

RMIT University

EEET2490 – Embedded System: OS and Interfacing, Semester 2023-2

ASSESSMENT 3 – GROUP WORK

OBJECTIVES

In this group project, students as a team work together to further sharpen their skills and enrich knowledge in OS and peripheral driver development.

DESCRIPTION

Students should read all of the tasks below before starting of implementation, since one task may support another task.

1. CONTINUED DEVELOPMENT FOR BARE METAL OS (30%)

a) Display image and video on the screen

So far, you have learnt all the basics about screen display, and can work well with it to draw with specific position and color. It is quite easy to draw a line, rectangle, circle on the screen. However, how about displaying images and videos?

In this task,

- i. Select a nice image of which size is larger than screen size and display it on screen that we can use **'w' and 's' keys** to scroll up and down.
- ii. Select three images and display on screen as a **slideshow** (only one image displayed on screen at a time, however, we can use **'a' and 'd' keys** to switch between them).
- iii. We can display images, how can we display a **video** on the screen? Think of it and find the answer for yourself. Select a video and display it on the screen. For the purpose of demonstrating, you only need to display a very-short video (just several seconds).

Supporting Resource:

- This tool can help you to convert an image into ARGB32 data (select RGB888 option), which is compatible with our Raspberry Pi: <https://javl.github.io/image2cpp/>

Note that, the tool convert an image into an array of data, in which each element is 32-bit data for each pixel. Follow the steps below

1. Select Image

All processing is done locally in your browser; your images are not uploaded or stored anywhere online.

Choose Files 17_PR_SYBO.jpg
17_PR_SYBO.jpg remove

2. **Setting for Output: select 3 bytes per pixel (rgb888)**

Code output format

Arduino code

Adds some extra Arduino code around the output for easy copy-paste into [this example](#). If multiple images are loaded, generates a byte array for each and appends a counter to the identifier.

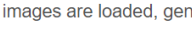
Identifier/Prefix:

epd_bitmap_

Draw mode:

Horizontal - 3 bytes per pixel (rgb888)

If your image looks all messed up on your display, like the image below, try using a different mode.



Swap bits in byte:

☐ swap

Useful when working with the `u8g2` library.

3. Image Settings: select Black for Background color

Canvas size(s): x glyph

Background color: ☐ White ☒ Black ☐ Transparent

Invert image colors ☐

Dithering:

Brightness / alpha threshold:

0 - 255; if the brightness of a pixel is above the given level the pixel becomes white, otherwise they become black. When using alpha, opaque and transparent are used instead.

4. **Check Preview:** *it should show your uploaded image with correct color. If not, reload the webpage and start again*



5. **Generate Code:** *it should now give you the pixel data of the image*

[Generate code](#)
[Copy Output](#)
[Download as binary file \(.bin\)](#)

[illegible]

- Sample **video** can be found here (or you can find your own video): <https://sample-videos.com/>

Hint: to display a video, it is simply just to display all frame images of it. Thus, use a tool to extract frame images from your video. Search for the keyword “**extract frames from video**” and there is a plenty of tools and ways to do so.

b) Implement a font to display text on the screen

Besides images and videos, displaying a text on screen is a conventional need. Basically, it is similar to the way that we draw any thing on the screen – we need to draw at corresponding pixels. However, to make it simple, we should have **FONT** which defines pattern of each single character [glyph](#).

The pixel data for the character glyphs is usually stored as a bitmap. In a bitmap, each pixel is represented by a single bit. If the bit is 'ON' (value 1), the pixel is to be drawn in the foreground color; if 'OFF' (value 0), the pixel is set to the background color (hidden).

Example of letter “A” in an 8x8 FONT:

```
0 0 0 0 1 1 0 0 = 0x0C
0 0 0 1 1 1 1 0 = 0x1E
0 0 1 1 0 0 1 1 = 0x33
0 0 1 1 0 0 1 1 = 0x33
0 0 1 1 1 1 1 1 = 0x3F
0 0 1 1 0 0 1 1 = 0x33
0 0 1 1 0 0 1 1 = 0x33
0 0 0 0 0 0 0 0 = 0x00
```

The letter A above can be represented by just an array of 8 bytes : {0x0C, 0x1E, 0x33, 0x33, 0x3F, 0x33, 0x33, 0x00}.

Requirement:

- I. Get an existing **font** image, such as at <https://fontmeme.com/fonts/static/235021/autolova-font-character-map.png> (you don't need to draw it by yourself), convert into pixel data and implement it for your OS.
- II. Display names of all team members on the screen. You should use several different colors for the text.

Supporting Resource: those tutorials listed below could be a good reference to start:

[1] https://github.com/bztsrc/raspi3-tutorial/tree/master/0A_pcscreenfont

[2] <http://cs107e.github.io/labs/lab6/>

[3] <https://isometimes.github.io/rpi4-osdev/part5-framebuffer/>

2. APP DEVELOPMENT FOR BARE METAL OS (70%)

Develop a small game that player(s) use screen and keyboards to play the game.

Your game should have colors (rather than just black and white). Basic requirement is to only have one fixed stage, excellent goods should have at least two stages.

Criterion for evaluation:

- Complexity of the game
- How well it is written (variables, resources usage, coding style...)
- Creativity: a little of creativity is welcome, but not compulsory. It is okay to transfer from existing game (but don't completely copy and paste source code of existing games or works of other students).

*Note that it is required to **test with the real board** for both Task 1 and 2 (after you complete the development with QEMU).*

Advice: Developing a small game is actually not difficult. If you haven't developed any game before, my advice is to try making a simple game on an drag-and-drop platform like Scratch
<https://scratch.mit.edu/>

Try with **Make an Apple Catcher Game:** www.youtube.com/watch?v=jFVJdRLZoQ4

Or **How to Make a Shooter Game:** www.youtube.com/watch?v=QXru0rSV2ZQ

Implement a game with the guide above in Scratch to understand how we can make a game with Background image, sprites (player, opponent, obstacles or other objects in your game), and their interactions. Then, you can think of your own game and implement it in C in a similar way.

Report and Demonstration

Complete your **report** with discussion of all parts. You should provide some background information, explanation for your work, and also include screenshots of output as evidence of a successful outcome. *A discussion on success and limitation of the outcome would be also meaningful.*

For demonstration, please record a **video** of maximum 20 mins that demonstrate and explain for your work. All members should present in the demo video.