ECE 2534
Project 2
Spring 2018

General guidelines for any assignment

- Before you start working on your assignment:
    - Read the assignment description (this document) in its entirety.
    - Read the starter code carefully and in its entirety.
    - If available, run the provided *.out on your board and see what the expected behavior is.

- Every time you sit down to work on your assignment:
    - In the beginning, go to Piazza, and read the "notes and FAQ," and preferably other questions/notes related to the assignment you are working on ***even if you took a two-hour break***.
    - In the end, push your changes to GitHub, ***even if you are going to take a two-hour break***!

1. Overview

The purpose of this assignment is for students to gain experience using the microcontroller's UART, timers, display, and implementing game logic. You will develop a Farming game that is displayed on the TI BoosterXL's LCD Screen and controlled with your laptop keyboard via a UART connection. A golden solution executable is provided along with this document. The golden solution and this document play complementary roles and are consistent with each other to the best of our knowledge. If some descriptions are not clear from the text, please refer to the golden solution for your answer. If the golden solution does not provide a convincing answer to you, feel free to use Piazza.

2. Description:

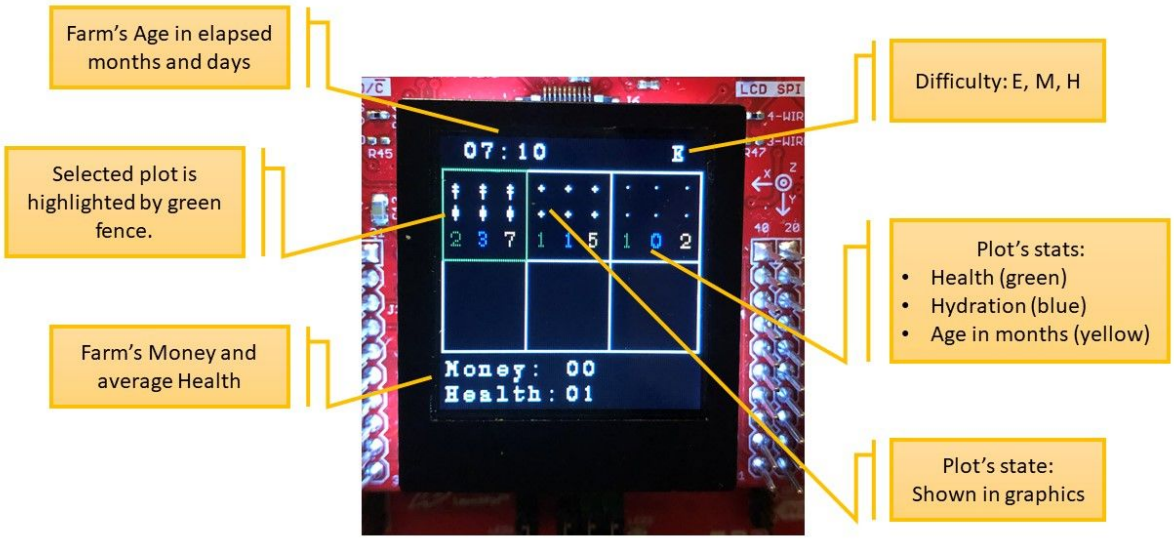
Project 2 implements a game where you farm by planting, watering, tending and harvesting plants in different plots on your farm. The Farm ***as a whole*** has Money, Farm health, and Time stats that will be displayed on the BoosterXL LCD Screen. Your farm's ***individual plots*** have Age, health, and hydration levels that correspond to the state of the plot's crop. You will interact with your farm by sending commands from your laptop through a UART connection to the MSP432P4. The unit of Time for the farm is one month, which for your game is a specific number of seconds depending on the difficulty level.

2.1. Initial Game State:

The initial game state starts with the farm with no crops planted in the plots. The game stats are: Time is initially zero, Money is 3, and Farm health is zero. The game timer for the game will not begin until the player presses any key on the keyboard while in MobaXterm. The Farm health will remain zero until at least one crop is planted in one of the plots.

2.2. Application Display:

Your Farm should have a border fence and the farm must be split into 6 individual plots. Above the farm in the top left corner, the time that has passed should be displayed. In the right top corner, the difficulty level should appear represented by 'E', 'M', or 'H'. Under the farm, you are to display the Money and Health levels of the farm. As the game progresses, the state and stats of the plot should be represented by either graphics or characters. The plot that the farmer is working on is highlighted by a green fence. The below figure shows a screenshot with three planted plots with the top left plot being the active (selected) plot.



