Strip4
Front
Front
Front
Front

1.2.4 The IC Chip



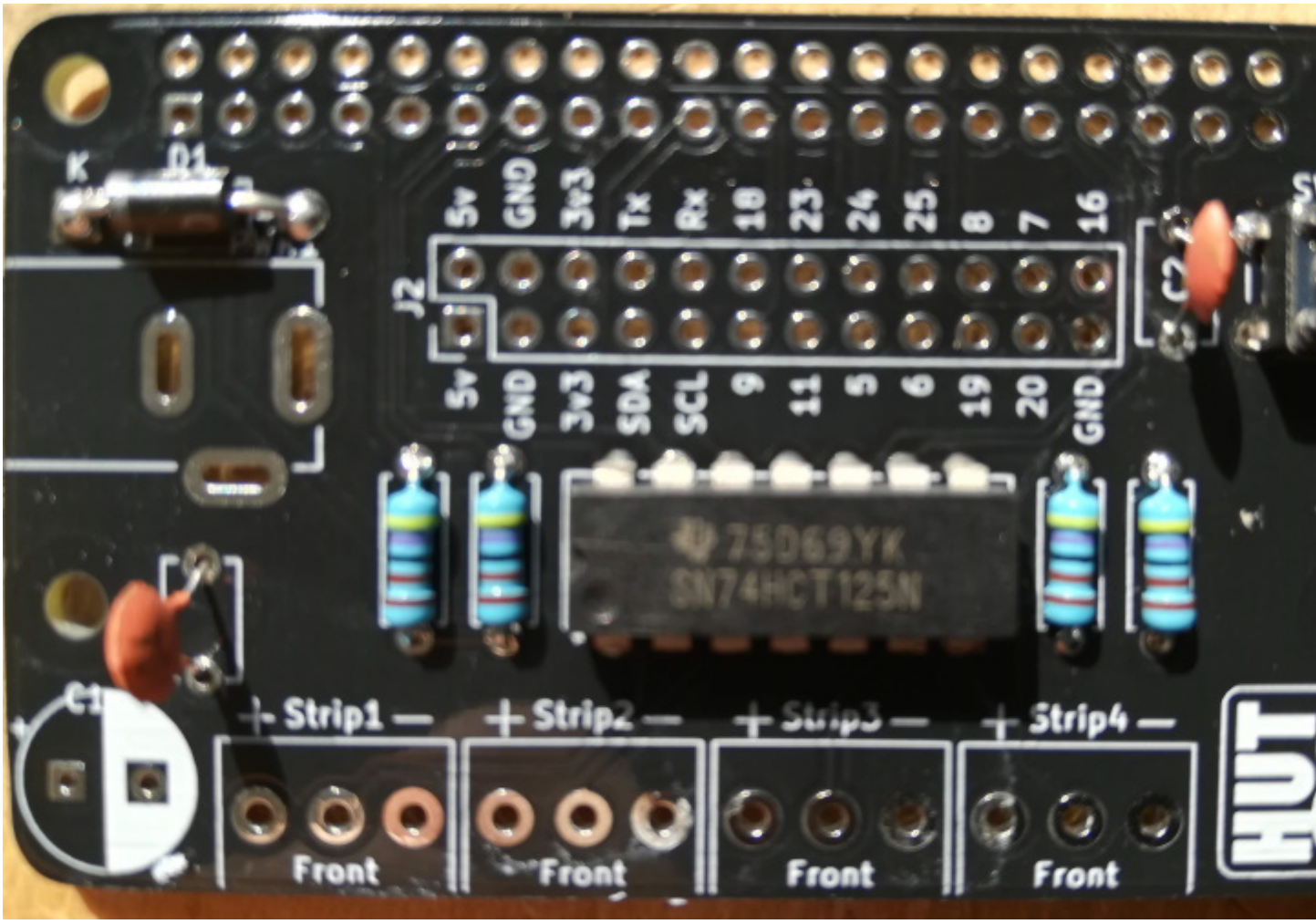
1.2.5 Button

The button will only fit one way around at location SW1.



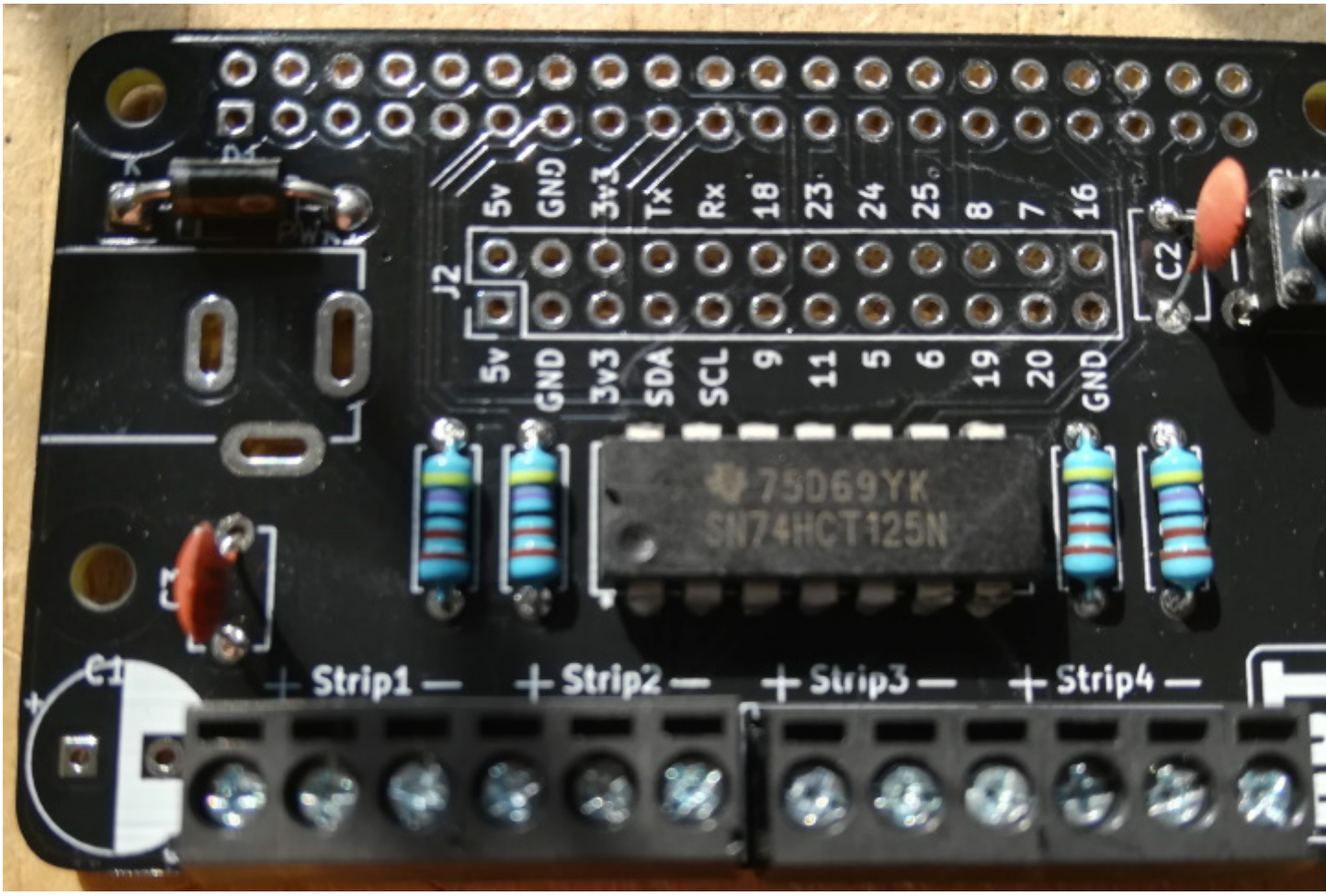
1.2.6 1pF Capacitors (optional, but recommended)

