Project 2: Farm Game

By: Rohan J. Dani

Intro to Microcontroller Interfacing

Project Time Frame: March 6th – March 22nd

# **Report Summary**

In this report, I’ll cover what this game we created does and my adventure through creating this game. It will also cover the Farm’s FSM I had to make in order to code it in as the crux of the game and the description of the contents of my Debug folder and what the compiler and linker relate to these files.

# **Project Description**

For this project, a game was developed to be displayed on the LCD screen of the boosterpack board and controlled with your laptop keyboard via a UART connection. There are three baud rates that had to be used 9600, 19200, and 57600. The different baud rates would be used as levels of difficulty of the game as the month length would decrease in correlation to the baud rate. You have to implement a game where you farm by planting, watering, tending and harvesting plants in different plots on your farm. The Farm as a whole has Money, Farm health, and Time stats that are displayed on the Booster LCD Screen. It has multiple keyboard inputs that will change the those stats like, p (plot), r (rain), t (tend), and h (harvest). The game should also be able to change in difficulty by pressing the l keyword and it displays ‘E’, ’M’, or ’H’. We were not allowed to use any global variables, had to write in Hardware Abstraction Layer format, and were required to use two structures which were one for plot and one for farm.

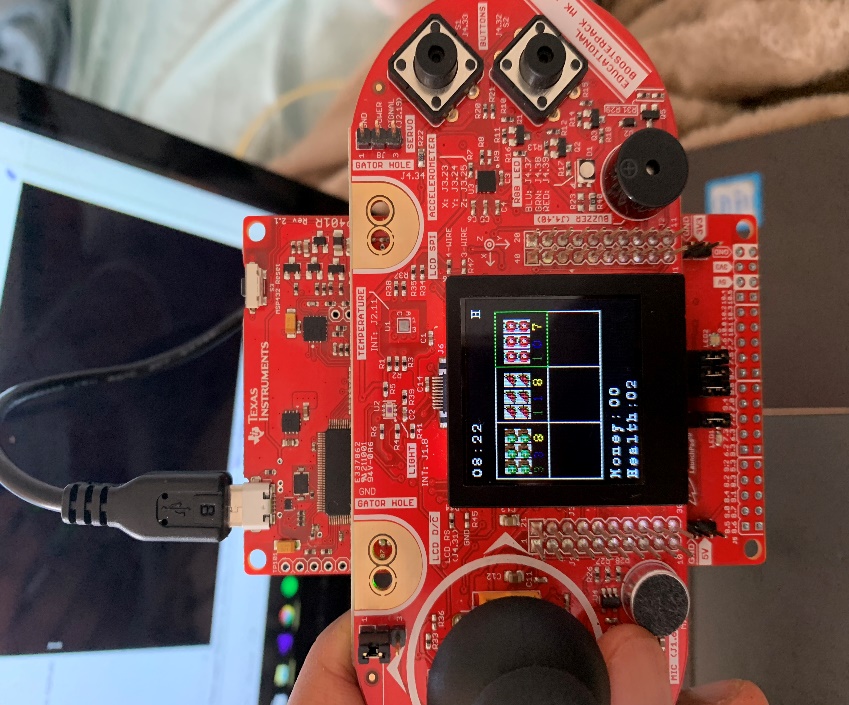
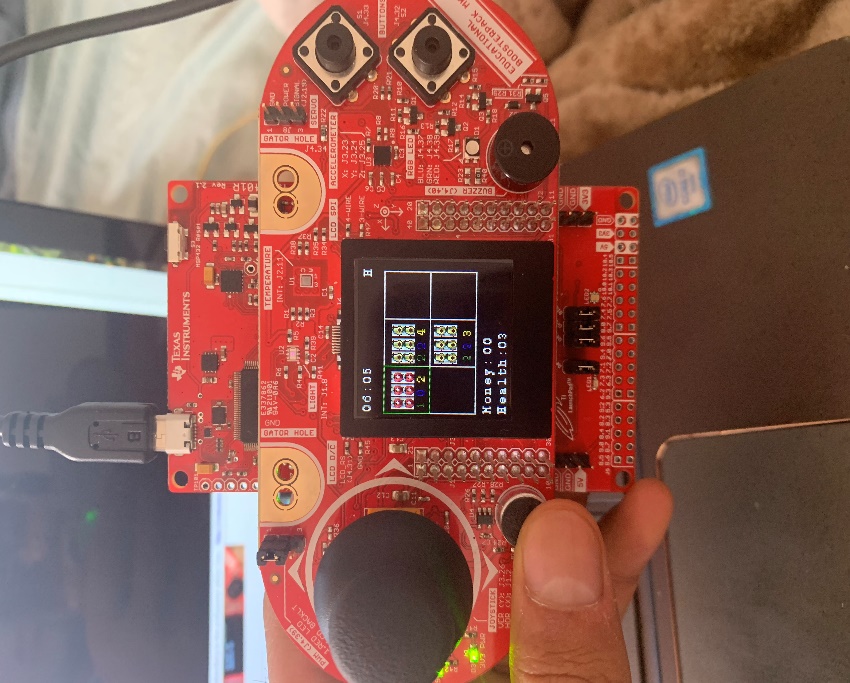
 

