AVR Microcontroller and C Programming: Simon LED Game – Final Draft

Muaz Rahman

Dr. Kie Eom

ECE 3520 Fall 2013 – Microprocessors: Software/Hardware

George Washington University

Washington, D.C

12-9-13

1. Introduction

1.1 Project Overview

The purpose of this project is to use and implement the information given by the previous work done both in class as well as during the labs using the Atmel AVR STK500 Microcontroller unit [MCU] and design a new use for this unit. The new use for this unit will be to first and foremost create a new project board and solder all the necessary parts together, which includes a speaker port, the microcontroller port, the serial cable port, and many others. This new project board will similarly use the Atmel AT90S4414 chip. The idea behind this project is to design the code for a Simon type led light game where the user will be presented with lit LEDs in a sequential order, which they must memorize and recreate using the push button switches in correct order. The game will have a total of five levels. Each level will increase the number of LEDs lit by one. If the user presses a switch out of order, a certain sequence will commence and the game will restart. The game will again start back up again from Level 1.

Completion of all five levels will be represented by a win method that we will describe in the later sections. Aside from creating the logic of the game itself, another one of the major aspects of this project was being able to successfully use and implement interrupt service timers into our projects. If used correctly, we are to generate successive tones and melodies to embed into our system. The game currently works as designed.

1.2 Project Materials

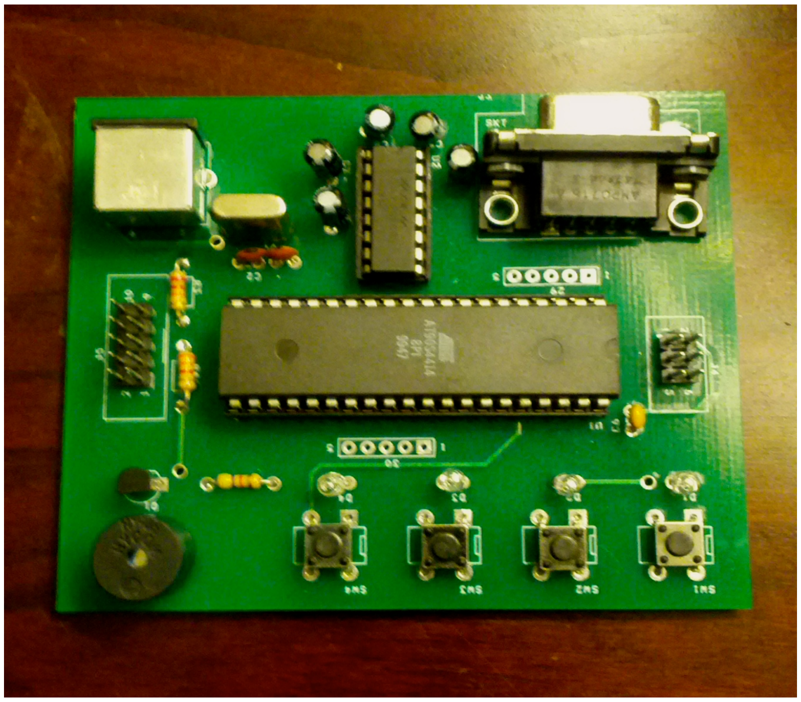
In this section, the materials required in completing the project are listed and elaborated on. Please see below.

1.2.1 Required Materials

- AVR STK500 Microcontroller Starter Kit. This includes
  - STK500 circuit board
  - ISP Programming Cable
  - Power Cable
- AT90S4414 Project Board. This includes
  - Correct soldering of all parts
  - Speaker port
  - Microcontroller port
  - etc.
- Computer with the following specs (very least):
  - Intel Core Duo Processor
  - 1 GM RAM
  - Windows 7
- AVR Studios version 4.13 for writing and debugging AVR applications.
  - Windows environment
  - Enough memory/drive capability in computer

1.3 Project Board Components

The project board used in this design comprises of a variety of different ports and accessories. It is almost like a smaller version of the STK500 board, except it includes a built in speaker. It incorporates the same chip in the AT90S4414 uC. It also comprises of a PS2 port along with a VGA cable output. It has four LEDs along with four push-buttons. It has an LED port as well incorporated into the system. For the development of the Simon Game, the major ports that were in use was the programming port, the uC obviously, the four LEDs and push-buttons, along with the speaker to play tones and melodies. This project board, in order to program and run, needs to be connected to the STK500 board to be able to receive programmable memory as well as power and current to function.