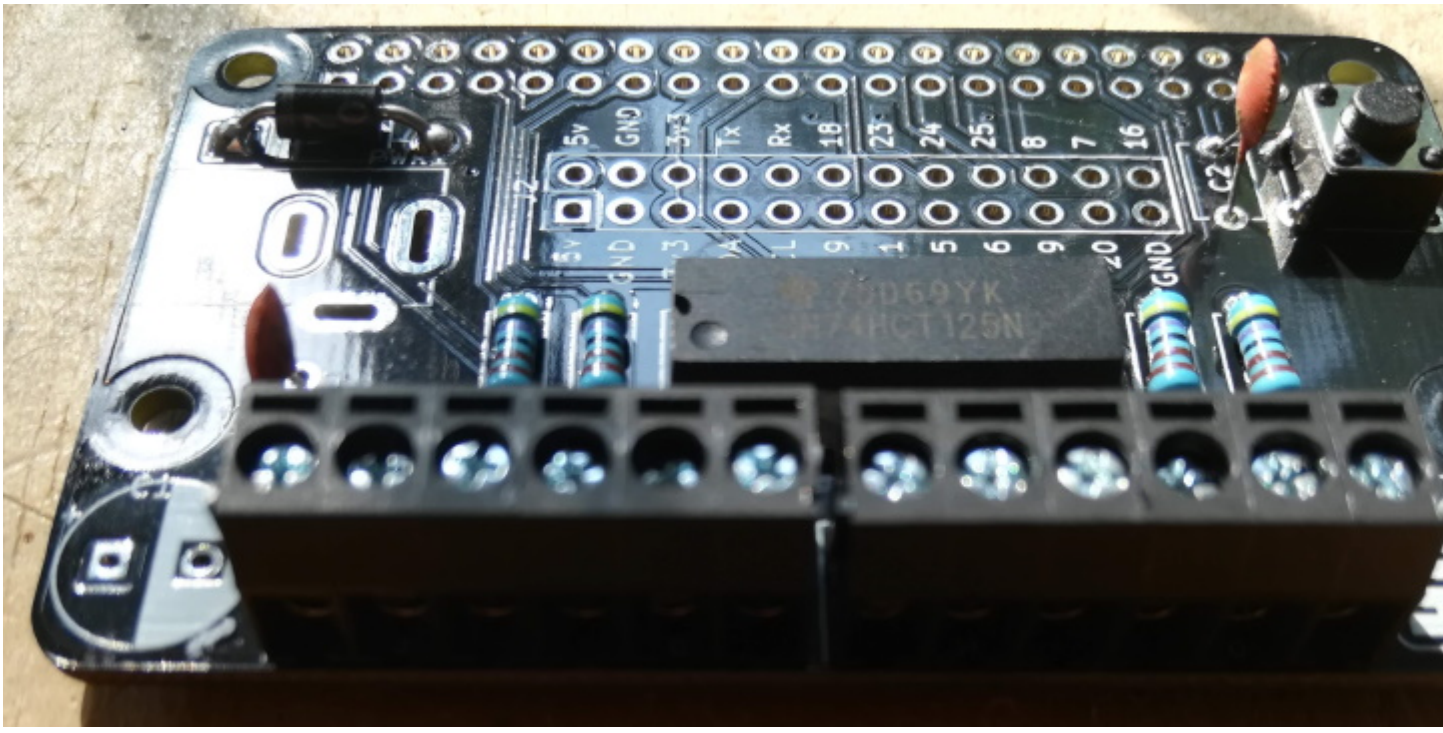
The two 1pF capacitors, locations C2 and C3, can be soldered either way around.



