**SECTION I - Requirements:**

**1. Overview**

**1.1. Objectives:**

The object of this design is to design and build an interactive game with components soldered onto a PCB. Educationally, this will allow us to learn how to interface a sensor with the embedded system in real-time, designing and building a PCB, and developing a game. Our goal is to create a fun interactive game that can be played with hand movements.

**1.2. Roles and Responsibilities:**

We’re designing and building a space-shooter that is controlled using a proximity sensor, therefore you can control the movements of the spaceship using hand motions. The engineers for this project are Tianyun Duan and Tahir Haideri, the client is the TA for EE45L and Dr. Valvano. The two engineers will work together to design the PCB using PCBartist. The two engineers will work on the software and hardware together, assigning specific tasks as needed.

**2. Function Description**

**2.1. 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.

**2.2. 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 an ultrasonic sensor for input. We sample the ultrasonic sensor at rate 10 times greater than we update the player ship according to the Valvano postulate to ensure we accurately represent player hand movements. We test the sampling rate using a logic analyzer. The sampling rate is limited by the time required to receive the echo ~10 microseconds.

Secondly, we want the game to be displayed at least at an acceptable frame rate, 20 FPS. To ensure there is no flicker, we shall be using timers and flags to only update objects when needed. The lowest possible resolution for updating an object will be 50ms i.e. equivalent to the frame rate.

**2.5. Usability:**

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

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 switches will be used to navigate in-game options

**3. Deliverables**

**3.1. Reports**

Testing procedure and testing data

YouTube video

**3.2. Outcomes**

A) Objectives

2-page requirements document

B) Hardware Design

Detailed circuit diagram of the system (from Lab 7)

C) Software Design (no software printout in the report)

Briefly explain how your software works (1/2 page maximum)

D) Measurement Data

Include data as appropriate for your system. Explain how the data was collected.

E) Analysis and Discussion (none). The YouTube video is required

**SECTION II – Software Design:**

The software consists of drivers to communicate with peripherals. In addition to peripheral drivers the software includes game loci and the main function which is responsible for running the game. The Timer.c file initializes 5 32-bit timers for use by other threads. The Proximity.c file contains initialization code for the ultrasonic sensor as well as functions to sample the ultrasonic sensor periodically. The ultrasonic sensor uses input capture to measure the distance. The DAC.c and Sound.c files contain the implementation for playing in-game music. Sound.c uses two timers to play music. The primary timer controls the duration of the notes while the second timer is used to output notes using the desired frequency. Switch.c contains the initializations for the switches as well as the implementation for debouncing. Switches are debounced using a one-shot timer that disable interrupts on the pressed switch for ~10 microseconds. Game.c implements the game logic and contains both public and private functions to update game state. We have data structures to store information on ships and bullets. Enemy ships are stored in a 2D array to simulate the physical grid on screen. Public functions in Game.c are called by the main thread to update position of objects and check for collisions. The ST7735.c and Graphics.c prints objects to the screen. The main thread is responsible for updating objects and screen and calling the appropriate graphics function. The program uses a hardware timer to determine how often to update individual classes of objects. This is done to reduce on screen flicker since not all objects have to be updated as frequently. This also makes it easier to control the speed of certain classes of objects.

**SECTION III – Hardware Design:**



