CS402 Project Phase II

Group 10: Michael Nolan

Idea

Our Arduino gadget is very simple, it is a rebuild of the 70s classic Space Invaders, created by Tomoshiro Nishikado [1]. The rebuild will run on the Atmega328p MCU and be displayed by a simple graphics LCD module. Input will be taken from as little number of buttons as possible both to reduce price (in keeping with the budget) and to keep the user interface as simple as possible.

1 Outline

Aliens are attacking. You, the user, will defend Earth using her last unmanned defense ship, Hope. The aliens are attacking in a formation of 2 rows and are gradually descending onto Earth. If they reach us, we all die. You mission is to destroy them with the laser canon we have installed onto Hope. We have provided you with a very simple 3 button controller as well as simple tactical interfacing display, otherwise known as an LCD. Combining these should yield you enough power to destroy all of the aliens - hopefully...

2 How to play

The game operates using a simplistic 3 buttons interface. The buttons are aligned in a logical order, the left button moves the player's ship left whilst the right button moves the player's ship right. The middle button can be used to shoot at the descending aliens. The game is won when all alien ships are destroyed and although the experienced user may expect to get another, possibly harder, level after he has destroyed all enemies, we decided to taunt the user with the 'game over' screen. The game is lost if either the player's ship is destroyed or any of the aliens make it to the planet (which is simply the bottom of the screen). The player's ship may be destroyed by the aliens firing down upon it. The player is unique in that they receive 3 lives (while aliens may only have 1). After these 3 lives have been used the player's ship will no longer re-spawn and the game will well and truly be over at which point the 'game over' screen appears.

The 'game over' screen is simply a message to inform the user that they may play again (and hence it resets all game variables to their initial states) should they wish by pressing any of the 3 buttons. This allows the player to play indefinitely.

3 Design II

We follow from the previous document (PhaseI) where we said that the design would be better explained here. As it is now known that we chose Space Invaders to replicate, we proceeded by firstly establishing the basic rules that make Space Invaders what it is. We found that in other

replicas of Space Invaders, the drop in y-coordinate of every alien when they reach either side is always present, as well as is usually followed by a speed increase. We also noted that there tends to be blocks at the bottom to represent some sort of shield for the planet, but given the limited constraints of the resolution of our screen, we thought that adding these blocks would be messy - since there are only 64 pixels vertically and between the aliens and the player's ship we are already using 24 pixels.

Since this was to be a simplified replica of the original 70s game, it was decided that simple rectangular collisions detection scheme should suffice for detecting hits between bullets and ships/enemy aliens.

4 How to build one

There will not be too much difficulty in reproducing this. The main issues lay with getting the LCD screen working and being able to check for input - these are the things where things will tend to go wrong as these parts of the project lay out-with the MCU and are thus 'alien' as far as the Arduino is concerned.

When we developed this, we broke it into several parts where firstly we ensured we had the ability to use the display, secondly we made a replica on a computer of the Space Invaders game, thirdly this was ported over to the Arduino - of course the appropriate display and input functions had to be changed from the computer screen and keyboard to the lcd screen and 3-buttons.

We ordered a small LCD display module off of ebay. The connections were not appropriate for breadboard so a miner amount of soldering was done. We thought of constructing a PCB in order to make things more presentable but did not pursue this as then the components would be tapered to one board whereas this way we had some leeway to experiment and we can, using the breadboard approach, move them between this project and others. The specification is much more detailed in the comments given in the source code - the actual implementation is also based on the protocol outlined by [2].

5 Construction

These steps give a simple overview of how the project was constructed. Note that some of the design (the circuit diagram for instance) was done on paper and a whiteboard which has not be included.

5.1 Arduino and KS0108 LCD module arrive (only LCD module is shown in this picture though)

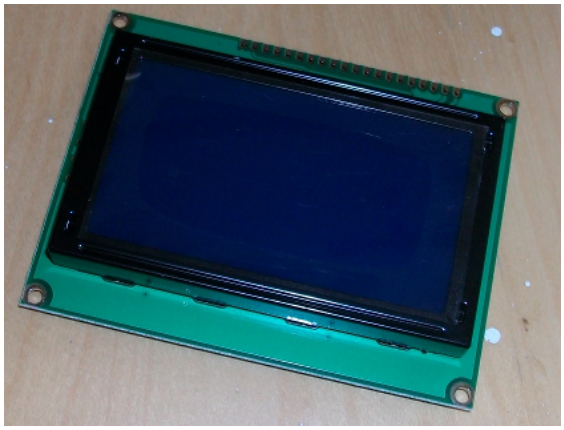

Figure 1: Step 1

5.2 Arduino arrives and is tested. 20x1 Header (snipped to size) is soldered to module and it is inserted into the breadboard