1.2.7 Screw Terminals

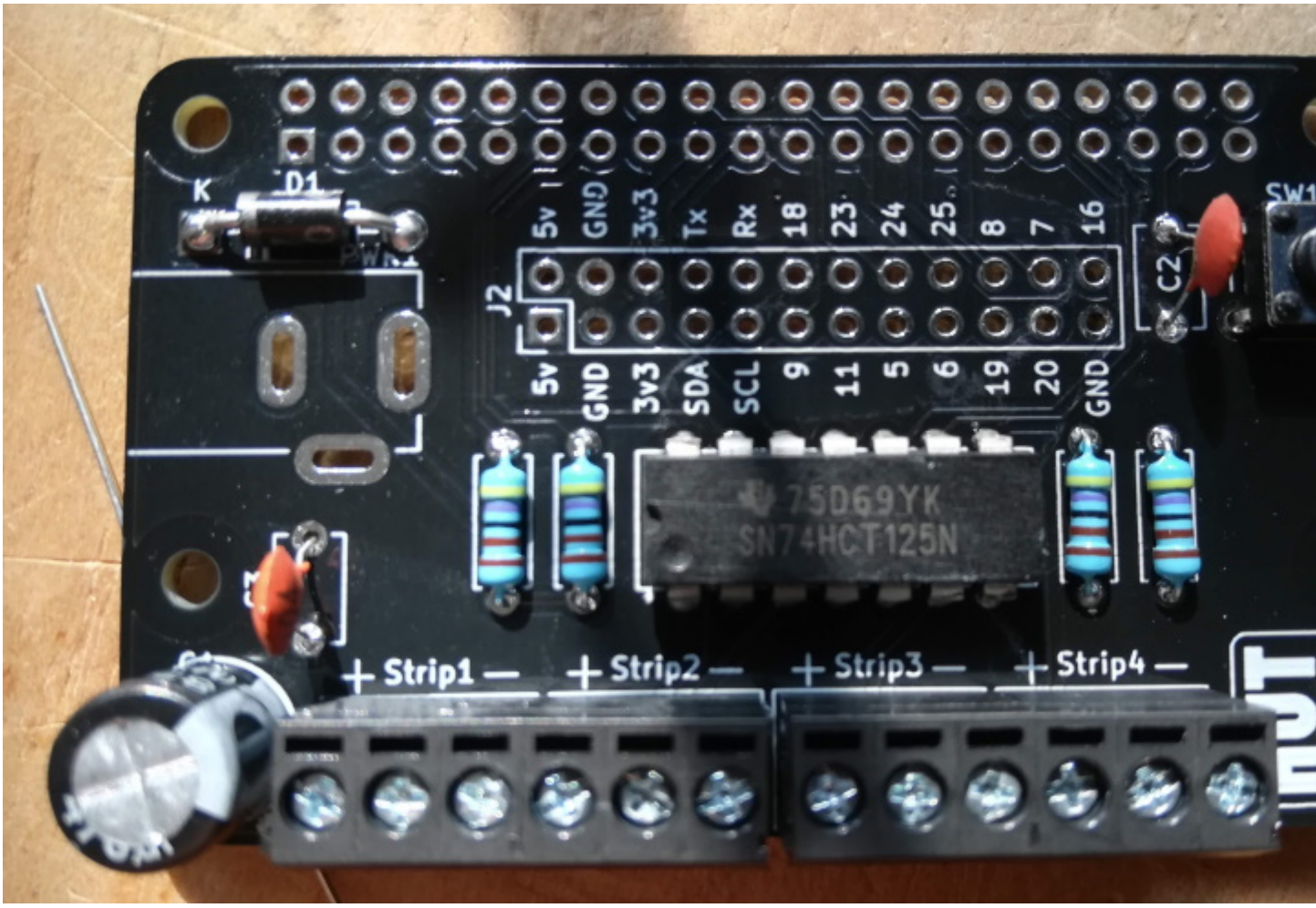
Connect two pairs of terminals together by sliding the edges together, then solder them onto the board with screw terminal access holes towards the front of the board.


5v
GND
3v3
Tx
Rx
18
23
24
25
8
7
16
J2
5v
GND
3v3
SDA
SCL
9
11
5
6
19
20
GND
C2
C1
Strip1
Strip2
Strip3
Strip4



1.2.8 470uF Capacitor (optional, but recommended)

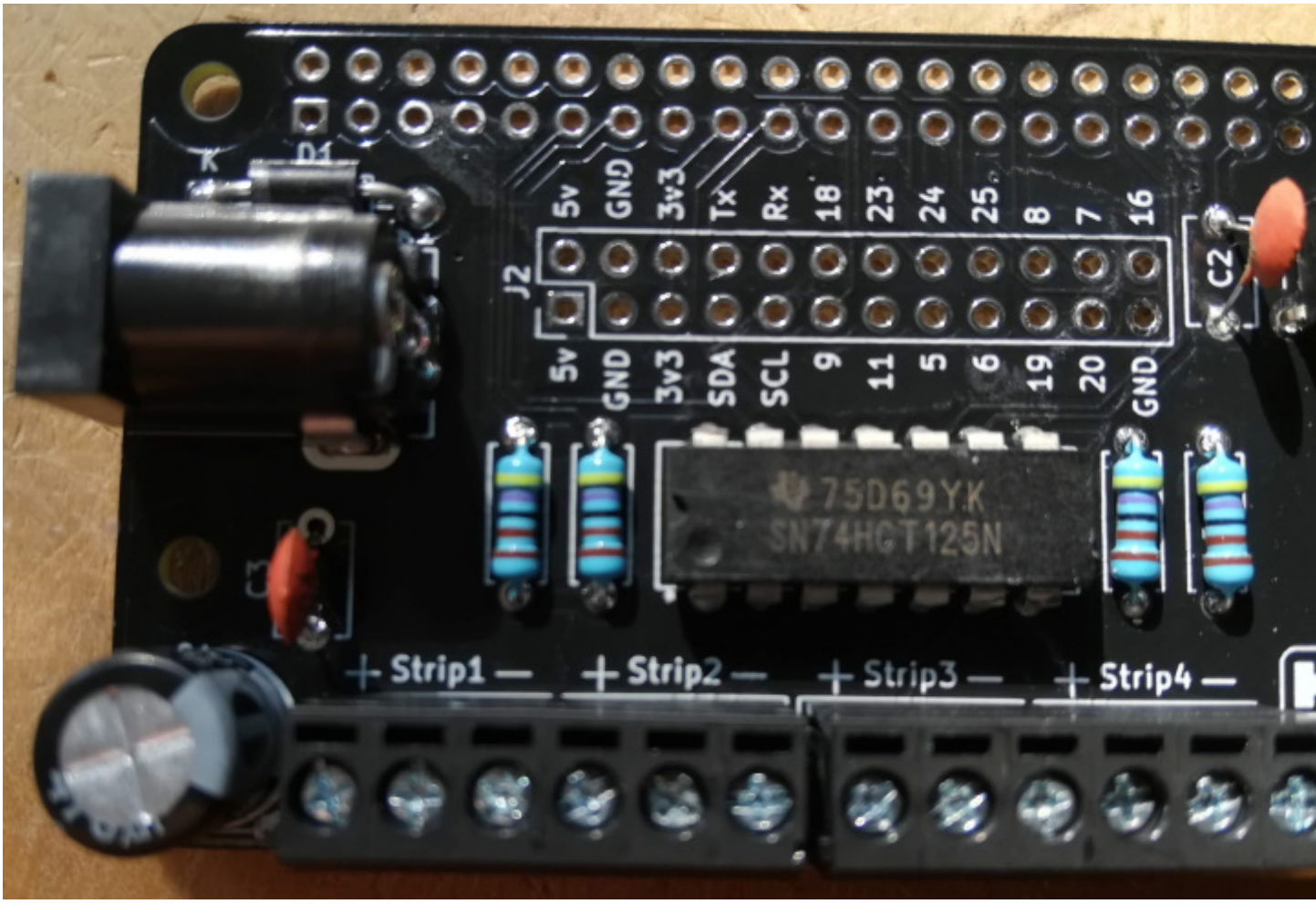
This capacitor has polarity, and therefore has to be soldered the correct way around. The white area of C1 indicates the -ve leg. The -ve leg is marked with a white line and - symbol.



