Embedathon 2022

Problem Statement

The game

Snake is a video game genre where the player maneuvers a growing line that becomes a primary obstacle to itself. The concept originated in the 1976 two-player arcade game Blockade from Gremlin Industries, and the ease of implementation has led to hundreds of versions (some of which have the word snake or worm in the title) for many platforms. 1982's Tron arcade game, based on the film, includes snake gameplay for the single-player Light Cycles segment. After a variant was preloaded on Nokia mobile phones in 1998, there was a resurgence of interest in snake games as it found a larger audience.

We all had a fantastic time playing the classic snake game, and now it's time to recreate it using your LED matrix. Your snake game implementation should have a snake whose size increases after each "orb" it eats (Yeah, I ran out of ideas to name it). The increase in the size of the snake should correlate with the number of orbs it ate. The game should also have a death condition. Usually, the death condition is triggered when the snake eats itself or when it hits a wall (implementation of walls is optional). The game should end when the death condition is triggered.

These are just the basic guidelines, and you are free to implement additional features and will be rewarded accordingly for implementing them.

Evaluation metrics

All teams shall be evaluated based on the following metrics; these metrics take into account both practical and theoretical understanding of the topics that have been covered so far.

- ### Readability of code

  Write code that has comments that explain your approach. Remember if we can't understand your code then we can't evaluate it, and if we can't evaluate it then you don't get points for it.

- ### Modularity of code

  A good code always makes use of functions wherever possible. If some operation is being performed multiple times then it probably means it's time to use functions.

- ## Use of correct registers

  Embedded C is all about configuring the right registers, the right way, and at the right time. This also includes if you use the right constants to configure the registers, as this shows you have genuinely understood the code.

- ## Functioning game

  Well if we can't play the game what's the point? Your game should run flawlessly and with as less lag as possible (embedded C will help you with this). Your logic should also deal with cases that are very rare but could break your game.

- ## Easy of use

  Your game should be very easy to use. It should be such that if you give your PCB to a layman then they should be able to easily understand how to play the game. We do not want someone to go through a steep learning curve just to enjoy the game.

- ## Innovation!

  Games are unique, and they express the creator's visions. So we encourage you, nay! We demand you to innovate. Surprise us with your wildest takes on the problem statement (Please don't go and make it like a souls game). Of course, these all should be within the rules of the problem statement.

- ## Arduino C?

  If you are struggling with embedded C then you can code it with Arduino C but, you shall be given a penalty for doing so. So will you whimper like a noob and use Arduino C, or will you have a clutch moment and ace this problem statement like a pro using AVR C?

Submission

You have to submit a video showing your solution. This has to be a seamless (unedited) video, and it should include -

1) Introduction of all team members.
2) Well-lit 360-degree view of the final soldered PCB.
3) Playing the coded game

The video should be named as - **Team_<team number>_Final_Submission**
Your code should be named as - **Team_<team number>_Final_Submission.ino**
Pictures of both the front and back of the final PCB -
Team_<team number>_Final_PCB_<front/back>

Both of these should be put in a zip file named -
Team_<team number>_Final_Submission.zip
(Use of other zipping formats is also allowed).

You have to submit your solution to the below google form -
https://forms.gle/R6LYgAHk9dEFj4Dk7

FAQs

Will I have to work for the complete 24 hours?

It's your 24 hours, so you are free to use it however you want. Prioritize and divide the work among your teammates. It is all about the efficient use of time. This is a marathon, not a 100-meter dash.

What if I have doubts during the final task?

Ask it on the channels that have been assigned to your team. You can ask it on the all-teams channel also. After asking your doubt one of the organizers will contact you at the earliest to help you.

Will previous tasks have a weightage for the final results?

There will be some weightage given to the previous tasks however, the final problem statement will have the highest weightage.