Final Project Presentation

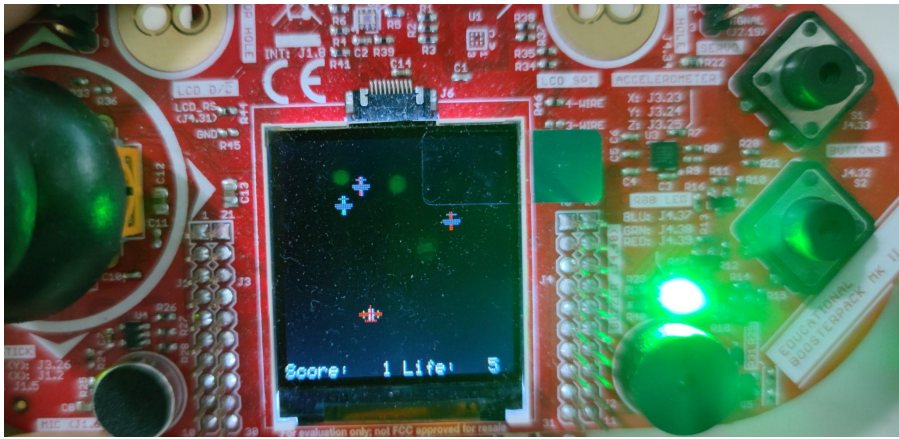
Team Iota

Space Lasers Game

- Ideas and features of the game
- Design and implementation methods
- Team responsibilities
- Demo and analysis
- Insights and lessons learnt.

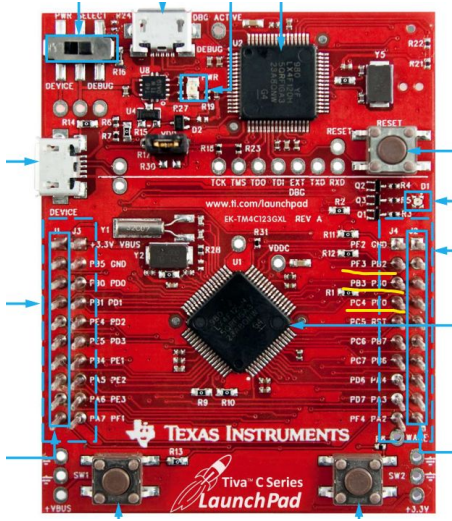
Ideas and features

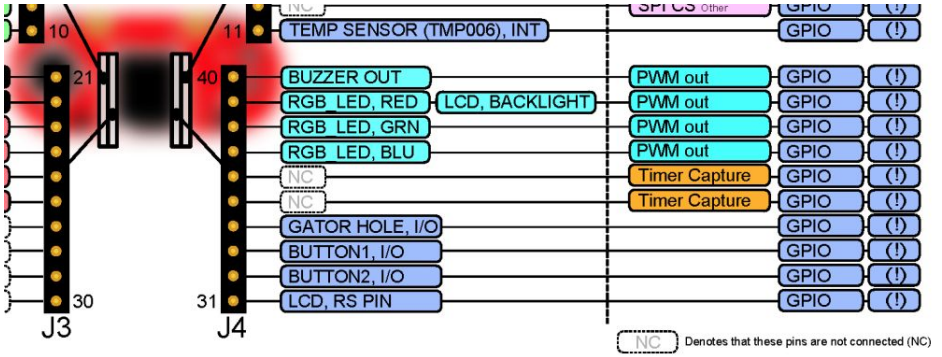
- Changes in game logics
  - The joystick controls a spaceship instead of a simple crosshair
  - Enemy aircrafts spawn from the top of the screen
  - Our spaceship can shoot lasers
  - Life reduction after crashing into an enemy aircraft, or enemy aircraft reaching the bottom of the screen
  - Game start & Game over screen
- Use more BoosterPack hardware features
  - RGB LED to indicate life count
  - Buzzer to play sound effects (planned)



Design and implementation methods: RGB LED

- A more engaging way of showing the life count
  - (❌) Life: 5
  - (✅) 🟢 🟡 🔴
  - The RGB LED is big and bright
  - It blinks when life is low
- Pinouts
  - R,G,B: PF3, PB3, PC4
- Initialization of the RGB LED
  - Activate clocks for Port B,C,F
  - Unlock GPIO for Port F
  - Disable analog features
  - Configure pins as GPIO
  - Configure pins as output
  - Disable alt features
  - Enable digital I/O





Design and implementation methods: RGB LED

- The color is configured by programming the GPIO registers of R,G,B channels
  - Also capable of PWM, but complex brightness and color changes are not necessary in this game
- The RGB LED is updated through a periodic background thread
  - Currently, OS_AddPeriodicThread() adds a periodic background task using either Timer1A or Timer4A
  - Timer1A is already used for the Producer thread
  - Set Timer4A to 4Hz