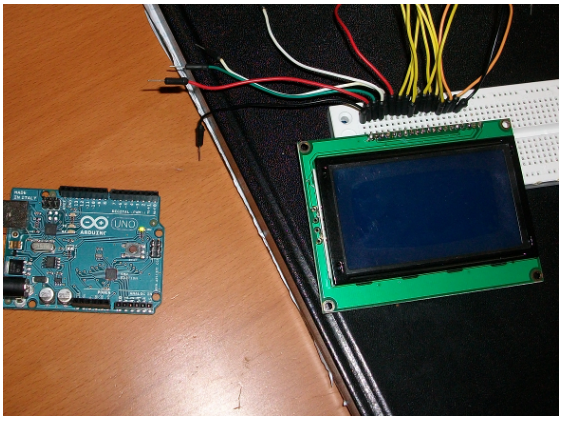

Figure 2: Step 2

5.3 Connections between Arduino and LCD are made, as well as shift register connected to LEDs for simple debugging if nessesery

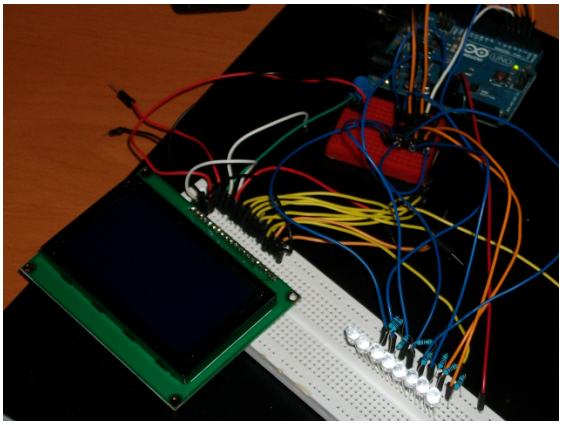

Figure 3: Step 3

5.4 KS0108 protocol implemented in code, first test demonstrates simple lines down one half of the screen (using only one of the two LCD controllers)