1.2.9 The Barrel Jack

The last component on the top of the PCB is the barrel jack. This can only fit in one way, with the hole of the barrel pointing away from the board.

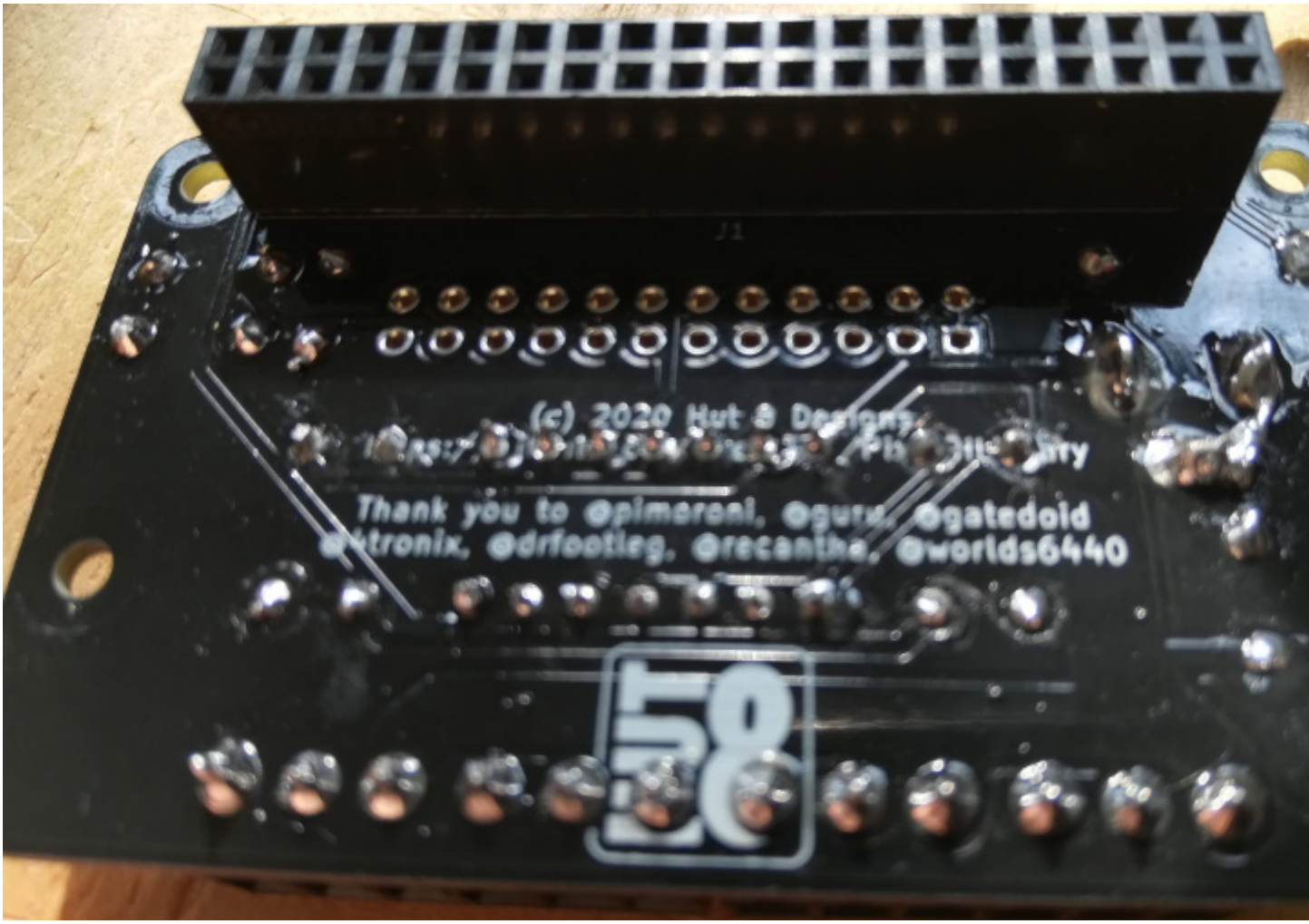
note Only use a 5v power supply with the PixelPi board. Anything higher will kill your Raspberry Pi as well as your LEDs.



