Inverted LED Display

Apr 2017

I made a mistake

When I designed the PCB for this project. I did what most people would do and I made the PCB the 'right way up' with 40 Way Raspberry pi header and the 7 Segment LED also the normal way up (so you can read it). Later on I realised this was a bit of a mistake, because this means the Raspberry Pi Zero Power and USB connections are now at the bottom, so you can't power this up and put it on a table :(.

Fear not though, with some help I have edited the code so that it writes the time upside-down, so if you rotate your Raspberry Pi 180° you can read the time correctly, and still put it on the table. Below is the adjusted code.

Please NOTE: You do NOT have to rotate the 7 Segment LED on the PCB, the code does this for you.

Above we can see the Clock working in 'Normal' mode on the left, and in 'Inverted' mode on the right.

<sxh [py][; options for SyntaxHighlighter]> #! /usr/bin/python # Python Script to run a 7 Segment Common Anode LED as a clock. # Version 1.0 import RPi.GPIO as GPIO import time

#Define numbering system for the IO pins Raspberry Pi GPIO.setmode(GPIO.BCM)

Define GPIO ports for the 7seg gpioSegments = (5,6,13,19,26,21,20,16) # 7seg_segment_pins (11,7,4,2,1,10,5,3) + 100R inline # Setup channels for output and set initial values for segment in (gpioSegments):

```
GPIO.setup(segment,GPIO.OUT)
GPIO.output(segment,GPIO.HIGH)
```

Define GPIO ports for the digits 0-3 gpioDigits = (18,23,24,25) gpioDigits = (25,24,23,18) # Pins (12,9,8,6) select digits 0-3 respectively reading LTR on display # Setup channels for output and set initial values for digit in (gpioDigits):

```
GPIO.setup(digit,GPIO.OUT)
GPIO.output(digit,GPIO.LOW)
```

#Define flags for comtrol of Colon separator on display colon_visible=True colon_counter=0

#Define segment arrays for each number to be displayed numbers = {

```
'0':(0,0,1,0,1,0,0,0),
  '1':(0,1,1,1,1,1,0,1),
  '2':(0,0,1,1,0,0,1,0),
  '3':(0,0,1,1,0,1,0,0),
```

```
  '4':(0,1,1,0,0,1,0,1),
  '5':(1,0,1,0,0,1,0,0),
  '6':(1,0,1,0,0,0,0,0),
  '7':(0,0,1,1,1,1,0,1),
  '8':(0,0,1,0,0,0,0,0),
  '9':(0,0,1,0,0,1,0,0),
  ' ':(0,0,0,0,0,0,0,0)}
```

#Cycle through each digit and its segments try:

```
  while True:
      time_string = str(time.ctime()[11:13]+time.ctime()[14:16]).rjust(4)
      for digit in range(4):
```

#select digit to display

```
          GPIO.output(gpioDigits[digit], 2)
```

#set required segments on

```
          for segment in range(0,8):
              GPIO.output(gpioSegments[segment], numbers[time_string[digit]][segment])
```

#check to see if we are on segment 3 of digit 2(LTR)

```
              if ((digit==2) and (segment==2)):
```

#when colon counter gets to set value flip colon display mode between TRUE (visible) and False (NOT.visible)

```
                if (colon_counter<=25):
```

#count value not reached turn colon ON or OFF based on current setting of colon_visible

```
                    if colon_visible==True:
                        GPIO.output(13, 0)
                    else:
                        GPIO.output(13, 1)
```

#colon counter set value reached so flip colon display mode and reset colon counter

```
                else:
                    colon_counter = 0
                    colon_visible =  not colon_visible
```

#display all selected segments for a short time

```
          time.sleep(0.005)
```

#turn-off All segments

```
          for segment in range(0,8):
              GPIO.output(gpioSegments[segment],GPIO.HIGH)
```

#turn-off current digit selector pin

```
          GPIO.output(gpioDigits[digit], GPIO.LOW)
```

#advance colon counter

```
      colon_counter=colon_counter+1
```

finally:

```
GPIO.cleanup()
```

</sxh>