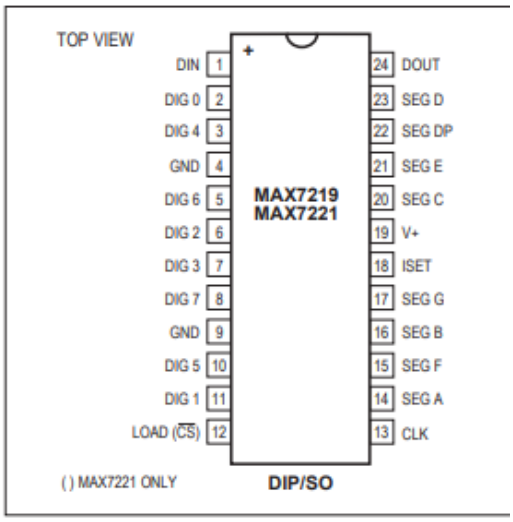
CP320 - Exploration Project

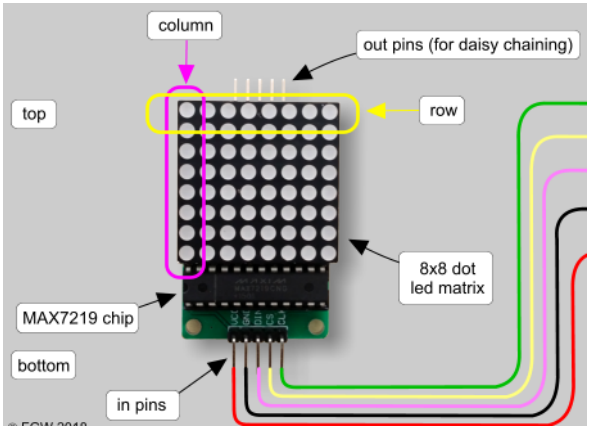
Daniel Nguyen - 201460890

Overview

The MAX7219 Matrix serves as a common-cathode display driver with serial input/output capabilities, facilitating the connection of microprocessors to 7-segment numeric LED displays featuring up to 8 digits, bar-graph displays, or arrays of 64 individual LEDs. Its on-chip components include a BCD code-B decoder, multiplex scan circuitry, segment and digit drivers, as well as an 8x8 static RAM responsible for storing each digit.

Pin Configuration





I wrote a program that displays a user-inputted single character in the middle of the matrix. This includes any number, letter, or symbol entered from the prompt in the console. Depending on the type of character entered, a different animation will be shown around the border of the matrix. There are three animations, one lights up each corner, another has trailing dots circling around the matrix, and the last one has one single dog circling the display.
My code also demonstrates how I utilized threading to ensure the user could enter their inputs while the other functions to display were still running.

Packages/Libraries Needed

- https://luma-led-matrix.readthedocs.io/en/latest/index.html
  - Python library interfacing LED matrix displays with the MAX7219 driver (using SPI) and WS2812 & APA102 NeoPixels (inc Pimoroni Unicorn pHat/Hat and Unicorn Hat HD) on the Raspberry Pi and other Linux-based single board computers

```
from luma.core.interface.serial import spi, noop
from luma.core.render import canvas
from luma.led_matrix.device import max7219
from luma.core.legacy import text
from luma.core.legacy.font import TINY_FONT
```

- https://docs.python.org/3/library/threading.html
    - Python module for thread-based parallelism

```
import threading
```

- https://docs.python.org/3/library/time.html
    - Python module for time access and conversions

```
import time
```

Challenges and/or Issues

Challenge: Installing the custom libraries on the raspberry pi. Usually on a computer computer, you would simply connect to the internet then use a command to install the library or custom packages. For a raspberry pi, the WiFi was not automatically disabled and installing the custom libraries needed a different approach
Solution: Added SSID and password for local WiFi to the raspberry pi. Enabled SPI interface on raspberry pi. Installed library using PIP install via WiFi

Challenge: Creating an LED pattern on the device. There are built in functions from the library to use patterns but creating a unique pattern requires knowledge about device functionality.
Solution: Mapping the device and testing which orientation and LEDs light up. Light individual LEDs using loops to create a cascading effect and pattern.

Challenge: Run several operations in a single program. Asking the user for input caused the LED patterns to pause.
Solution: Taking CP386 at the moment allowed me to learn about threading in C. After researching python threading, I was able to use its functionality to apply my knowledge in the other class to run multiple functions at once, keeping the LED pattern as well as user prompt for input.

3 Useful URLs

- https://www.analog.com/media/en/technical-documentation/data-sheets/max7219-max7221.pdf
    - MAX7219 Data Sheet is useful for the technical aspect of the LED matrix. It allows for information on the mechanical and electrical components including the

control requirements of this device. This is important to ensure the device operates as specified and optimally. The characteristics and feature information allows for generating a program to utilize these functions e.g. brightness, matrix/segment controls, interface, etc.