2. Problem Description

2.1 Project Overview

The project will be a Simon type light game where the user will be presented with lit LEDs in a sequential order that they must memorize and recreate using the switches in correct order. Each LED that lights up will have a corresponding tone associated with it. There is a total of five levels in the game. When the game begins, there will be an initial melody created by using timers and it will be based off the Harry Potter theme. If the user wins a round, a sequence of "running" LEDs will flash along with the melody of the song "Payphone" by Maroon 5. If the user presses a wrong key, then the game will restart and a losing round melody will also play along with a sequence of LEDs. The melody for losing any round will be based off the Imperial March theme. And finally, if the user completes all levels then a prolonged melody will begin coinciding with a prolonged sequence of LEDs. The melody that will generate if the user passes all levels is based off the Castlevenia Bloody Tears song.

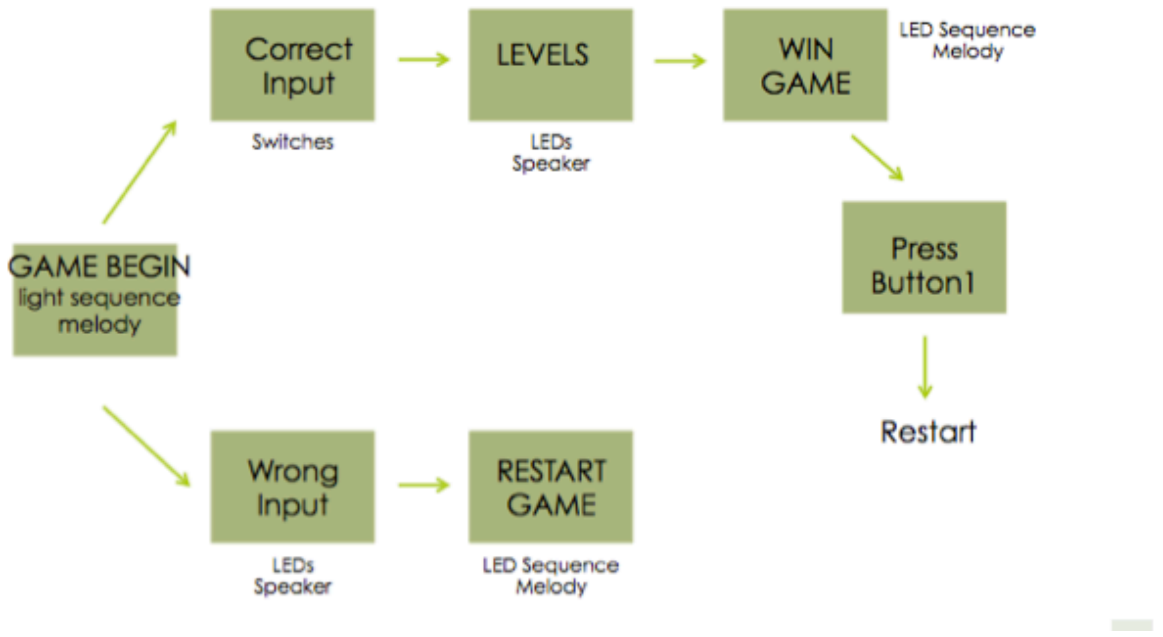
2.1.1 Assumptions

- The light game will operate on the AT90S4414 Microcontroller Unit.
- The user will understand the functionality of the game and be able to play based on documentation.
- There will be no need to reprogram the board once the game has been loaded onto the board.
- No external factors will disrupt the microcontroller or the project board during the duration of the game.