Note If you have one of the alpha boards, you will need to solder two pins together due to a small design issue.

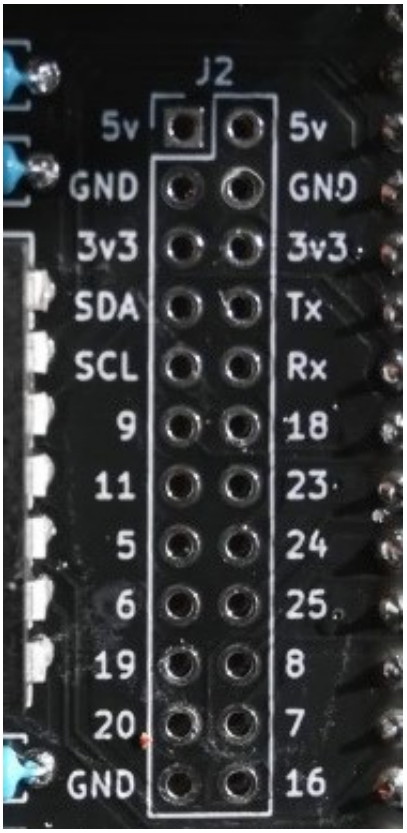
1.2.10 Raspberry Pi Header

The Raspberry Pi header should be mounted on the underside of the PCB, soldered from the top.



1.2.11 (Optional) Breakout Headers

If you plan to use any of the breakout pins, you may solder header pins in place.


J2
5v
GND
3v3
SDA
SCL
9
11
5
6
19
20
GND
5v
GND
3v3
Tx
Rx
18
23
24
25
8
7
16

CHAPTER 2

Installing the Python Library

This PixelPi Library is a fork of the Pimoroni (http://pimoroni.com) Plasma code (https://github.com/pimoroni/plasma), massively simplified and extensively modified to be used with the PixelPi Raspberry Pi 'small' HAT from Hut 8 Designs.

Note The library has been designed to only work with Python 3.

Before installing the library, you need to prepare you Raspberry Pi OS. These instructions assume you are starting with the latest version of the Raspberry Pi OS Lite.

It is always good to start from an updated OS, so log in and run the following commands:

```
sudo apt update && sudo apt upgrade
```

2.1 config.txt

The PixelPi board needs some of the Raspberry Pi interfaces turned on, and one turned off. The Raspberry Pi audio and the PWM channels used by the PixelPi cannot be used at the same time, so if you are using strip outputs 2 and 4 you need to disable audio on the Pi. Do the following to ensure the audio is completely turned off by editing the boot config file with:

```
sudo nano /boot/config.txt
```

The uncomment this lines, and add the second:

```
hdmi_force_hotplug=1
hdmi_force_edid_audio=1
```

Ensure the following line is commented by placing a # at the start:

```
#dtparam=audio=on
```

Uncomment or add the following to in the optional hardware interfaces:

```
dtparam=spi=on
```

Add the following line to the end of the file to set memory frequency:

```
core_freq=250
```

2.2 Disable Sound in the Kernel

You will also need to edit the kernel module blacklist:

```
sudo nano /etc/modprobe.d/snd-blacklist.conf
```

Add the following line:

```
blacklist snd_bcm2835
```

2.3 Using strip4

Strip 4 uses SPI. You should increase the buffer size to handle most eventuallities of LED strip length. Edit the boot commandline file:

```
sudo nano /boot/cmdline.txt
```

Add the following to the end of the only line to increase the SPI buffer size:

```
spidev.bufsiz=32768
```

2.4 Reboot

You need to reboot your Raspberry Pi for the above changes to take effect:

```
sudo reboot
```

2.5 Installing the PixelPi Library

The PixelPi library is available on GitHub. to retrieve it you need to install git:

```
sudo apt install git -y
```

The library can be downloaded then installed (with all prerequisites) with the following commands:

```
cd ~
git clone http://github.com/geekytim/PixelPiLibrary
cd ~/PixelPiLibrary
sudo bash install.sh
```

CHAPTER 3

Identifying LED Types

When you attach your LEDs you may not know the exact LED type they, and if it is a matrix, you may not know the order of the LEDs. Sample code has been given to help you identify both LED type and matrix shape.

3.1 Connecting the LEDs to the PixelPi board

WS281x LEDs have three wires which are usually red, green and white. The red wire should be connected to + on a terminal, the white to - and the green to the centre.

Strings and matrices that use the same LED type may be connected together to form longer strings or larger matrices but you should note that LED brightness will diminish the further away from the start they are.

Note Only ever connect a 5v power supply to the PixelPi board. Higher voltage will destroy the board, the Raspberry Pi, the LEDs or all three.

3.2 Strings and Matrices

An LED strip, or string, is a set of LEDs on a long, flexible PCB. They come in many different lengths and LED spacing.

LED matrices are a 2D array of LEDs:

3.3 Finding LED Types

Once connected to the board, use the following code (in /PixelPiLibrary/examples/string_test.py or /PixelPiLibrary/examples/matrix_test.py if you are using a matrix):

```
#!/usr/bin/env python3

import time

from pixelpi import Strip

"""
Change the parameters below until you see the colours indicated on the screen:

    terminal = The screw terminal your LEDs are connected to

    size = The number of LEDs in your terminal (can be (x, y) for a matrix)

    shape = The 'shape' of the terminal.
            * `straight` - A led string (default)
            * `reverse` - A led string which starts at the opposite end
            * `matrix` - a normal matrix where the led order goes left to right. i.e:
              1 2 3 4
              5 6 7 8
              9 . . .
            * `zmatrix` - A matrix where the shift in the first row go left to right, the next one
```

(continues on next page)