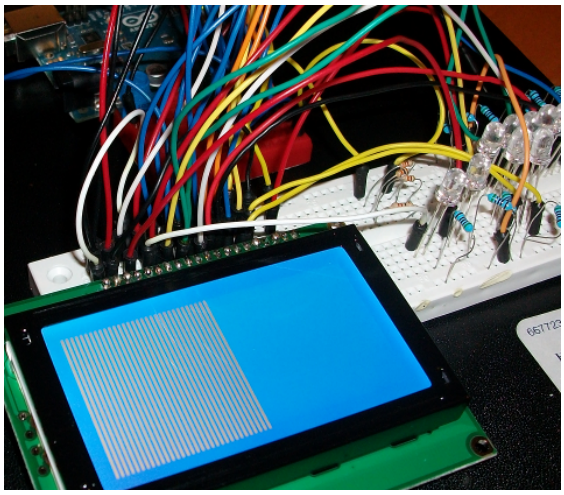

Figure 4: Step 4

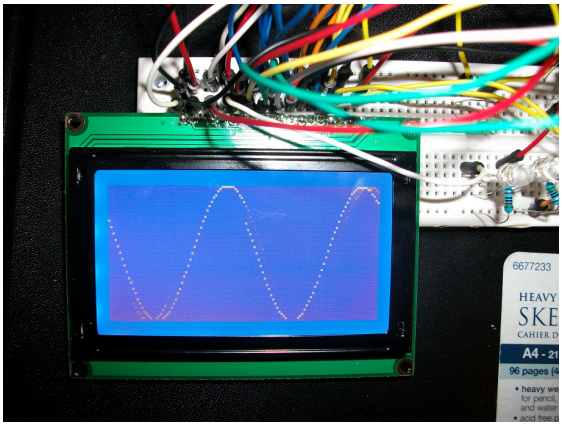

Figure 5: Step 5

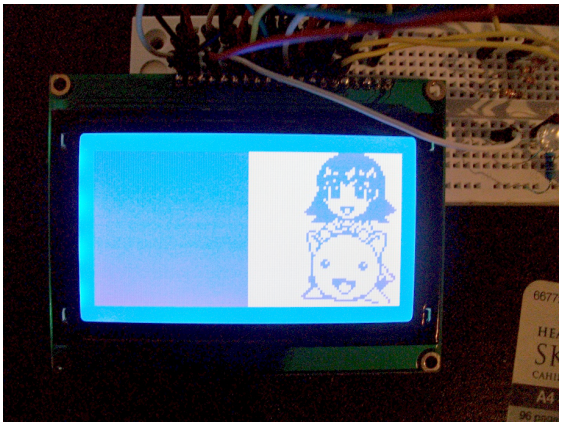

Figure 6: Step 6

5.5 Second test demonstrates a scrolling sin-wave, the picture also demonstrates the slow refresh rate of the module as we can actually see the pixels discharging (not quite white, but not quite blue pixels) - could be used for dithering effect i.e. monochromatic

5.6 First demonstration of an image being displayed using the module (ensuring that we have sufficient knowledge of how the screen works and how we can plot individual pixels

5.7 Second demonstration now utalizes both LCD controllers so that we can plot to both halfs of the screen

5.8 After deciding what game to replicate, some simple game logic was sketched up on paper

5.9 Finally we reproduced a little spin off Space Invaders replica game on the computer and then ported it to the Arduino

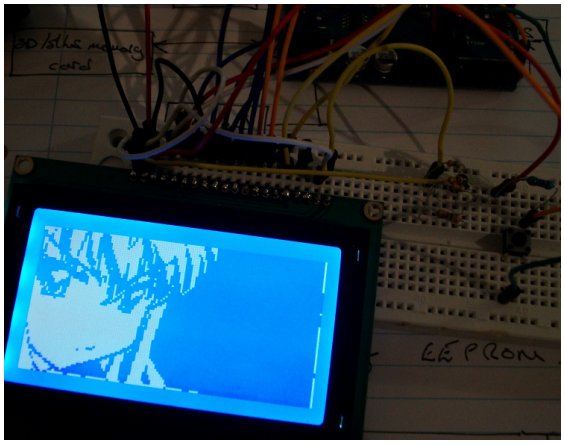

Figure 7: Step 7

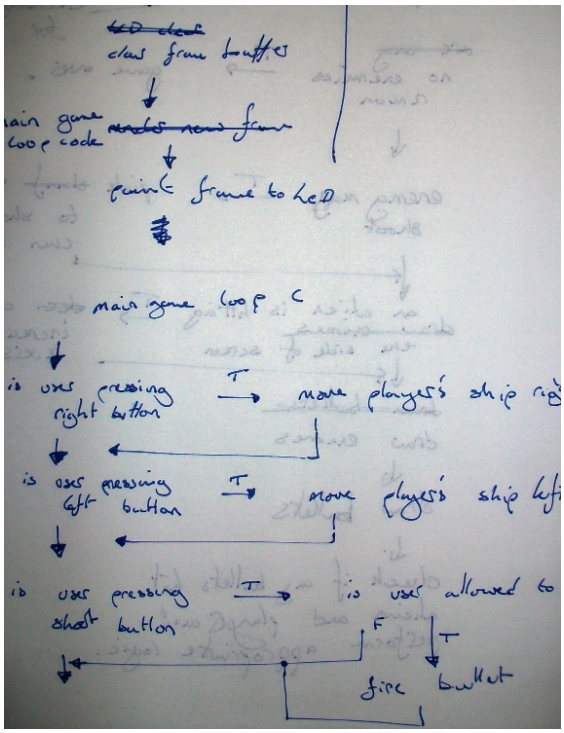

Figure 8: Step 8

portable between all of the Arduino boards, we have chosen to keep the GPIO and most of the hardware functionality very specific to our layout and thus reducing the code complexity. We read the spec-

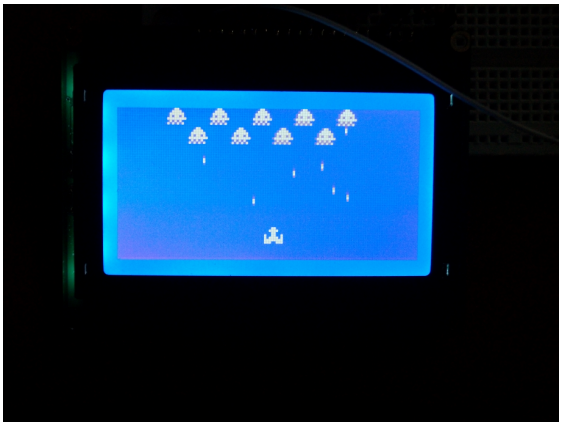

Figure 9: Step 9

ification on the MCU we were using and discovered that the Arduino code provided does not provide functionality to change the PORTC analogue inputs into digital GPIO pins as they could be used, as such we modified the code to allow us to use these pins so that we could prevent all of the wires from going down one side of the Arduino (and hence looking unbalanced) and also to allow us the extra GPIO pins of course.