2.1.2 Event Table

Event Name	Event Stimuli	External Responses	Internal data & state
Game Begins	Harry Potter Melody sounds to represent start of game	User presses any switch on board to start	Game Begins
Game starts	First LED is lit, along with its respective tune.	User presses associated switch	If correct, next level. Else, failure of game.
Levels 2-5	LEDs light in sequence along with their respective tunes.	User presses associated switches in sequence.	If correct, next levels. Else, failure of game.
Next Level	If the user presses all the correct numbers in sequence. When entering inputs, if correct button is pressed, corresponding LED will light up confirming that it is correct.	"Walking" LEDs light up going up and down the board and the Payphone Maroon 5 Melody plays in the background.	ISR is generating the correct melodies based on the PROGMEM tables, and next question is now being generated and reproduced on the board for the user to simulate.
Game Failure	User presses switch out of sequence.	User will hear the Imperial March melody and the LEDs being light up two at a time. One time the outside two and one time the inside two LEDs. They will switch off as the melody is being played.	Game will be using the ISR to generate the melody and will be getting ready to restart the game from the beginning.
Game Success	User presses all switches in sequence	The Castlevania Bloody Tears melody will play in the background along with both flashing and walking LEDs playing in the background, signifying end of game.	Game will generate the melodies using the ISR and the game will then commence to restart if the user decides to do so by pressing the corresponding buttons.

2.1.3 Case Diagram



3. Solution

3.1 Solution Plan

The plan for creating a solution to the problem described in Section 2.1 is a three-phase plan of action.

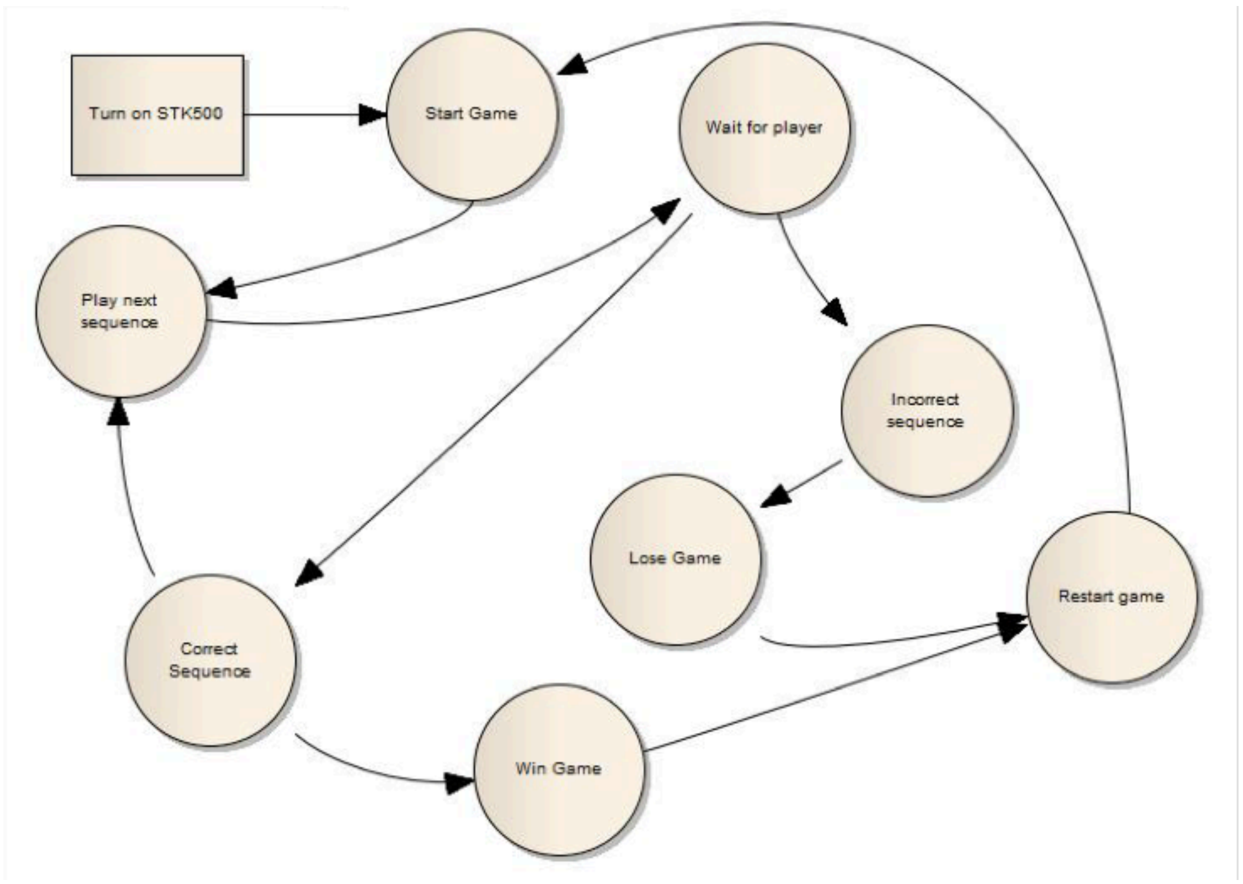
- Phase 1 – Soldering and building the project board and making sure that it works properly. Making sure all parts have been installed correctly and properly on the board itself. Downloading and programming the three demo programs onto the board as well to make sure everything is working properly. And most importantly, making sure the Atmel AT90S4414 microcontroller chip is installed correctly. And once the functionality is tested and successful, Phase 2 can start.

