Space Invaders

final project

CECS 347

Kassandra Flores, Adan Hernandez

12-15-15

For this project, we built an 80's game like space invaders using the TM4C123 LaunchPad microcontroller with C code.

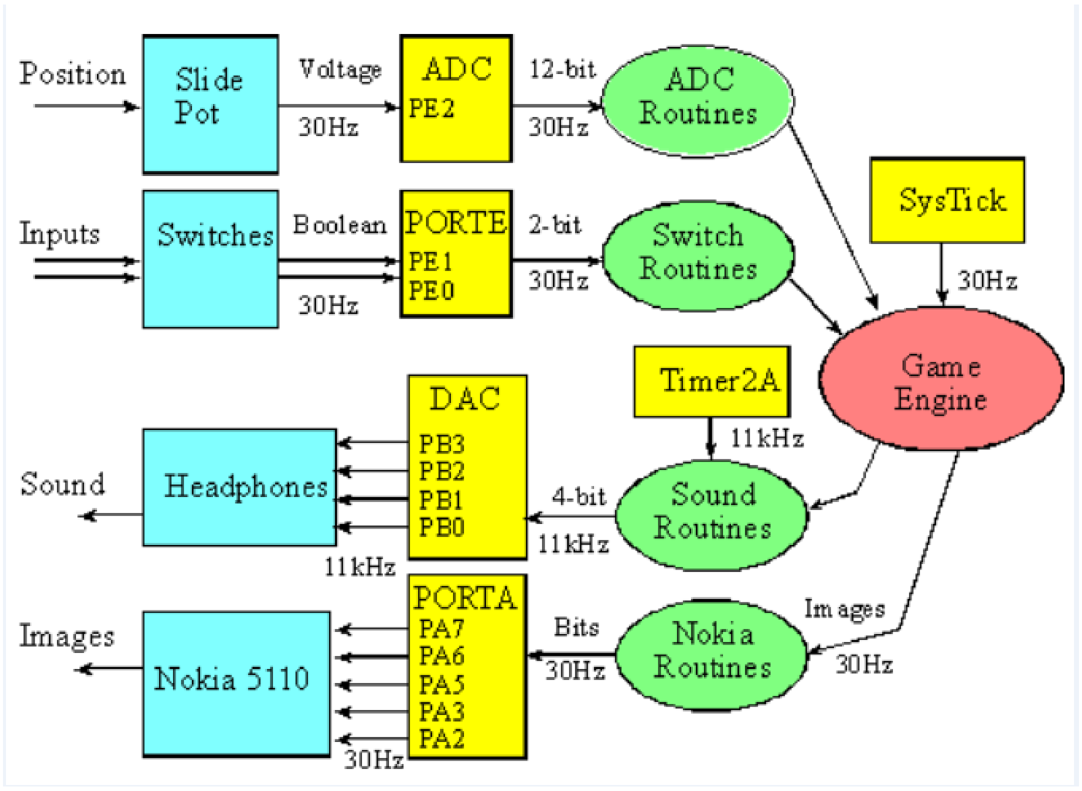
Introduction: We used the TM4C123 Launchpad to build the 80's like space invaders game using C code. We had the starter code for this project which was provided in the Keil folders. We followed the flow methodology that was suggested in the project description by outputting to the four bit DAC for sound in the Timer2A Interrupt Service Routine(ISR). The Timer2A runs at 11.025KHz. The turning on of the LED's, when you killed an enemy a green LED would turn on and a bunker being destroyed the red LED, was determined by the Timer2A ISR. The rest of the game was ran through the Systick ISR which had a frequency of 30Hz through 120Hz. The 30Hz to 120Hz depended on the number of enemies you killed, the more enemies that we're destroyed, the faster the frequency it would be. This would cause the enemies to move faster because of the Systick ISR. This took care of the difficulty in our game. The starter code that was in the folder had the support routines for the LCD graphics.

