# EE/CPE 3280 Assignment 2 – STAR Blasters

This assignment will introduce the use of GPIO through pushbuttons and LEDs, and further your embedded C programming skills.

## Overview

***Basic game (for a maximum grade of 80/100):*** Star Blasters is a simple cockpit-view game in which you pilot a spaceship and blast stars. The game shows a view into space out the cockpit window of a space ship. The space background is punctuated by approximately 100 stars. A set of crosshairs highlights the aiming point for your ship’s blaster; the crosshairs may be moved in any direction using the ABCD buttons on the Joystick Shield. Pressing the joystick inward, toward the shield, will fire the blaster.



Vide

Video demonstration at <http://youtu.be/7ef2Wpg-Frw>.

***Extensions (for additional credit, as indicated. Max grade: 100/100):***

1. (10 points) Twinkling stars. Make the stars twinkle (briefly go black at random times). The more realistic, the more points you get.
2. (10 points) Three-level blaster charge-up. The user must wait for the blaster to charge before using it. If the user waits long enough, a hyper-blaster mode will be engaged, which gives a larger explosion. Using the blaster in either regular or hyper-blaster mode will deplete the charge to zero.
3. (10 points) Scrolling starfield. Whenever the crosshairs are moved fully to one edge of the screen, the starfield will scroll to indicate further movement in that direction.

## Specifications

### Basic Game

1. Stars. Upon startup and throughout the game, 100-200 stars shall be displayed on a black background. Stars shall be indicated by single pixels colored white and arrayed in a pseudo-random fashion.
2. Crosshairs: A set of crosshairs shall be displayed on the screen. The crosshairs may be any suitable shape or color but shall have a size of between 20 and 40 pixels on each side.
3. Crosshair movement. The crosshairs shall move in the direction indicated by the ABCD buttons. Upon reaching an edge of the screen, the crosshairs shall cease to move in that direction. Movement from one side of the screen to the opposite side shall take between 1 and 3 seconds.
4. Blaster. When the joystick is pressed inward, into the board, the game shall display an “explosion” sequence centered at the location of the crosshairs. The explosion sequence shall be 40-60 pixels in diameter and shall last no more than 1 second. This specification may be modified if the Blaster Charge-up extension is implemented.

### Extensions

1. Twinkling stars. Stars shall temporarily go dark (black) and back to white according to a random sequence. The dark time shall be between 0.1 and 0.5 seconds in length. The number of stars that are black at any one time shall be 1-3% of the total number of stars.
2. Three-level blaster charge-up. The blaster shall operate in three stages:
   1. Empty. The LED shall be illuminated red to indicate the Empty state. The program shall ignore the center button of the joystick when in the empty state. The state shall change to “Charged” after 3-5 seconds in the Empty state.
   2. Charged. The LED shall be illuminated green to indicate the Charged state. When the center button is pushed during the Charged state, a normal size explosion shall be displayed on the screen. The state shall change to “Hypercharged” after 3-5 seconds if the blaster is not used in the Charged state. If the blaster is used in the charged state, the state changes to “Empty”.
   3. Hypercharged. The LED shall be illuminated blue to indicate the Hypercharged state. When the center button is pushed during the Hypercharged state, an extra-large size explosion shall be displayed on the screen. If the blaster is used in the charged state, the state changes to “Empty”.
3. Scrolling starfield. Whenever the crosshairs are located along any edge of the screen, and the ABCD buttons indicate movement in the direction of that edge, the scrolling mode shall be engaged. When scrolling mode is engaged, all stars shall appear to move away from the edge on which the crosshairs are located. Any stars that disappear off of the opposite edge shall be made to appear on the edge that the crosshairs are located on. After scrolling one entire screen-width, the starfield shall be in the same configuration as before the scrolling began.

## Important tips

1. You’ll need to add pins for the ABCD and Joystick center buttons, and three LEDs.
2. Remember to set the drive mode to “pull-up” mode for all pins connected to buttons.
3. Programming strategy. The simplest way to get this done is to write one big loop that continuously:
   1. Checks the ABCD buttons and moves the crosshairs (and stars if scrolling mode is implemented)
   2. Checks the fire button and fires the blaster
   3. Redraws the stars (in case the crosshairs or blaster took them out) and makes them twinkle (optional)
4. Timing. There are many ways to accomplish timing with microcontrollers, but for now all we have are code delays and such. To implement components with different timing needs try this: Put a counter in your main loop (if you declare it as a uint32, it will roll over to zero after hitting 4 billion). If you have a section of code that needs to be run only once every 1000 loops, you can put in in an if clause and use the modulo function, such as “if (counter % 1000 ==0)”.
5. Stars. Since the twinkling and scrolling extensions require that the MCU knows the location of each star, you’ll need to keep track of this. An array of x,y coordinates, one pair for each star, will do nicely.
6. Random numbers. Include <stdlib.h> and use the rand() function. It returns a random integer. You can use the modulo function to limit its range. For example, for a random number between 0 and 120, use “rand() % 121”.

# To Turn In

Turn in the following:

1. Submit a single MS-Word document containing the following through Canvas
   1. Top-level schematic (select all of your drawing, copy and paste it into the Word doc)
   2. main.c (Select all text with ctrl-a, copy and paste into the Word doc)
   3. Copy in any other files that your created or modified
2. Demonstrate your program to the instructor during class time on the due date. (Online students may send a video demo to the instructor)