- Phase 2 – The Simon LED game will be planned out using diagrams before any code is developed. Once this occurs the development of the C code that will produce the desired game will begin. Once the code has been written it will be loaded onto the microcontroller unit and Phase 3 will begin. Phase 2 will be discussed more in detail in Section 3.2 (Solution Design).

- Phase 3 – After the code is developed and successfully loaded onto the microcontroller unit, testing will commence. The testing plan will be developed and then each individual test that needs to be run to achieve optimal game play will be run in sequential order to ensure the game is working properly. The test plan will be updated and modified continuously in the case that there are unforeseen problems that arises during testing.

3.2 Solution Design

3.2.1 Diagram



3.2.2 Pseudo Code

```
1. Switch (current state)
    a. Case (setup)
        i. Reset LEDs
        ii. Current sequence reset
        iii. All LEDs off
        iv. When any push button is pressed
            1. Start sequence
            2. ISR, Harry Potter Melody
            3. Delay
            4. Play next sequence
    b. Case (play next sequence)
        i. Turn on sequence LED
        ii. Delay
        iii. Turn off sequence LED
        iv. If current sequence position = current level
            1. Wait for player
```

c. Case (wait for player)
   i. Turn on LED every time corresponding button is pressed
   ii. Debouncing
   iii. Delay
   iv. If pressed button = correct button
      1. Increase Level
      2. Play win round sound/led sequence
   v. Else
      1. Lose Game, back to beginning.
      2. Play lose game sound/led sequence

d. Case (win game)
   i. ISR, play win game sound melody
      1. Castlevania Bloody Tears Sound
   ii. Light up sequential LEDs
      1. Walking LEDs
      2. Switch off LEDs two by two.

3.3 Detailed Procedure Explanation

This section will go through a detailed explanation of every type of method and algorithm and functionality that is being used in the implementation of this game.

3.3.1 Header Files and Libraries

We first start off the code by making sure to include all the necessary libraries and header files as needed. This includes the delay, progmem, and interrupt libraries. We then go on to define some declarations which basically details the numbers associated with the certain situation that I am trying to manipulate. These names specifically are not used directly in the code but essentially they are there to make sure any reader going over the code can understand what exactly is going on.