(continued from previous page)

```
22              right to left. i.e:
23              1 2 3 4
24              8 7 6 5
25              9 . . .
26
27      ledtype = One of the supported terminal types:
28              WS2812, SK6812, SK6812W, SK6812_RGBW, SK6812_RBGW, SK6812_GRBW, SK6812_GBRW,
    ↪ SK6812_BRGW,
29              SK6812_BGRW, WS2811_RGB, WS2811_RBG, WS2811_GRB, WS2811_GBR, WS2811_BRG,␣
    ↪WS2811_BGR
30
31      brightness = The default brightness for all LEDs (0-255).
32  """
33
34  strip = Strip(terminal=1, size=256, shape='straight', ledtype='WS2812', brightness=40)
35
36  try:
37      while True:
38          print("Red")
39          strip.setLEDs(rgb=(255, 0, 0))
40          strip.showLEDs()
41          time.sleep(1)
42
43          print("Green")
44          strip.setLEDs(rgb=(0, 255, 0))
45          strip.showLEDs()
46          time.sleep(1)
47
48          print("Blue")
49          strip.setLEDs(rgb=(0, 0, 255))
50          strip.showLEDs()
51          time.sleep(1)
52
53          strip.clearLEDs()
54          strip.showLEDs()
55
56          for led in range(strip.getLength):
57              print("White: ", led)
58              strip.setLEDs(rgb=(255, 255, 255), led=led)
59              strip.showLEDs()
60              time.sleep(0.25)
61              strip.setLEDs(rgb=(0, 0, 0), led=led)
62
63  except KeyboardInterrupt:
64      strip.clearLEDs()
65      strip.showLEDs()
66      del strip
```

Edit line 34 to set:

- Which terminal your LEDs are connected to.
- The number of LEDs in the string (or matrix size, in (width, height) format))
- The LED type.
- If you are using a matrix, the `shape` must also be set to either `matrix` or `zmatrix`.

To run the code, use the following:

```
sudo python3 string_test.py
```

or:

```
sudo python3 matrix_test.py
```

Note `sudo` is currently required.

If the LEDs do not display colours in the order RED, GREEN, BLUE followed by a white moving LED you should change the LED type to one of the following:

WS2812, SK6812, SK6812W, SK6812_RGBW, SK6812_RBGW, SK6812_GRBW, SK6812_GBRW, SK6812_BRGW, SK6812_BGRW, WS2811_RGB, WS2811_RBG, WS2811_GRB, WS2811_GBR, WS2811_BRG, WS2811_BGR

For example, your LEDs may display BLUE, GREEN then all go white. This will not damage them, but indicates you are using the incorrect type. Change the type until the colors are displayed in the right order.

For a matrix, the white LED should move in the same direction for each row. If it does not, swap the `shape` between `zmatrix` and `matrix`.

CHAPTER 4

Example Code

Example code has been supplied to illustrate the methods available.

Note: Remember to change the Strip definition to suit the LEDs connected. Each of the four terminals may use a different LED type and shape.

Run the code using:

```
sudo python3 example.py
```

4.1 Rainbow

Displays a moving rainbow on your LEDs.

```
#!/usr/bin/env python3

import colorsys
import time

from pixelpi import Strip

# Change the terminal type to the type you have
strip1 = Strip(1, 180, ledtype='WS2811_GRB', brightness=50)
strip2 = Strip(2, 180, ledtype='WS2811_GRB', brightness=50)
strip3 = Strip(3, 180, ledtype='WS2811_GRB', brightness=50)
strip4 = Strip(4, 180, ledtype='WS2811_GRB', brightness=50)

spacing = 360.0 / 16.0
hue = 0

try:
```

(continues on next page)

(continued from previous page)

```
    while True:
        hue = int(time.time() * 100) % 360
        for x in range(256):
            offset = x * spacing
            h = ((hue + offset) % 360) / 360.0
            r, g, b = [int(c * 255) for c in colorsys.hsv_to_rgb(h, 1.0, 1.0)]

            for strip in [strip1, strip2, strip3, strip4]:
                strip.setLEDs(rgb=(r, g, b), led=x)

        for strip in [strip1, strip2, strip3, strip4]:
            strip.showLEDs()

        time.sleep(0.001)

except KeyboardInterrupt:
    for strip in [strip1, strip2, strip3, strip4]:
        strip.clearLEDs()
        strip.showLEDs()
        del strip
```

4.2 Shifting LEDs

The LED colours can be shifted by a number of LEDs and direction:

4.2.1 Shifting LED Strings

```
#!/usr/bin/env python3

from pixelpi import Strip

strip1 = Strip(1, 180, ledtype='WS2811_GRB', brightness=50)
strip2 = Strip(2, 180, ledtype='WS2811_GRB', brightness=50)
strip3 = Strip(3, 180, ledtype='WS2811_GRB', brightness=50)
strip4 = Strip(4, 180, ledtype='WS2811_GRB', brightness=50)

strip1.clearLEDs()
pattern = strip1.getLEDs()

for pixel in range(len(pattern)):
    pattern[pixel] = [0, pixel, 0, pattern[pixel][3]]

for strip in [strip1, strip2, strip3, strip4]:
    strip.setLEDs(pattern=pattern)

try:
    while True:
        for strip in [strip1, strip2, strip3, strip4]:
            strip.showLEDs()
            strip.shift("down", 1)

except KeyboardInterrupt:
```

(continues on next page)

(continued from previous page)

```
    for strip in [strip1, strip2, strip3, strip4]:
        strip.clearLEDs()
        strip.showLEDs()
        del strip
```

4.2.2 Shifting LED Matrices

```
#!/usr/bin/env python3

from pixelpi import Strip
import time

strip = Strip(4, (8, 32), shape="zmatrix", ledtype='WS2812', brightness=30)

for x in range(strip.getWidth):
    strip.setLEDs(led=(x, 0), rgb=(128, 0, 0))

strip.showLEDs()

try:
    while True:
        strip.shift("up", 2)
        strip.showLEDs()
        time.sleep(0.2)

except KeyboardInterrupt:
    strip.clearLEDs()
    strip.showLEDs()
    del strip
```