Figure 2: Hard difficulty with Plant and Dying Stage

Figure 1: Seeding, Growing, Dying Stage

# **Farm’s FSM**

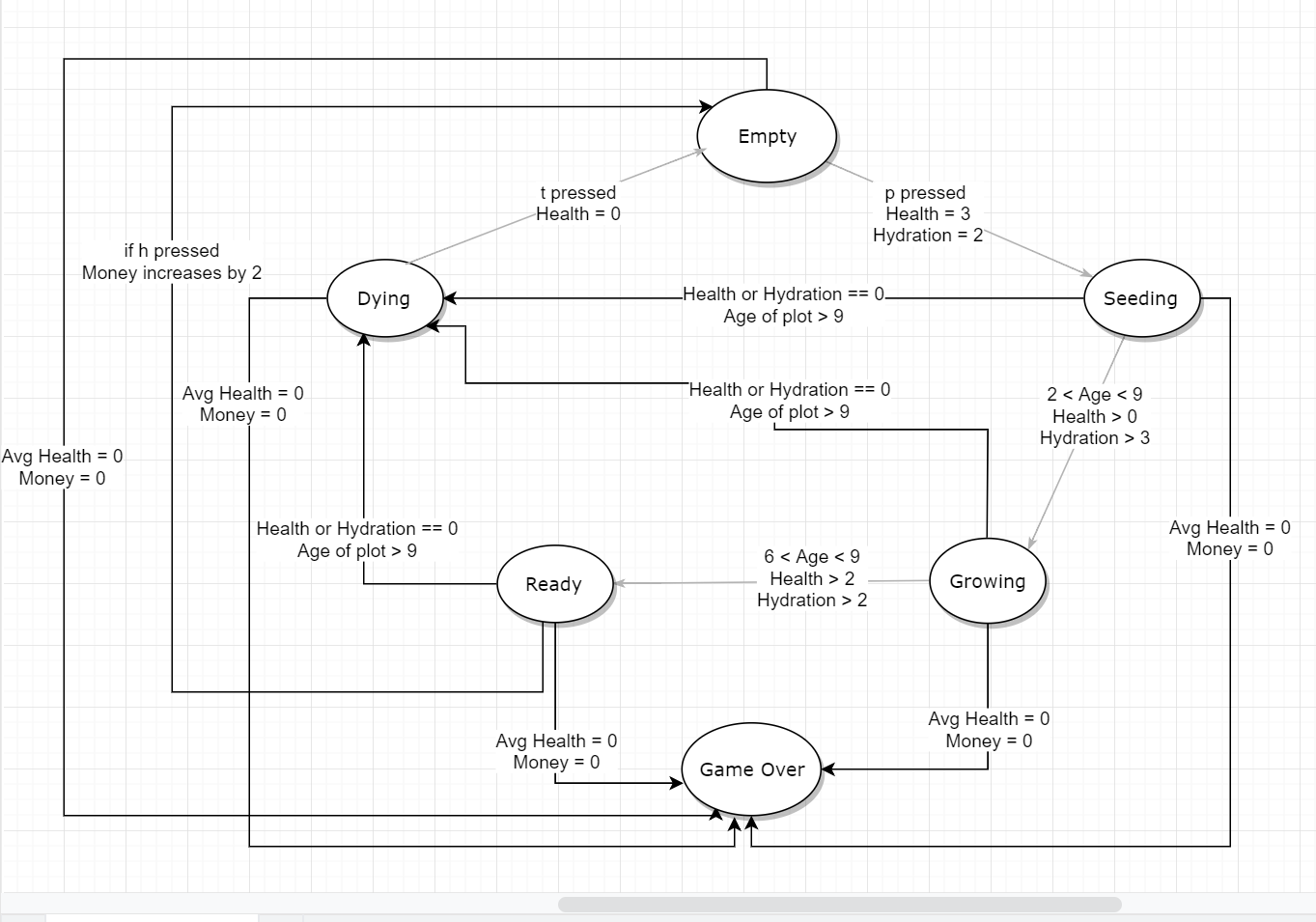


Figure 1: Farm Finite State Machine

This FSM is a cycle of 5 main stages which are Empty, Seeding, Growing, Ready, and Dying that are triggered by key inputs that cause 3 stats to change which are Health, Hydration, and Money. Age is controlled by the timer that we have created and the month count on each plot is equal to its Age. As you can see you can go to the Dying stage from three different stages Seeding, Growing, and Ready which is controlled by health, hydration, and age. As you can see the Game over can be reached from any stage and that occurs when the average health and money is 0.

