

This repository


Search


Pull requests


Issues

Marketplace


Gist




 [greenfox-academy](#) / [teaching-materials](#) Private

 Watch

13


 Star


4


 Fork


22

<> Code

 Issues 11

 Pull requests 0

 Projects 0

 Wiki

Insights ▾

Branch: master ▾


[teaching-materials](#) / [project](#) / [hardware](#) / [led-matrix](#) /

Create new file


Upload files

Find file


History

 [koincidencia](#) feat(hw-led-matrix): add task Latest commit 69d96d1 4 hours ago


..

 [workshop/Projects/STM32746G-Discovery/GreenFox](#)


feat(hw-led-matrix): add template4 hours ago

 [README.md](#)


feat(hw-led-matrix): add task4 hours ago

 [SKILL-IO.md](#)

feat(hw-led-matrix): add materials3 days ago

 [workshop.zip](#)

feat(hw-led-matrix): add zip4 hours ago

 [README.md](#)

LED matrix controller

Create a multithreading LED matrix controller

Objectives

- Learn inter-thread communication methods
- Practice STM32 peripheral usage

Materials & Resources

Material	Duration
Multi-dimensional Arrays in C	-
Electronic Basics #5: How to Multiplex	4:53
Lecture 1, unit 1: Introduction to Concurrency	12:02
Lecture 1, unit 2: Races and synchronization	7:04
Lecture 1, unit 3: Locks	8:59
Lecture 2, unit 1: Introduction to Semaphores	7:50
Lecture 2, Unit 2: semaphore values	5:43
Lecture 4, unit 1: Deadlock	4:54
Mutex Lock	1:17
CMSIS-OS mutex	-
CMSIS-OS message queue	-

Material review

- LED matrix multiplexing
- 2D arrays
- Race condition
- Mutex
- Semaphore
- Deadlock

- Message queue
- Other inter-thread communication methods
 - Memory pool
 - Signal events
 - Mail queue

Workshop

[LED matrix datasheet](#)

Based on the provided LED matrix datasheet and on the User manual of the STM32F746G-DISCOVERY board make the connections between them. Don't forget to connect 100R resistors in series with every LED!

If you are ready to go further then ask a mentor to verify the hardware. You can ruin your board with a simple mistake!

LED matrix driver with multiple threads

The first task is to write the driver or the LED matrix. Use the provided template project! You will find `TODO:` comments in the code.

The goal is to be able to run simple animations on the LED matrix. The program should store each LED state in a 2D array. A thread should continuously update the LED matrix based on the information in the 2D array. The animation should run on a different thread and should only modify the state storage 2D array.

Because the updater thread and the animation thread is accessing the same memory (the 2D array), they can mess up each other's data. Hence please use a mutex to protect the 2D array from multiple simultaneous access!

Change animation speed with a potentiometer

Connect a potentiometer to the board (don't forget that the board runs on 3.3V voltage!) onto the `A0` pin.

Add a new thread to your program. This thread should read the voltage of the potentiometer with an ADC and put the data into a message queue continuously.

Modify the animation thread to get the ADC data from the message queue, and based on that value modify the speed of the animation.

Dot based on touch panel coordinate

Based on the touch screen data show a dot on the LED matrix. For example if the user touches the bottom left corner of the screen the bottom left LED should be lit on the matrix.

Characters on the matrix

Make the program capable of displaying characters on the LED matrix! You can use lookup tables or any other methods to solve the task.

Characters from the network

Start a socket server in an other thread and show the received string on the LED matrix!

Solution

[Solution](#)

