CS132 Lab Session 6-7-8

Dr A. Hague and Dr R. Suma,
adapted from Dr R. Kirk

24/11/2025

Welcome to a new CS132 lab. In this and the coming labs you will be working in teams at specific projects. The aim of this project is to build the classic 2-player game **Pong**, using the LED Panel provided and the STM32 controllers (and joysticks) discussed in Lab 5.
If you have not completed Lab 5, it is **strongly recommended** that you do before continuing the project. By the end of the project, you should have two paddles that you can control with the joysticks, a ball that bounces off the paddles as well as the top and bottom of the screen, and some way to display the current score (either like a 7-segment display or as a bar).

REMINDER!

Please do not pull out anything out of the computer, the LED Panel or the STM32 Controller! There are a limited number of kits, and they can be fragile. As such, they can take a long time to repair. Please be careful when handling the equipment.

1 LED Display Pins

Table 1: STM32 Pin Mapping for LED Panel

STM32 Port/Pin	**LED Panel Pin**	**Description**
C6 (output)	Input signal	This allows for the right-most pixel to be set to 0 or 1
C7 (output)	Clock	This pulses the clock for the selected row
C8 (output)	Latch	This determines whether the new memory be displayed (active low)
C2 – C5 (outputs)	Row selection	Binary representation of which of the 16 rows to select

2 How the LED Panel works

The panel is split into 2 smaller panels, each 32 pixels long and 16 pixels high. The top half of the panel is the primary panel and is where the inputs from the STM32 originate from. Each row and each colour in the LED Panel acts like a **shift register**. Each shift register starts on the right hand side of the row, and will shift the data left with each clock pulse. To start off, the red register is first, followed by blue, followed by green. Once it has passed through all the colours, the data moves down to the

corresponding row on the second half of the panel. This follows the same pattern of red, blue and green shift registers. Therefore, you can think of the panel as having **16 × 192 bit shift registers** (3 for the colours × 2 for the two halves of the panel × 32 pixels in each row).
Row selection is done using `C2` through `C5`, where `C5` is the most significant bit and `C2` is the least significant bit. The binary representation corresponds to a row. Therefore, if you would like to change the row you are operating on, change `C2` through `C5`.
You don't want to display the memory whilst it is updating. Therefore, the panel has a latch which keeps the current memory visible whist the underlying data is updated. You can then deactivate the latch to display the new memory. If you set the latch to always be low/clear, the latch will hold and will not display any new information. If you set the latch to always be high/set, the memory will always be shown as the latch will never be set. **The latch is active low.**

3 Where to start with the LED Panel

1. Set the latch pin. This will mean that all the data is displayed, but that can be useful for debugging purposes.
2. Clear all the row selection pins. This will mean you are working with rows 0 and 16.
3. Clear the clock pin. This will tell the LED Panel to get the next piece of data.
4. Then set the input pin. This will set the first memory cell in the first row to be on.
5. Set the clock pin. This will push the results of the input pin into memory. You should see the top right LED go red.
6. Repeat Steps 3 through 5 until you have all the LEDs at the top of the board turn red.
7. Repeat Steps 3 though 5 again. The top right hand pin should turn magenta (red and blue).
8. Repeat Step 6 until all the LEDs are magenta.
9. Repeat Steps 3 through 5 again. The top right hand pin should turn white (red, blue and green).
10. Repeat Step 6 until all the LEDs are white.
11. Repeat Steps 3 through 5. The LED on the right of row 16 (just over half way down) should go red.
12. You can make all the LEDs on rows 0 and 16 turn white by repeating Steps 6 through 10.
13. Select a new row by changing `C2`, `C3`, `C4` or `C5`.
14. Repeat Steps 3 through 12 for that row to make it go red, then magenta, then white.

To make your life easier, it might be worth building functions that can perform the following actions:

- A function that performs an input and a clock pulse
- A function that takes 32 bits (say, an integer) and performs a clock pulse for each binary input (you might need to look up how to perform binary shifts in C for this...)

- A function that takes 32 bits **and a row number** and sets the row before calling the previous function
- A function that takes **6x 32 bits** and a row number, one for red, green and blue for each half of the display, and calls the previous function 6 times, one for each colour and sub-panel. The function should latch the display before updating, and stop latching once all the data has been passed.

4 Building the game

There are many ways to start building the game. One of the easiest places to start is to build functions that can display the paddles. This can be done by setting particular pixels to a given colour, using the previously mentioned functions. This should give you a good idea on how they can be adjusted so the paddles can be moved up and down. From Lab 5, you should be able to detect the minimum and maximum of each joystick direction, so it is a case of remapping the value read from the joystick between its minimum and maximum value to between 0 and 1. You can adjust the paddles based on this value (0.5 is in the middle, 0 is at the top, and 1 is at the bottom).
Next up would be the ball. I would start by drawing a static ball in the middle of the screen. You will then need to set a velocity for the ball, and get it to move in that direction. After this, detect when the ball would hit a paddle(s), and adjust the velocity accordingly. Do the same for the 4 walls. If the ball hits the top or bottom walls, do the same as hitting the paddle. If the ball hits the left or right wall, then reset the ball.
Finally is the score. Decide how you want to display the score, and build the display for that. Make sure you can still see the ball, even if it overlaps with the score display. After this, you will need two counters that you update when the ball hits the left or right wall. If a player hits the top score, then you should congratulate them and restart the game by resetting both the ball and the scores.

5 Extensions

- Start the ball's initial movement when **either** player clicks on the joystick
- Make the celebration for the winner as elaborate as possible
- Add power-ups to the game (make the ball bigger, faster etc.)
- Speed the ball up as the game progresses