4.3 Mirroring in an Axis

4.3.1 Mirroring LED Strings

```
#!/usr/bin/env python3

import time

from pixelpi import Strip

strip1 = Strip(1, 180, ledtype='WS2811_GRB', brightness=50)
strip2 = Strip(2, 180, ledtype='WS2811_GRB', brightness=50)
strip3 = Strip(3, 180, ledtype='WS2811_GRB', brightness=50)
strip4 = Strip(4, 180, ledtype='WS2811_GRB', brightness=50)

strip1.clearLEDs()
pattern = strip1.getLEDs()

for pixel in range(int(len(pattern) / 2)):
    pattern[pixel] = [255, 0, 0, pattern[pixel][3]]
    pattern[strip1.getLength - pixel - 1] = [0, 0, 255, pattern[strip1.getLength -␣
↪pixel - 1][3]]
```

(continues on next page)

(continued from previous page)

```
for strip in [strip1, strip2, strip3, strip4]:
    strip.setLEDs(pattern=pattern)

try:
    while True:
        for strip in [strip1, strip2, strip3, strip4]:
            strip.mirror()
            strip.showLEDs()
        time.sleep(1)

except KeyboardInterrupt:
    for strip in [strip1, strip2, strip3, strip4]:
        strip.clearLEDs()
        strip.showLEDs()
        del strip
```

4.3.2 Mirroring LED Matrices

```
#!/usr/bin/env python3

import time

from pixelpi import Strip

strip = Strip(4, (8, 32), shape="zmatrix", ledtype='WS2812', brightness=30)

for x in range(int(strip.getWidth / 2)):
    for y in range(strip.getHeight):
        strip.setLEDs(led=(x, y), rgb=(128, 0, 0))
        strip.setLEDs(led=(strip.getWidth - 1 - x, y), rgb=(0, 0, 128))

strip.showLEDs()

try:
    while True:
        strip.mirror("horizontal")
        strip.showLEDs()
        time.sleep(0.2)

except KeyboardInterrupt:
    strip.clearLEDs()
    strip.showLEDs()
    del strip
```

4.4 Displaying an Image

4.4.1 Displaying Images of LED Matrices

```
from time import sleep
```

(continues on next page)

(continued from previous page)

```
from PIL import Image
from pixelpi import Strip

im = Image.open("image.png").convert(mode='RGB', colors=256)

# strip1 = Strip(2, (8, 8), ledtype='SK6812_GRBW', shape="matrix", brightness=0.2)
strip = Strip(terminal=4, size=(8, 32), ledtype='WS2812', shape="zmatrix",␣
↪brightness=30)

try:
    i = 0
    while True:
        i = i + 1
        if i >= 32:
            i = 0

        for r in range(4):
            im = im.transpose(Image.ROTATE_90)
            strip.setLEDs(led=(0, i), image=im)
            strip.showLEDs()
            sleep(0.5)

        strip.clearLEDs()
        strip.showLEDs()

except KeyboardInterrupt:
    strip.clearLEDs()
    strip.showLEDs()
    del strip
```

Note You can of course display an image on an LED string by supplying a 1 by x image.

CHAPTER 5

Using the PixelPi Python Library

The PixelPi Library is intended to be used with the PixelPi PCB from Hut 8 Designs.

If you have not already installed the library, see *Installing the Python Library*

5.1 Creating a Strip Object

class pixelpi.**Strip**(*terminal*, *size*, *shape='straight'*, *ledtype='WS2812'*, *brightness=255*)

Creates an led Strip with the provided configuration.

Parameters

- **terminal** (*int*) – Which terminal the terminal is attached to (1-4).
- **size** (*int or tuple*) – The size of the LED string or matrix. Either the number of LEDs for a string, or (width, height) for a matrix.
- **shape** (*str, optional*) – The 'shape' of the LEDs attached:
  - `straight` - A LED string (default)
  - `reverse` - A LED string which starts at the opposite end.
  - `zmatrix` - A matrix where the LEDs in the first row go left to right, the next one right to left. i.e:

    1 2 3 4
    8 7 6 5
    9 . . .

  - `matrix` - a normal matrix where the led order goes left to right. i.e:

1 2 3 4
5 6 7 8
9 ...

- **ledtype** (*str*) – One of the supported LED types:

  `WS2812` *(default)*, `SK6812`, `SK6812W`, `SK6812_RGBW`, `SK6812_RBGW`, `SK6812_GRBW`, `SK6812_GBRW`, `SK6812_BRGW`, `SK6812_BGRW`, `WS2811_RGB`, `WS2811_RBG`, `WS2811_GRB`, `WS2811_GBR`, `WS2811_BRG`, `WS2811_BGR`

- **brightness** (*int*) – The default brightness for all LEDs (0-255).

getHeight
Returns the height of the matrix (or length of an LED string).

getLength
Returns how many LEDs are in the strip/matrix.

getStripNumber
Returns the terminal the LEDs are attached to.

getStripType
Returns the set LED type.

getWidth
Returns the width of the matrix (1 if an LED string).

updateStatus
Returns or sets whether output is currently enabled for the LEDs.

When set to `False`, the current LED pattern will remain unchanged even if the pattern is changed and *`showLEDs()`* is called.

Getter Returns the status.

Setter Sets the status.

Type bool

5.2 Setting LED Colours and Brightness

`Strip.`**setLEDs**(*led=None*, *rgb=None*, *brightness=None*, *image=None*, *pattern=None*)
Sets the RGB value, and optionally brightness, of one or more LEDs.

If `led` is not supplied or set to `None`, all LEDs will be set to the defined colour and brightness.

If a matrix is being used, `led` can either be the LED count from the first LED, or the (x, y) location.

If a brightness value is not supplied, or set to `None`, the current LED brightness will be kept.

If an image is supplied, then it must be in RGB format (see Pillow image file formats)

If a pattern is supplied, it must be a list consisting of a tuple with four elements, `(red, green, blue, brightness)`.

Parameters

- **led**(*int, tuple or None*) – The LED location or None to set all LEDs to the desired colour.