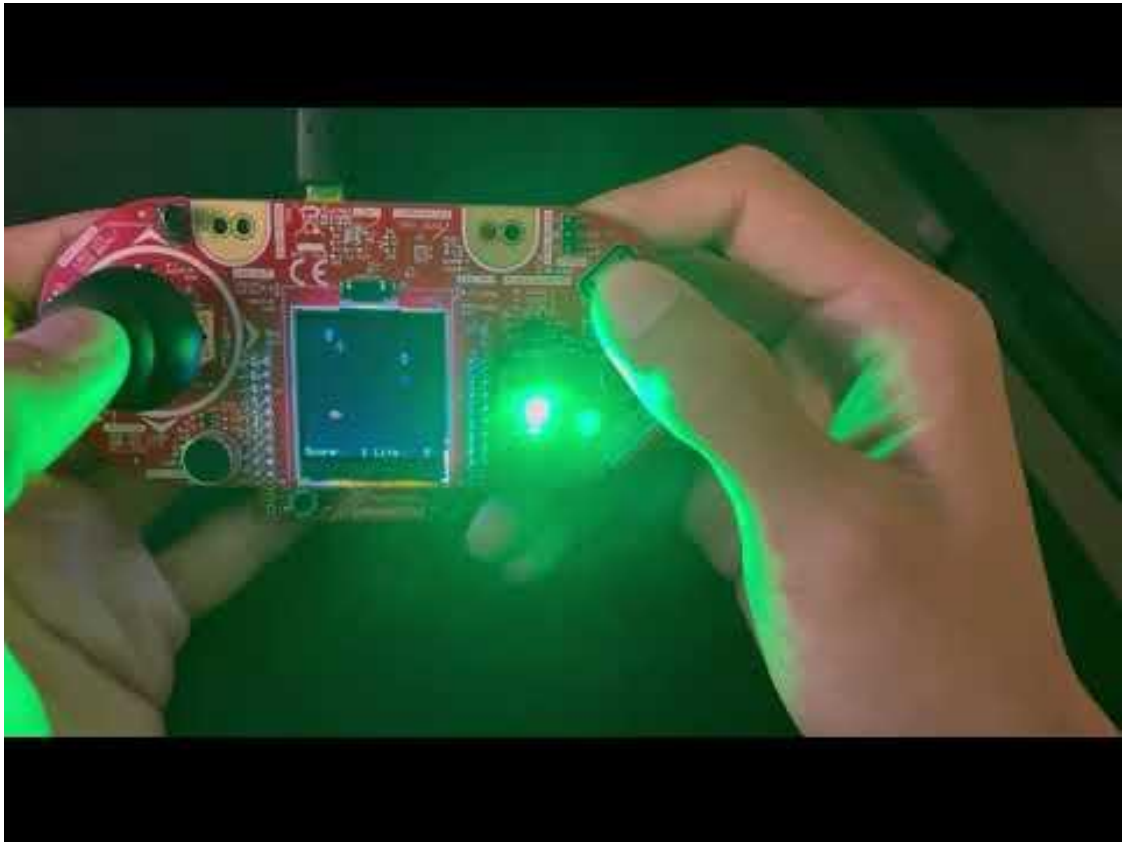
Game Algorithm

- Deadlock prevention
  - BlockArray struct holds the information of occupied blocks of current cubes (similar to the tcb linked list)
  - Cube struct holds the information of each cube
- Laser thread
  - A foreground thread that gets created after pushing button1
  - Show a vertical laser starts from the location of the craft for 15 ms (OS_Sleep)
- Gameflow
  - A foreground thread that shows the beginning image for 1000ms and then kills itself (Game begin thread)
  - Check if life is zero => kill all foreground threads using NumSamples = RUNLENGTH and display game over (CubeRefresh thread)
- Some other changes:
  - All targets appear randomly from the top of the screen, randomized movement but cannot go up
  - Provide 12 x 12 cells for the enemy crafts to move randomly
  - Different logic for killing the cube thread (through laser and crush) and decrementing the life (crushing into enemy crafts or enemy crafts escapes from the bottom)
  - Different images representing the cubes and the crosshair

```
typedef struct {
uint32_t position[2];
uint32_t available;
} block;
block BlockArray[5];

typedef struct {
        uint32_t position[2];
        int color;
        int lifetime;
        int index;
} Cube;
```

Demo https://www.youtube.com/watch?v=mg24uE-WCd4



Insights and Lessons Learned

- Round Robin scheduling worked well because we had a max of 8 threads
  - 2 milliseconds for each task
  - Each round of execution is 16 milliseconds at most
  - Some threads are cooperative (calls OS_Suspend()) because they do not need the full 2ms time slice
  - No noticeable delay between first and last task and the game runs smoothly.
- If more threads were added, we need other methods for the scheduler
  - Priority scheduling with aging to give the other tasks a chance to run.
  - Use OS_sleep to let lower priority tasks run.
  - OR we will need to change the RTOS to set up frequency and deadlines for each thread.
- More Functionalities that could be added in the future
  - Buzzer: how it would play the music using a pwm signal
  - Tried to use the provided example in the book, but couldn’t get it to work

Foreground Thread Comparison Table

Thread Name	Functionality	Max Number of Threads	Cooperative (calls OS_Suspend())
Consumer	Reads joystick input from FIFO	1	Yes
Display	Updates the LCD with score and life count	1	Yes
CubeRefresh	Increases number of cubes	1	Yes
CubeThread	Cube movement and hit logics for each cube	4	No
Laser	Displays the laser for a short period of time	1	Oneshot

Thank you

Part 2 Team Responsibilities:

Chenyang Zhong: LED driver and logics; game logic debug and optimization;
Yimin Gao & Zhenghong Chen: Majority of game logic changes;
Bassam Mohmaud: Tried to bring up the buzzer; performance analysis