- https://github.com/rm-hull/luma.led_matrix/tree/master/examples
    - This repository by Richard Hull contains a great deal of information for the device and examples of how to use the library. There are different operating modes, available fonts, built in functions, an emulator, and descriptive images displaying the device.
- https://github.com/stechiez/raspberrypi-pico/tree/main/pico_max7219
    - This repository has additional information and more images displaying the connection from the raspberry pi to the device. There are sample codes to display different functions of the matrix and lighting individual LEDs. More examples with video examples (https://www.youtube.com/watch?v=0oSXJf_RgPA)

Code

```
#Import libraries
from luma.core.interface.serial import spi, noop
from luma.core.render import canvas
from luma.led_matrix.device import max7219
from luma.core.legacy import text
from luma.core.legacy.font import TINY_FONT
import time
import threading

#Initialize max7219 device
serial = spi(port=0, device=0, gpio=noop())
device = max7219(serial)

INPUT_CHARACTER = "  "

#Device function to display 'ring pattern' via threading
def max_led_pattern():
    global INPUT_CHARACTER


    #Three different scenarios if character is a digit, alpha, or
```

```
other
    if INPUT_CHARACTER.isdigit():
        numeric_display()

    elif INPUT_CHARACTER.isalpha():
        alpha_display()
    else:
        other_display()
    return

def numeric_display():
    matrix_edges = 7
    matrix_start_position = 0
    max_matrix_sleep_seconds = 0.01
    #Top vertical
    for trailing_count in range(matrix_edges):
            with canvas(device) as draw:
                for i in range(matrix_edges):

draw.point((matrix_start_position,trailing_count-i), fill="white")
                    time.sleep(max_matrix_sleep_seconds)
            character_display()
    #Left horizontal
    for trailing_count in range(matrix_edges):
            with canvas(device) as draw:
                for i in range(matrix_edges):
                    draw.point((trailing_count-i,matrix_edges),
fill="white")
                    time.sleep(max_matrix_sleep_seconds)
            character_display()
    #Bottom vertical
    for trailing_count in range(matrix_edges, -1, -1):
            with canvas(device) as draw:
                for i in range(matrix_edges):
                    draw.point((matrix_edges,trailing_count+i),
fill="white")
                    time.sleep(max_matrix_sleep_seconds)
            character_display()
    #Right horizontal
```

```
    for trailing_count in range(matrix_edges, -1, -1):
            with canvas(device) as draw:
                for i in range(matrix_edges):
                    draw.point((trailing_count+i,0), fill="white")
                    time.sleep(max_matrix_sleep_seconds)
            character_display()
    return

def alpha_display():
    matrix_edges = 7
    matrix_start_position = 0
    max_matrix_sleep_seconds = 0.01
    #Top vertical
    for trailing_count in range(matrix_edges):
            with canvas(device) as draw:
                for i in range(matrix_edges):
                    draw.point((matrix_start_position,0),
fill="white")
                    time.sleep(max_matrix_sleep_seconds)
            character_display()
    #Left horizontal
    for trailing_count in range(matrix_edges):
            with canvas(device) as draw:
                for i in range(matrix_edges):
                    draw.point((0,matrix_edges), fill="white")
                    time.sleep(max_matrix_sleep_seconds)
            character_display()
    #Bottom vertical
    for trailing_count in range(matrix_edges, -1, -1):
            with canvas(device) as draw:
                for i in range(matrix_edges):
                    draw.point((matrix_edges,7), fill="white")
                    time.sleep(max_matrix_sleep_seconds)
            character_display()
    #Right horizontal
    for trailing_count in range(matrix_edges, -1, -1):
            with canvas(device) as draw:
                for i in range(matrix_edges):
                    draw.point((7,0), fill="white")
```

```
                    time.sleep(max_matrix_sleep_seconds)
            character_display()
    return

def other_display():
    matrix_edges = 7
    matrix_start_position = 0
    max_matrix_sleep_seconds = 0.01
    #Top vertical
    for trailing_count in range(matrix_edges):
            with canvas(device) as draw:
                for i in range(matrix_edges):

draw.point((matrix_start_position,trailing_count), fill="white")
                    time.sleep(max_matrix_sleep_seconds)
            character_display()
    #Left horizontal
    for trailing_count in range(matrix_edges):
            with canvas(device) as draw:
                for i in range(matrix_edges):
                    draw.point((trailing_count,matrix_edges),
fill="white")
                    time.sleep(max_matrix_sleep_seconds)
            character_display()
    #Bottom vertical
    for trailing_count in range(matrix_edges, -1, -1):
            with canvas(device) as draw:
                for i in range(matrix_edges):
                    draw.point((matrix_edges,trailing_count),
fill="white")
                    time.sleep(max_matrix_sleep_seconds)
            character_display()
    #Right horizontal
    for trailing_count in range(matrix_edges, -1, -1):
            with canvas(device) as draw:
                for i in range(matrix_edges):
                    draw.point((trailing_count,0), fill="white")
                    time.sleep(max_matrix_sleep_seconds)
            character_display()
```

```
    return

def character_display():
    #Sleep time for each LED segment
    global INPUT_CHARACTER
    max_function_sleep_seconds = 0.1

    with canvas(device) as draw:
        #Points (2,0) is offsetting to the center
        text(draw, (2, 0), INPUT_CHARACTER, fill="white",
font=TINY_FONT)
        time.sleep(max_function_sleep_seconds)


    return

def matrix_thread():
    global INPUT_CHARACTER
    max_function_sleep_seconds = 0.1
    while True:

        with canvas(device) as draw:
        #Points (2,0) is offsetting to the center
        text(draw, (2, 0), INPUT_CHARACTER, fill="white",
font=TINY_FONT)
        time.sleep(max_function_sleep_seconds)
        #Range(8) because 8 there are 8 leds
        for trailing_count in range(8):
            for x in range(8):
                with canvas(device) as draw:
                    character_display()
                    max_led_pattern()



#Main function
def main():
    threadMaxMatrix = threading.Thread(name='matrix_thread', target =
matrix_thread)
```

```
    threadMaxMatrix.start()

    while True:
        #Character to be displayed from user input
        #Error checking for a single character
        global INPUT_CHARACTER
        INPUT_CHARACTER = input("Enter a character to display: ")
        if len(INPUT_CHARACTER) > 1:
            print("Error: you must enter a single character (e.g.
'A')")
        #Runs character_display function
    return


main()
```