CECS 346 Project 2

Drag Race

Preparation:
You will need a LaunchPad, one to three push buttons (**not switches!** Ask instructor for buttons if needed. Number required depends on number of on LaunchPad buttons used), one to three 10kΩ resistors, eight color LEDs (2 red, 4 yellow, and 2 green), and eight resistors for the LEDs (between 330Ω to 1kΩ each).

Book Reading: Textbook Sections 2.7, 4.2, 6.5, 9.4, 9.5, 9.6

Starter Project: Lab5/Project 1

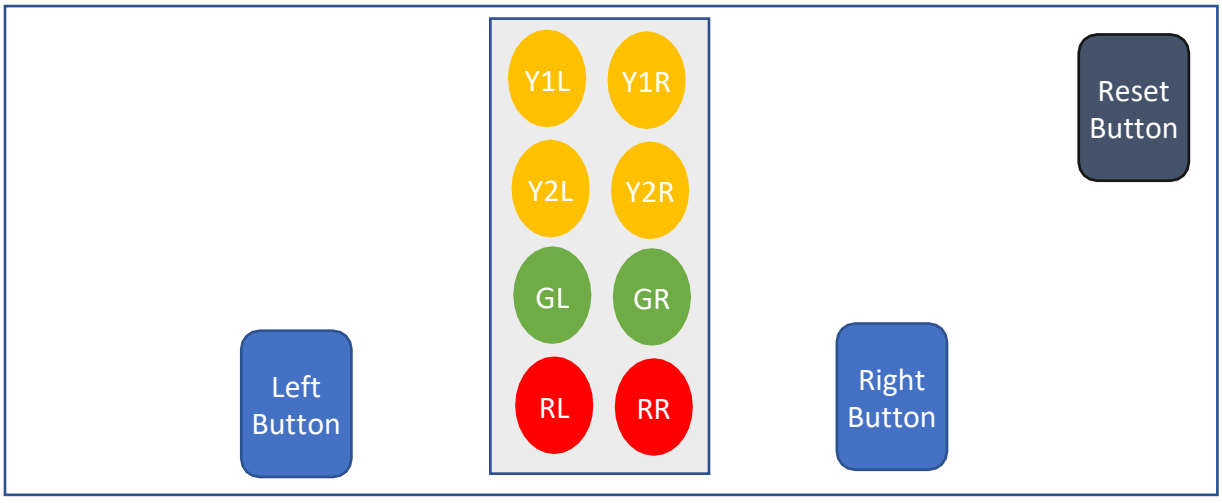
Purpose:
The purpose of this project is to combine the concepts of

- interfacing to switches and LEDs,
- using a Moore finite state machine (FSM), and
- using interrupts (GPIO and SysTick).

You will perform explicit measurements on the circuits to verify they are operational and to improve your understanding of how they work.

System Requirements:
Design a system based on a drag race. A 'Christmas Tree' is the light signal that drivers use to determine when to accelerate (ie start the race), which is simulated by our LEDs. A track has multiple sensors to determine if a car is in the proper position ("staged"), simulated by pressing our "lane sensor" buttons, one per lane (left lane and right lane). Our project will determine the winner based on who releases their button fastest after the green LEDs are lit. A reset button will be used to override the state machine current state and reset the board back to the initial state (similar to the reset button on the LaunchPad).

Consider a drag strip shown in Figure 1. There are two lanes (Left and Right), with a "Christmas tree" showing LEDs for both lanes.



There are three inputs to your LaunchPad:

- Reset – board immediately goes to **Initialize** state regardless of current state and SysTick timer value
- Left lane sensor – Represents if car in left lane is in proper position to start race
- Right lane sensor – Represents if car in right lane is in proper position to start race

Logic:

- The system starts in the **Initialize** state, during which all LEDs are lit. This state must (re)start the SysTick timer. After 1 second, the system unconditionally transitions into Wait for Staging state, where no LEDs are lit.
- While in the **Wait for Staging** state, it checks every 0.5 seconds if both left lane is staged(left lane button pressed) and right lane is staged (right lane button pressed). If both are pressed, the system transitions to the Countdown Y1 state. Otherwise the system stays in the Wait for Staging state.
- In the **Countdown Y1** state, only LEDs Y1L and Y1R are lit. After 0.5 seconds, if the left lane button or right lane button is released (or both), the system goes into the False Start Left/Right/Both state (those are 3 different states). Otherwise the system transitions to the Countdown Y2 state.
- In the **Countdown Y2** state, only LEDs Y2L and Y2R are lit. After 0.5 seconds, if the left lane button or right lane button is released (or both), the system goes into the False Start Left/Right/Both state (those are 3 different states). Otherwise the system transitions to the Go state.
- In the **Go** state, only LEDs GL and GR are lit. Every 0.05 seconds, the system checks if thelane buttons are pressed. If only left lane button is released, the system transitions to Win Left. If only right lane button is released, the system transitions to Win Right. If both lanebuttons are released, the system transitions to Win Both.
- In the **False Start Left** state, only LED RL is lit. In the **False Start Right** state, only LED RR islit. In the **False Start Both** state, only LEDs RR and RL are lit. From any of these 3 states, the system transitions to the Wait for Staging state after 1 second.
- In the **Win Left** state, only LED GL is lit. In the **Win Right** state, only LED GR is lit. In the **Win Both** states, only LEDs GR and GL are lit. From any of these 3 states, the system transitions to the Wait for Staging state after 1 second.
- While the reset button is pressed, immediately (without waiting for state SysTick timer to change) stop the SysTick timer and transition to the Initialize state. **You do not need to consider the reset button as an input to your state machine.** So, there are only 2 inputsto your state machine: left lane sensor and right lane sensor.

Implementation Requirements:

- A two-button interface using positive or negative logic (or combination of both) and either on board buttons or on breadboard buttons, for the left lane and right lane sensor buttons. **Use GPIO interrupts with priority 2 to detect both edges for both sensors.**
- An eight LED interface that implements either positive or negative logic (or combination of both) using on breadboard (not on LaunchPad) LEDs
- A one button interface **on a separate port than the left and right lane sensor buttons** to represent the reset button. **Use GPIO interrupts with priority 1 to detect level sensitive**

(when pressed) for reset button.
Implement a Moore finite state machine (FSM). There should be a 1-1 mapping between the FSM graph and data structure. For a Moore FSM, this means each state in the graph has a name, an output, a time to wait, and 4 next state links (2 inputs, so $2^2 = 4$ combinations of inputs). The data structure has exactly these components: a name, an output, a time to wait, and 4 next state pointers ($2^2 = 4$ combinations of inputs). There is no more and no less information in the data structure than the information in the state graph. **There should be onlya single output variable; use bit masking and bit shifting if you need to separate values.**

There can be no conditional branches in the program. This will make debugging the FSM engine trivial.

You must use SysTick interrupt with Priority 3 to implement state delays.

The state graph should define exactly what the system does in a clear and unambiguous fashion. In other words, do not embed functionality (e.g., flash 3 times) into the software that is not explicitly defined in the state graph.

Each state has the same format. This means every state has exact one name, one output field, one time to wait, and 4 next indices.

Use good names and labels (easy to understand and easy to change). Examples of bad state names are **S0** and **S1**.

Procedure:

1. Decide which port and pins you will use for the inputs and outputs. Avoid the pins with hardware already connected.
2. Define bit-specific addresses for input and output, one per direction (input/output) per port used.
    a. **Your Input variable must have the 2 switches mapped to the 2 least significant bits. Perform right shifts (>>) and any other operations needed to make the bits that way.** Imaginary example:
    ```
    Input = SENSOR_PORTA >> 2 + SENSOR_PORTF >> 1;  // read sensors and shift bits
    ```
    b. **You may also need to 'split' your OUTPUT variable to go to the different output ports.** Imaginary example:
    ```
    LIGHT_PORTB = FSM[S].Out & 0x3F; // set lights port b
    LIGHT_PORTF = (FSM[S].Out & 0x0A00) >> 8; // set lights port f
    ```
3. Design a Moore finite state machine that implements the system described. Draw a graphical picture of your FSM showing the various states, inputs, outputs, wait times and transitions, as well as a state table for the FSM.
4. Write and debug the C code that implements your system.
    a. Your GPIO interrupt handlers should update global variable(s) based on the GPIO.
    b. Your SysTick handler should:

i. Determine the new current state based on the global variables set by the GPIO interrupt handlers and the current state,

ii. Set the system outputs (LEDs) based on the new current state,

iii. Configure the SysTick delay to be the delay needed for the new current state.

