40-Pin

7-Segment Display (TM1637) 5a

A cheap, simple, and effective way to provide user feedback and information from embedded systems and existing projects.

Teaches: Defensive programming

60 mins

1. What it does

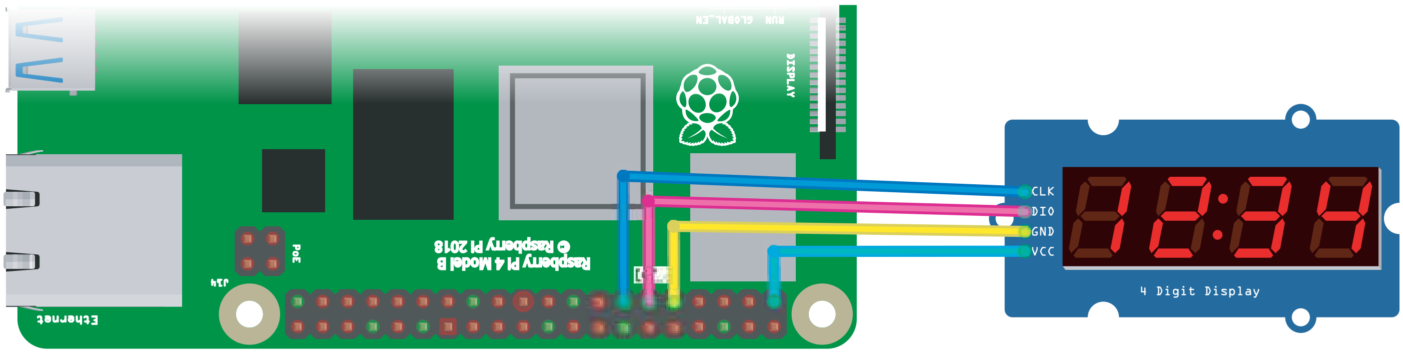
Seven-segment displays are everywhere – telling us the time, temperature, or counting-down the days to major events. This tutorial shows you how to display numbers and basic text characters on the common TM1637 module which features four seven-segment displays.

2. Component list

1 x TM1637 four x seven-segment display module

4 x Female to female jumpers

3. Build the circuit



Although the TM1637 display module states that 5v is required, you should use the 3.3v GPIO instead. This is particularily important when using a Raspberry Pi 4 which may overload the TM1637.

4. Download the TM1637 Library

Open a new Terminal window. Use the `CD` command to change the current directory to the one where you will be storing your Python program. Now enter the following command to download the TM1637full library:

```
wget https://www.mr.langford.co/libraries/tm1637fullBCM.py
```

You should now have a file called `tm1637fullBCM.py` in your folder. Type `ls` to check.

Please note: tm1637fullBCM is based on a library by Richard IJzermans, available at:

https://raspberrytips.nl/tm1637-4-digit-led-display-raspberry-pi/ Please visit the website and support the original author.

5. Test your wiring and library

Now we will test the wiring and library. Create a new Python program in the same folder as your tm1637full library. Type the code below:

```
1.  # Import libraries
2.  from time import sleep
3.  import tm1637fullBCM
4.
5.  # Set up display
6.  CLK = 13 # Clock
7.  DIO = 11 # Digital Input / Output
8.  Display = tm1637fullBCM.TM1637(CLK, DIO, tm1637fullBCM.BRIGHT_TYPICAL)
9.
10. # Clear display
11. Display.Clear()
12.
13. # Set brightness (0 - 7)
14. Display.SetBrightness(7)
15.
16. # Display colon LED
17. Display.ShowDoublepoint(1)
18.
19. # Define output characters
20. outputData = "1234"
21.
22. # Split outputData into separate characters
23. a = outputData[0]
24. b = outputData[1]
25. c = outputData[2]
26. d = outputData[3]
27.
28. # Display data
29. Display.Show([a,b,c,d])
```

Set a **1** to light the colon in the centre of the display, or a **0** to turn it off.

You can enter any number, capital letter, space, dash, or an asterisk here, but it must be a string.

6. Defensive programming

When you have successfully run the code above, it's time to add some more functionality. The code is very dependent on line 20 containing the correct number of characters stored as a string. If a user was to enter more characters, unsupported characters, or an integer, the output may be undesirable or the program might even crash!

Defensive programming is a term used to describe code which continues to function correctly even in unforseen circumstances. Error messages should help the user understand their mistake and avoid repeating it.

Modify your program so that:

1. An input command allows the user to enter the data, rather than coding it directly into the program.
2. If the user enters more or less than four characters, they will be asked to try again until they enter the correct number of characters.
3. Lower-case characters are automatically converted to capital letters
4. If the user enters an unsupported character, they are asked to try again.

7. Project ideas

There are lots of projects which you can use seven-segment displays with. Here are some ideas:

- **Countdown timer / stopwatch** – Add a button to start the counter. Add a buzzer which sounds when the countdown reaches zero. Additional buttons could increase time or reset the clock.
- **Scoring system** – Modify your Whack-A-Mole or Simon games (projects 3 and 4) to add a score. Add a second TM1637 to show the current high score.
- **Distance measure** – Add an ultrasonic distance sensor and display how far away objects are.