# **Debug Folder Description**

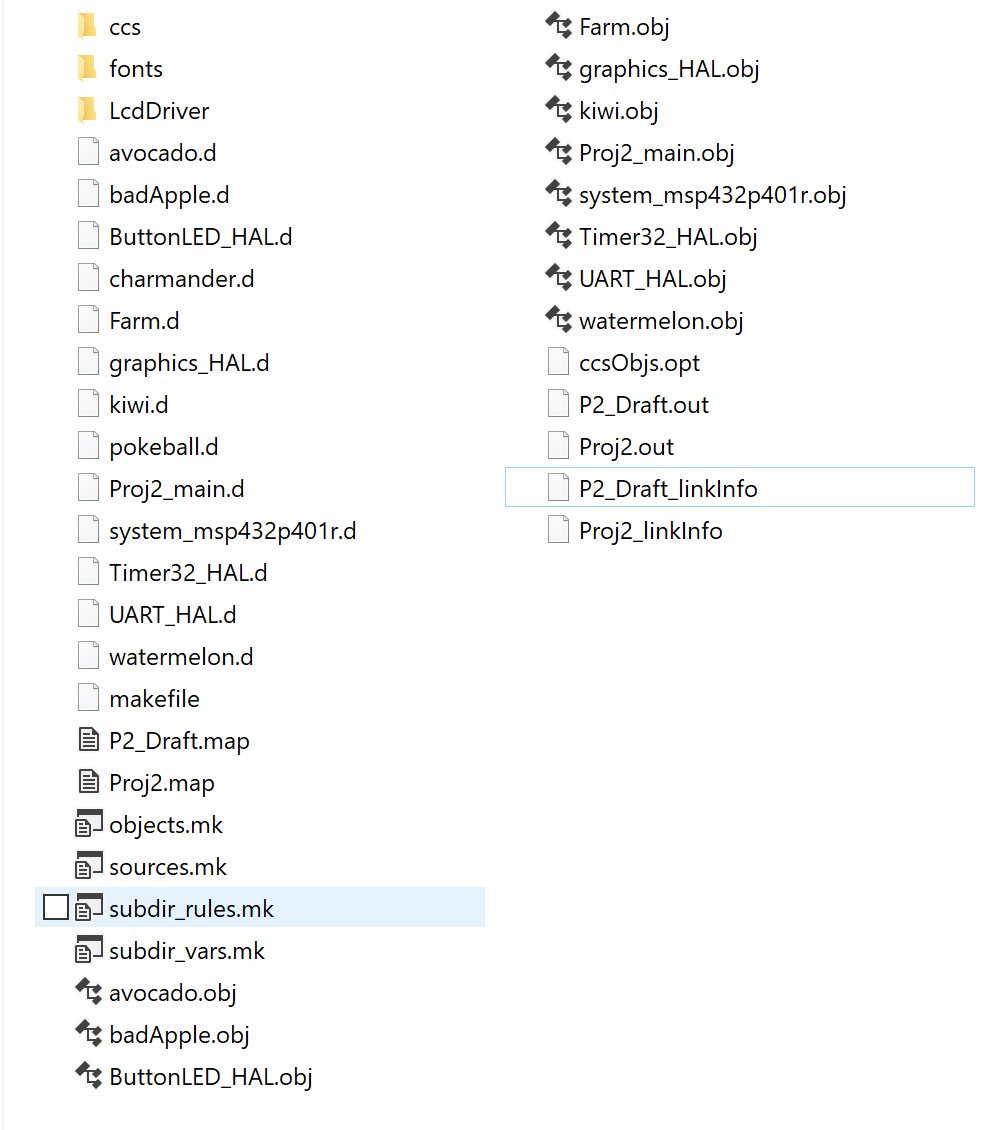


Figure 2: Contents of Debug Folder

|  |
| --- |
| .d |
| .map |
| .mk |
| .obj |
| .opt |
| .out |
| .xml |

Table 1: All the file extensions in the Debug folder

**.d file**

As you can see a .d file is one of the files that is in this directory. It is a file that is generated by GCC, which is the C compiler. In this it has all the dependencies for each of C files which are then used to create the compiled objects which are the files.

**.map file**

.map file is another one in my directory as it is another GCC generated file. It seems like it maps out the amount of data and location that is used in that specific C file. It helps you know when you are running out of program memory.

**.mk file**

.mk file is a tool mainly for building the executables from the many modules. A makefile is also generated and that is what creates the executable. It builds an executable from the multiple .obj files that are within this directory.

**.obj file**

A .obj file is created when you compile multiple of the source files and that is created with the compiler. The object file is mainly in machine code that allows the linker to see information like native code and the various symbols that are included in the code.

**.opt file**

A .opt file seems like to compile the list of all the object files that are generated from the code written. Seems like it adds the driverlib includes that were in the source files as well. This must make it easier because everything is in one place so it can find the files with ease.

**.out file**

The .out file is the executable that is created from this entire process and it is the final product to run the project. It is made using the .mk file which generates from the .obj files.

**.xml file**

This .xml file seems like it prints out all the linker information and when the project is built in CCS you see this information first as it includes all the .obj files that were used to create the executable to run this project.

# **Conclusion**

In conclusion, I learned a lot from this project as it wrapped together all the techniques we learned from the homework from the UART communication, HAL functions, structs, FSM, Image reformer, etc. This will definitely help me in the future as I am becoming more and more familiar with this project. I also learned about the different stages of converting to an executable from analyzing the Debug folder which can be pertained to any software project.