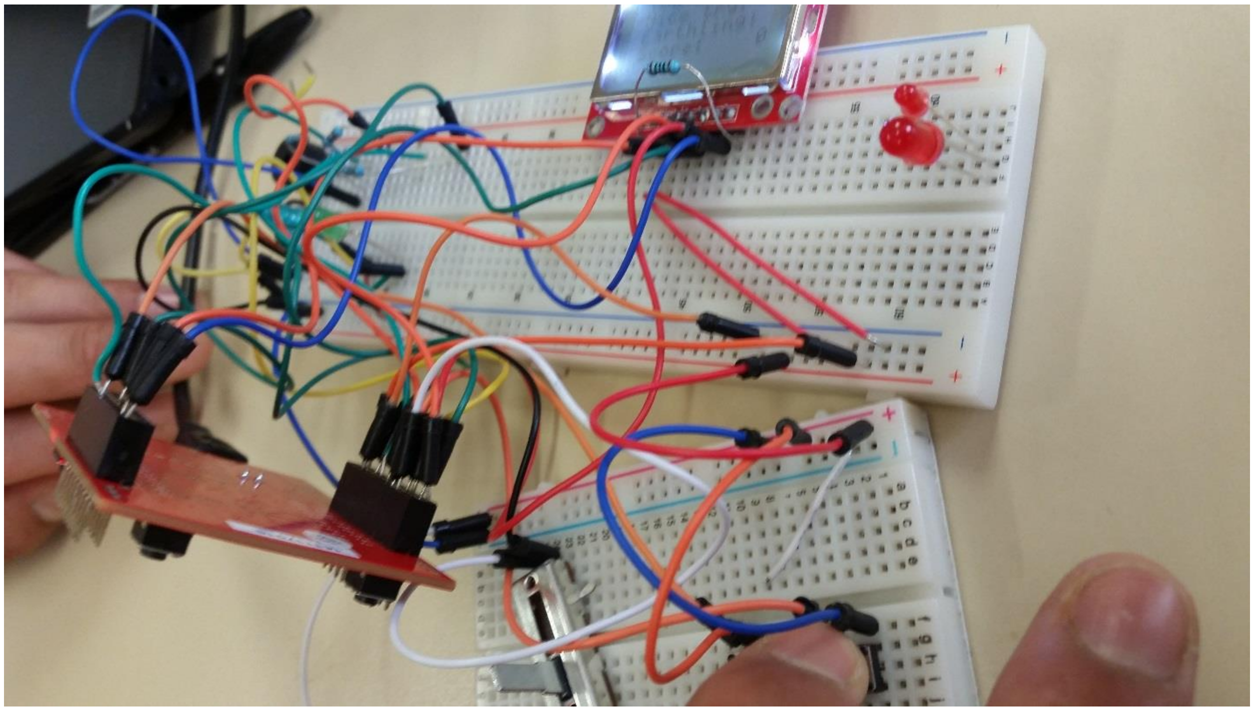
Operation: Our game ran on the TM4C123 LaunchPad microcontroller. We used two push buttons to send out the missiles to the enemies. The Nokia5110 LCD was used to display the contents of the game while it was being played. For the sound, we used a small two pin speaker, and also used two LED's to indicate when you have killed an enemy and also when a bunker has been destroyed. In order for the speaker to work, we used a DAC, which was a four bit binary weighted DAC. Finally, in order to move the player left and right, we used the slide potentiometer from our previous lab.