3.3.2 Program Memory and Arrays

We then go on to define the specific arrays and program tables that need to be initialized and set up. The first couple arrays define all the necessary LEDs and Buttons are noted. The next step is all the progmem memory information. This mostly includes all the frequencies that are necessary to the implementation of the code. This includes the sounds associated with every button as well as all the melodies that we will be using. The start game melody is based on the Harry Potter theme music. When the user loses a specific round, the lose game melody will begin which is the Imperial March melody. When the user wins a specific round, another melody will commence which is based off the Payphone Maroon 5 melody. And lastly if the user wins all five levels, the Castlevania Bloody Tears melody will begin. The other program memory tables we will be using are for the five levels that we are trying to make. Each level is already predefined in their respective tables for quick access and functional use.

3.3.3 Variables and Functions

We initialize a variety of different variables for the purposes of this project, some of which are for debouncing. Some of which are for control purposes, some of which are for the various case statements, etc. We also have various functions that we use. Most of these functions are set to play the melody and light sequences themselves. We have functions built when the user wins a game, loses the game, wins the round, etc. We also have an initialize function as well to initialize all the ports.

3.3.4 Case Statement

This is the main methodology of the game. Using the case statements we are basically going through every single case that can possibly happen when playing this game. What happens is that our first case is our initial begin of the game, where the user presses a button and if so the game begins and generates a melody. The next few cases deals with the five different levels found in the Simon game. In each case statement, the board will first play the specific note and LED as necessary and will then wait for the user to see which button they will be pressing. If the button presses match the sequential LEDs played, then the case statements will proceed onto the next levels. If the wrong buttons are pressed then the case statement will reset and we will go back to the beginning.

3.3.5 Playing Melodies

We already described which melodies this game will incorporate in the above sections. Now to how the melodies are implemented. They are implemented through the use of Interrupt Service Timers. When used properly we can generate various frequencies that we pass along to the speaker port that we toggle in this case to turn on. When this occurs we will then pass on to the speaker the specific melodies we want to play from the program memory tables, and by incorporating the table lookup method the speaker will successfully play the necessary sound and music. Inside the ISR, there are various case statements. These case statements refer to each state of the game. In each state there are a variety of inside case statements. The reason for this is for the ISR to know where exactly in the game the user is and to make sure to play the correct melody.

3.3.6 Reading Player Input

When reading the push buttons switches the user inputs into the system. What happens is that we incorporate a variety of delays and debouncing methodologies to make sure that the timing and all processes are correct and to the point and efficient. To do this we first and foremost use debouncing as our main course of action and then from that point on we compare if the button pressed is the correct one based on our table lookup method. If the user presses the correct switches for all the buttons in that level, then game will proceed on. If any of the buttons are wrong, and we can verify this by cross checking with our program tables, then the game will restart automatically.

3.4 Source Code

Source code is attached separately. Look under source code in project zip folder.

4. Conclusion

The Atmel AT90S4414 microcontroller chip and project board was successfully the Simon LED Game was a success. The code was written properly and the logic was implemented with proper use and functionality. The game worked properly, and any person of any age will enjoy it. The unique game features such as the various melodies and sounds and tunes and lighting up of LEDs also worked correctly.

This project helped me personally to learn a lot more about AVR C programming and showed me the various ways these kinds of embedded projects can be implemented in the real world today. It was a precursor to various kinds of more complex and thorough machines and circuitry and systems that are being built today.

Building this game was a great experience.

5. References

[1] Atmel Corporation, ATMEL STK500 User Guide, Atmel Corporation, September 9, 2009,

http://www.atmel.com/dyn/resources/prod_documents/doc1925.pdf

[2] Arild Rødland, Novice's Guide to AVR Development, AVRFreaks, September 16, 2009,

http://www.atmel.com/dyn/resources/prod_documents/novice.pdf

[3] Atmel Corporation, AVR Instruction Set, Atmel Corporation, October 15, 2009,

http://www.atmel.com/dyn/resources/prod_documents/doc0856.pdf