- **rgb** (*tuple or None*) – A tuple consisting of 3 elements, (red, green, blue), with each value being between 0 and 255.
- **brightness** (*int or None*) – A value between 0 (dim) to 255 (very bright) or None to take the default.
- **image** (*image or None*) – An image callingclass (see PIL or Pillow libraries) containing an RGB formatted image.
- **pattern** (*list or None*) – A list of the RGB and brightness values for each LED, in numerical order from the start of the strip/matrix.

Strip.**getLEDs**(*led=None*)

If led is supplied, returns the RGB and brightness values of a specific LED.

If led is not supplied or set to None a list of red, green, blue and brightness values for each LED is returned in a list.

Parameters **led** (*int, tuple or None*) – The led location, either the LED count from the start, or the x,y matrix location, or if None, a list of all LEDs will be returned

Returns (red, green, blue, brightness) or a list of (red, green, blue, brightness)

Strip.**clearLEDs**()

Clears the LEDs (sets them to black), leaving the brightness as it is.

5.3 Manipulating LED Colours

Strip.**shift**(*direction='UP', shift=1*)

Shifts the LEDs on the matrix or string by *shift* LEDs in the direction specified.

Parameters

- **direction** (*str*) – The direction the LEDs should shift (LEFT, RIGHT, UP, DOWN)
- **shift** (*int*) – The number of LEDs the matrix/string should be moved in the specified direction.

Strip.**mirror**(*mirror='VERTICAL'*)

Mirrors the matrix LEDs either in the VERTICAL or HORIZONTAL plane.

Reverses the LED colours on an LED string. The mirror parameter is not required for strings.

Parameters **mirror** (*str*) – VERTICAL *(default)* or HORIZONTAL

5.4 Updating LEDs

Strip.**showLEDs**()

Once you have set the colours of the string/matrix LEDs, use *showLEDs* to update the LEDs.

CHAPTER 6

The PixelPi Button

6.1 Using the Button

class pixelpi.**PixelPiButton**(*callingclass=None*, *shortpresstime=0.5*, *shortpress=None*, *longpresstime=2.0*, *longpress=None*)

A class to handle the button on the PixelPi board for the Raspberry Pi.

If no parameters are supplied, the Raspberry Pi will be rebooted after the button has been held for 0.5 seconds, or will be shut down if the button is held for 2 seconds or all.

Alternatively, a local Class can be created with two methods to handle a short and a long press. Create an instance of the class and supply the class, method names and press-times to this class. For example:

```
class MyButtons:
def __init__(self, strips=None):
    self.__strips = strips

def clear(self):
    try:
        if self.__strips is not None:
            for strip in self.__strips:
                strip.clearLEDs()
                strip.showLEDs()
    except:
        raise AttributeError('The strip list contained an error.')

def whitelights(self):
    try:
        if self.__strips is not None:
            for strip in self.__strips:
                strip.setLEDs(rgb=(255, 255, 255))
                strip.showLEDs()
    except:
        raise AttributeError('The strip list contained an error.')
```

Create an instance of the MyButtons class:

```
mybuttons = MyButtons([strip1, strip2, strip3, strip4])
```

Then create an instance of the PixelPiButton class, passing in the local class and methods:

```
button = PixelPiButton(callingclass=mybuttons, shortpresstime=0.5, shortpress=
↪"clear", longpresstime=1.0,
longpress="whitelights")
```

When the button is held for shortpresstime, the 'clear' method will be executed, unless it is held for longpresstime after which 'whitelights' will be called.

Parameters

- **callingclass** (*object*) – The 'local' class.
- **shortpresstime** (*float*) – The short press time.
- **shortpress** (*str*) – The method in class `callingclass` that is called after the button has been pressed for shortpresstime seconds.
- **longpresstime** (*float*) – The long press time.
- **longpress** (*str*) – The method in class `callingclass` that is called after the button has been pressed for longpresstime seconds.

6.2 Example

```
#!/usr/bin/env python3

import colorsys
import time

from pixelpi import Strip, PixelPiButton


class MyButtons:
    def __init__(self, strips=None):
        self.__strips = strips

    def clear(self):
        try:
            print("clearing strips")
            if self.__strips is not None:
                for leds in self.__strips:
                    leds.clearLEDs()
                    leds.showLEDs()
        except:
            raise AttributeError('The strip list contained an error.')

    def whitelights(self):
        try:
            if self.__strips is not None:
                for strip in self.__strips:
                    strip.setLEDs(rgb=(255, 255, 255))
                    strip.showLEDs()
```

(continues on next page)

(continued from previous page)

```
        except:
            raise AttributeError('The strip list contained an error.')


# Change the terminal type to the type you have
strip1 = Strip(1, 256, ledtype='WS2811_GRB', brightness=30)
strip2 = Strip(2, 256, ledtype='WS2811_GRB', brightness=30)
strip3 = Strip(3, 256, ledtype='WS2811_GRB', brightness=30)
strip4 = Strip(4, 256, ledtype='WS2811_GRB', brightness=30)

mybuttons = MyButtons([strip1, strip2, strip3, strip4])

button = PixelPiButton(callingclass=mybuttons, shortpress="clear", longpress=
↪"dosomethingelse")

spacing = 360.0 / 16.0
hue = 0

try:
    while True:
        hue = int(time.time() * 100) % 360
        for x in range(256):
            offset = x * spacing
            h = ((hue + offset) % 360) / 360.0
            r, g, b = [int(c * 255) for c in colorsys.hsv_to_rgb(h, 1.0, 1.0)]

            for strip in [strip1, strip2, strip3, strip4]:
                strip.setLEDs(rgb=(r, g, b), led=x)

        for strip in [strip1, strip2, strip3, strip4]:
            strip.showLEDs()

        time.sleep(0.001)

except KeyboardInterrupt:
    for strip in [strip1, strip2, strip3, strip4]:
        strip.clearLEDs()
        strip.showLEDs()
        del strip
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index