5. Implement your system on the real board. Use the same ports you used during simulation. Do not place or remove wires on the protoboard while the power is on. Build and test the switch circuits.

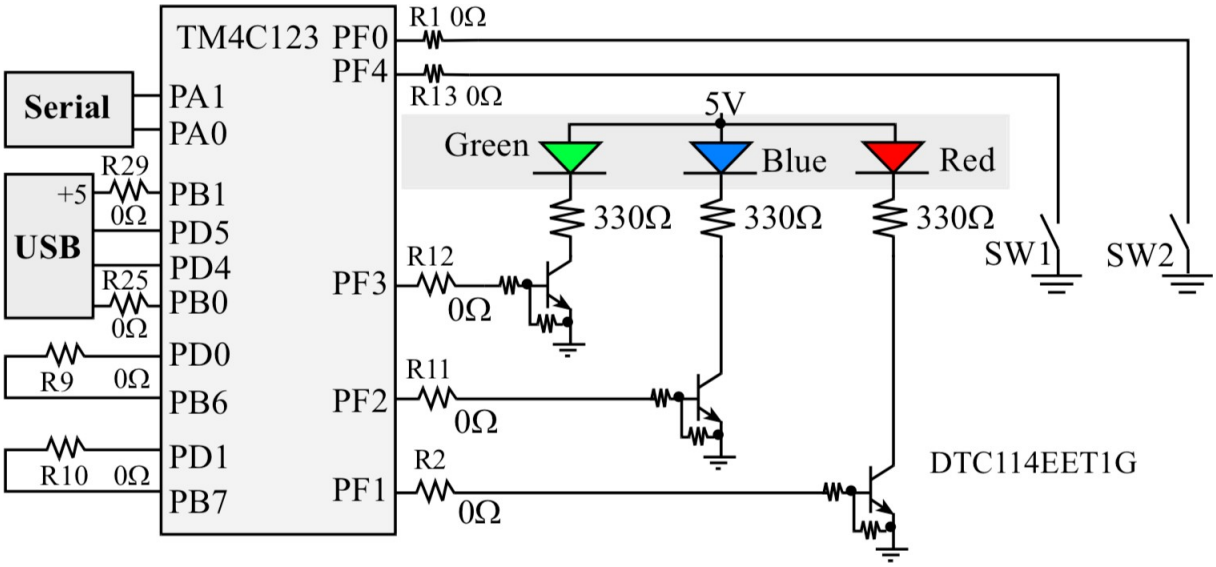


Figure 2 Switch and LED interfaces on the Stellaris® LaunchPad Evaluation Board.

Deliverable:

1) Demonstrate your lab. **Label the physical board using e.g. post its, cut up index cards, etc so that each LED and switch has an obvious name/label before demonstrating.**
   a. In simulator
   b. On board
2) Submit a project report (eg Word Document) to the Beachboard Dropbox containing:
   a. Class name, lab number and name, your name and your partner's name (if applicable)
   b. Table showing list of GPIO ports + pins used (similar in format to "Recommended GPIO Pins" in Project 1 description)
   c. Description of which bits in the input map to which GPIO ports + pins.
   d. Description of which bits in the output map to which GPIO ports + pins.
   e. State table for your Moore FSM
   f. State diagram for your Moore FSM
   g. Photographs of your hardware system: One per state.
   h. Labeled photograph of your hardware system with all LEDs and buttons labeled. (Can be a same picture as above.)
   i. Diagram of your hardware system (see Figure 2 for an example)
   j. Software source code: The .c file
   k. Include short description (for example, a couple sentences) describing the extra credit changes you made.

Code Setup:
Recommended starting point is Lab5 or Project 1.

Additional References (Optional):

- https://en.wikipedia.org/wiki/Christmas_tree_(drag_racing)
- https://startdragracing.com/drag-racing-christmas-tree