(A) Requirements

1.1. Functionality

Overall, this system is a space shooter game controlled by a proximity sensor. The spaceship alway stay on the left hand side of the screen, and the background will advance horizontally towards the right hand side, and the enemies will come out from the right. There will be different types of enemies, some weak and some strong. And there will be bosses in each chapter as well. The player's spaceship may pick up upgrades along the way and have a higher firepower. The player will also have access to a limited amount of bombs in each chapter.

1.2. Performance

Performance is critical in our system. There are two main measurements that we will take to ensure that we have a satisfactory gaming experience for the players: input delay and frame rate.

First off, we will be using a proximity sensor for input. Since we have not yet received the part, we do not know how long the sampling typically takes. In short, we want the delay between when the user moves his/her hand and when the spaceship actually moves on the screen to be as short as possible. We will use the logic analyzer to check how long the sampling ISR takes to ensure that this delay is within a range.

Secondly, we want the game to be displayed at least at an acceptable frame rate, 20 FPS. This can be easily measured. We can simply use SysTick to count down 1 second, and check how many times the loop was executed (the screen gets updated each time the loop was executed).

1.3. Usability:

The user will interact with the game via a proximity sensor and a switch.

For the proximity sensor, the user would move his/her hand up and down directly above the proximity sensor (facing upward) in order to control the spaceship on the screen to move up and down. The switch will be used for the classic full-screen bomb. Normal bullets will be shot automatically, so there is not switch for that.

(B) Hardware Design

See our .sch file

(C) Software Design

See our .c and .h files

(D) Measurement data

Debouncing Tests: We clicked our button 1 and button 2 50 times, respectively. Our purpose is to check how many times the clicks get registered, not registered, or registered as a double click

- Button 1:
  - Success: 43
  - Not registered: 2
  - Double click: 5
- Button 2:
  - Success: 45
  - Not registered: 0
  - Double click: 5