This is a very simple thing to do since the AVR architecture provides very easy-to-use registers that can be modified by c-code. We changed the direction of PORTC using the DDRC register, which controls which direction each pin corresponds to. We also deleted the analogue support file provided with the Arduino to clean up some code and keep things as simple as possible.

7 LCD screen

[1] We took care to document as much of the lcd code as we could as we are sure it will provide an invaluable resource in future projects. When we first came to implement code that would talk to the LCD module the specification provided by the seller was a little ambiguous. There were many details left out that had to be tested and checked by experimentation. The LCD module acts almost as external memory. We display pixels on the LCD module by writing into the appropriate memory location on either of the 2 controllers that the LCD modules uses. Each controller accounts for one half of the screen and they act as memory arranged using 8 pages (on both controllers) of memory each page containing exactly 64 bytes. Because of the layout of the LCD module, there are 64 pixels resolution in the height meaning that the 8 pages roughly corresponds to the 8 lines of 8 pixels each. After realising how the LCD module was meant to be used (i.e. using pages instead of pixels) it became a very simple matter to form some functions in our Arduino code that would allow us to plot pixels at just the right place.

It should be noted that for speed we render everything to a frame buffer of 1024 bytes before putting all of this onto the LCD module after each frame.

The LCD module unfortunately has no double buffer support so some care was needed to ensure that the LCD repainting function was fast enough so that human eye would not be able to easily recognise the screen refreshing - this was originally done in assembly but has since been changed to

[1] We felt that this section will be of the most interest to others wishing to use these LCD modules on other projects, as such we took care to explain this a bit better than other sections.

C as the project no longer requires anywhere near the speed that was originally sought after.

The LCD is connected, and hence controlled, from the Arduino using 12 pins. There are 8 taken from PORTD of the Atmega328 chip and these are used as the DB0:7 pins on the LCD, i.e. the LCD's data bus lines. Another 4 are required to properly communicate between the LCD and the Arduino. The LCD modules as mentioned uses two controllers, one for each half of the screen, and depending on which half we are writing to: one of the two chip selecting (we have connected these two to pins C0 and C1 on the MCU) lines of those controllers must be LOW and the other HIGH in order to select which half the data on data-bus is intended for. There are two types of information passed by using the data bus, there are register commands or 8-bit pixel plotting data. When we want to control where the next pixel should go on the screen we must change the page and memory address offset on the LCD module appropriately, these are register commands and the register-command pin (on B1) must be set low to allow the appropriate controller to know this, otherwise when the register-command pin is high the data on the data bus will be plotted into the screen. After each instruction or pixel data is passed to the LCD module, it must be 'keyed-in', i.e. the LCD module must be made aware that the data on the bus is intended for the LCD and that it should use it. This is done by strobing the enable pin on the LCD module, this pin is connected to pin B1 on the Atmega328.

8 Button input

Each of the 3 push buttons are connected to pins C3, C4 and C5. They are by default reading HIGH until depressed (we chose to use pull up resisters to prevent any noise from triggering a button). There was thought no need to make use of external interrupts to implement this simple input since that would have added unnecessary code to the project.

9 Conclusion

This replica was put together very quickly as we had failed to implement the original intended project (the Gameboy emulator) had failed. Even so, we still feel that this small Space Invaders replica served as an enjoyable little project that demonstrated some of the class content.

We note that unfortunately we could not demonstrate a working RTOS along side this, although we had tested DuinOS on the board and it worked very well. The reason we couldn't use DuinOS and our Space Invaders project was simply because we store the data to be rendered in SRAM (see framebuffer comments in lcd.c) prior to putting it into the LCD and we had run out of space to use both things - DuinOS seems to take at least killobyte of SRAM. We did implement our on RTOS which was far more minimised and could have worked along-side the Space Invaders replica, but we deemed this would add unnecessary code complexity (we are only performing one task so busy-wait strategy seemed sufficient) - but might serve as a basis for future projects.

Hopefully this and the previous document will serve as a basis for any others wishing to reproduce Space Invaders, use

the KS0108-protocol LCD module or simply want to have fun.

References

[1] http://en.wikipedia.org/wiki/Space_Invaders.

[2] http://www.techtoys.com.hk/Displays/JHD12864J/ks0108.pdf.