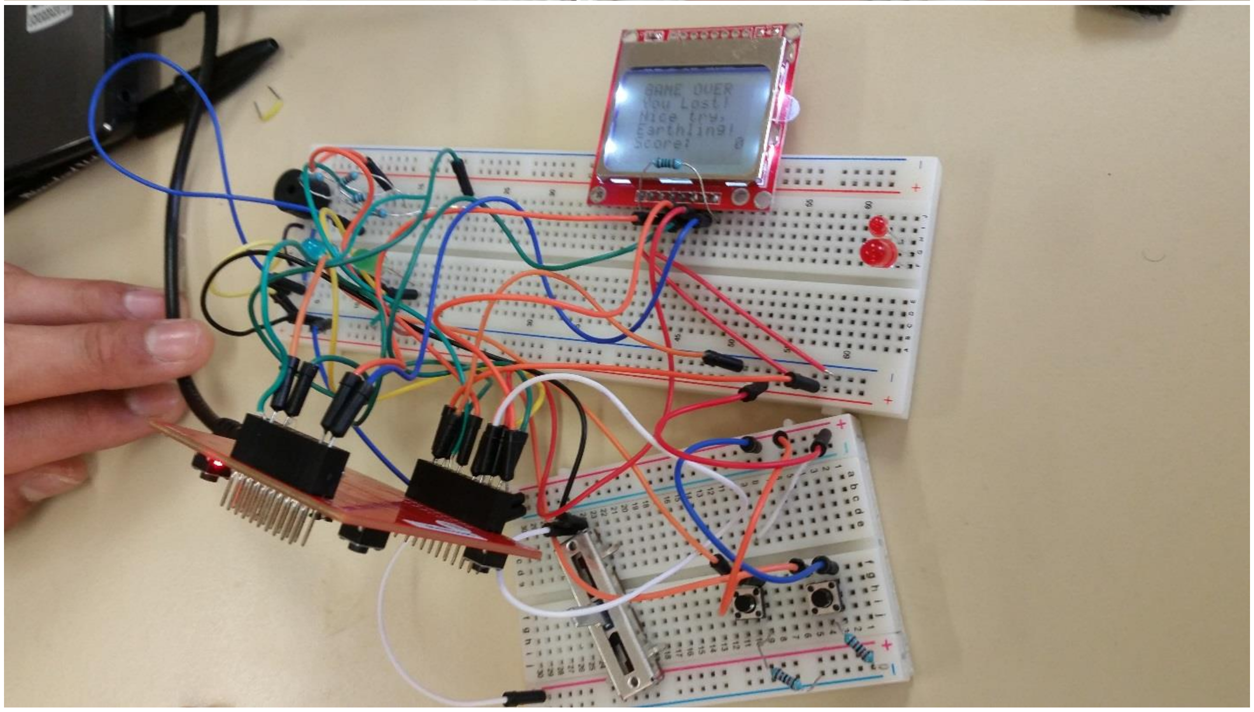
Theory: We used structures, which in C are another user defined data type available in C that allows us to combine data items of different kinds. For us, this helped us keep track of the missile attributes all in one. This helped us create the image of the missile being fired, in the x and y coordinate. We used timer0.c for our program which used Systick interrupts to implement a single 440Hz sine wave. We also used modular design which allowed us to put all the components of the class into one system. The ADC was used for the pot which allowed us to move the player right and left. The DAC allowed us to hear the sound of the game. The switches we're our edge triggered interrupts which we're positive edge triggered. When we pushed the button, a reaction or shot came from the ship, which meant that it was being triggered at the positive edge. We used port E as the switches which we're outputs in the GPIO. The LCD was the Nokia5110 that was being interfaced to show the game being played.

Hardware Design:

Block Diagram:







video-1450247377.mp4.mp4

video-1450247366.mp4.mp4

Software Design:

GameEngine.c : This file drew out the environment for the player. This includes the number enemies, number of missiles. The module took care of the layout of the program and where everything went specifically on the LCD screen.

ADC.c : This provides the functions that initialize ADC0 SS3 to be triggered by software and trigger a conversion, wait for it to finish and then return a result.

Sound.c : This module was provided to us and runs on any computer. Sound assets based off the original space invaders and basic functions to play on. These we're provided by Jonathan Valvano.

DAC.c : The was an implementation of our 4 bit Digital to Analog converter. Port B3-0 was used. This would play the struct that had the sound of the space invaders which outputted from the hardware, which was the speaker.

Texas.c : Periodic timer interrupts data collection. The PLL must be on for the ADC to work. ADC1 PD3 Ain4 continues sampling.

Nokia5110.c : This moduel uses SSIO to send an 8bit code to the Nokia 48x48 pixel LCD to display text, images, or other information.

SwitchLed.c : This module used Port E bits 1-0 which are inputs. The fire buttons are connected to PE0 and the special fire weapons fire from PE1. Port E bits 5-4 are outputs. The LED's are on PB4 and PB5. The clock for Port E is already activated in ADC_init which is called before functions switchLed_init() in Main(). The function switch_fire() receives input from fire button and outputs when either a 0 or 1, depending whether the button was pressed, when being positive logic. We also have a success_Ledon() function which turns on the LED which is connected to PB4 when an enemy is hit or game is won. Failure_Ledon() function turns on the PB5 LED when a player is hit, the game is lost or a bunker is hit.

Spaceinvaders.c : This file which drove the entire game by having one super loop in the main which called the gameengine.c functions that started the game. When it noticed that the game had ended, it paused and restarted the game again. The Systick_handler() function ran at frequency of the Systick interrupts and toggled PF1. The handler would set up the game environment for the player, if either the game was currently being played or if the game had ended and had to set up the environment for the player to begin playing once more.

Timer0.c : This file uses Systick interrupts to implement a single 440Hz sine wave. This would increase the frequency of the DAC, the sound, which made it sound louder, or increase.

Conclusion: The successes we're having the limited amount of hardware which made it easier to troubleshoot. We had also had previous labs which helped build the system. Failures we're being able create a game engine with the only knowledge of structures in C code. Luckily we had many resources to finally create the final project which we're provided by Dr. Min He and Jonathan Valvano as well as other resources.