2.3. Player Game Input:

Your farm game will receive input characters from the player, through the UART connection, to perform the following actions:

* (Any character) – To begin the game.
'l' – To change the level of the game to cycle through Easy, Medium, and Hard.
'p' – to plant the plot of land
'r' – To water the plot (r stands for rain!)
't' – to tend the plot of land (i.e remove the weeds)
'h' – Harvest the selected plot
'w', 's', 'a', 'd' – Moves around the farm to select a plot. w/s for up/down and a/d for left/right.

2.4. Farm Plot's States:

Each of the plots can have one of the following states:

1. **Empty:** No plants, and does not factor into the overall health of the farm.
    a. 'p' will update the plot to **Seeding**, with a health of 3 and hydration level of 2.

2. **Seeding**: The plot remains in this state until one of the following conditions is met:
    a. When the Age is greater than 2 months but not greater than 9 months with a hydration level greater than 3 and health level more than zero, the state updates the plot's state to **Growing**.
    b. If the Plot Health or Hydration falls to 0 or the age of the plot exceeds 9 months, the plot's state updates to **Dying**.

3. **Growing**: The plot remains in this state until one of the following conditions is met:
    a. When the Age is greater than 6 months but not more than 9 months with Health and Hydration levels of greater than 2, the plot's state updates to **Ready** (to harvest).

b. If the Plot Health or Hydration falls to 0 or the age of the plot exceeds 9 months, the plot's state updates to **Dying**.

4. **Ready**: The plot remains in this state until one of the following conditions is met:
   a. If the User harvests by pressing 'h' on the keyboard the plot will become **Empty** and the farm gains 2 levels in money.
   b. If the Health or Hydration becomes 0 or the Age exceeds 9 months the plot goes to **Dying**.

5. **Dying**: A Dead plot has a health level of 0 that *contributes* to the overall farm health. The plot remains in this state until the below condition is met:
   a. If the User tends to the plot by pressing 't', the plot goes to **Empty** and the health level of the plot is reset to 0.

2.5. Farm Plot's Properties:

1. **Hydration**: It can change in two ways
   a. Raining ('r') the individual plots will increase the hydration level of the selected plot by 1 up to a hydration level of 5.
   b. Hydration levels will decrease by 1 for each plot that is not **Empty** or **Dying** after each month.

2. **Health**: It can change in two ways:
   a. Tending the individual plots will increase the health level of the selected plot by 1 up to a health level of 5.
   b. Health levels will decrease by 1 for each plot that is not **Empty** or **Dying** after each month.

3. **Age**: After the plot enters **Seeding**, the age increases by 1 as each month passes.

2.6. Farm Health:

The farm health is calculated by the average (simple integer division) health of all non-empty plots on the farm. ***If the health of the farm reaches zero and the farm does not have any more money, the game is over.*** When the farm dies the LCD screen should update to a game over screen. You can create any game over screen you like, so be creative.

2.7. Game Difficulty:

The leveling mechanism in the game will cycle Easy -> Medium-> Hard -> Easy. The Settings required for each level can be seen in Table 1.

Table 1 Game Level Settings

Level	Easy ('E')	Medium ('M')	Hard ('H')
Baud	9600	19200	57600
Month length	15s	10s	5s

2.8 Graphics

Each state of the plot should be represented by a unique design. ***This design should be different from the golden solution but does not need to be very advanced.***

3. Starter Repository

The starter repository provides a basic project as a starting point for your code called Proj2. You are required to use two structures one for the farm and one for the plot. ***The two structures in farm.h are provided as a suggestion. You are welcome to add or remove members to these structures. Add other sturctures to your code, etc.*** Some HAL files have examples of functions they should contain. You can delete them and declare and define your own functions. You are also welcome to delete the main function and write everything from scratch.

The starter code also includes an empty project called playground where you can use as you wish. The content of this project is not graded unless you ask us to do so in your report. A partial grade will be given to the content of this project at grader's discretion. Please note you can also add as many projects to the starter repository as you wish. If by the deadline, you have some code that works, but some code you have written that when added to the main project breaks everything, feel free to submit that as an extra folder in the repo. You need to describe this very clearly in your report. Again the code that is not in your main folder is graded at grader's discretion.

4. Proposed Milestones and Guidelines on Getting Help

I recommend you start by playing with the golden solution and comparing and contrasting with Section 2 of this document. We have tried to eliminate any inconsistencies. If you feel there are inconsistencies between the two, please use piazza to let me know.

I propose a three-prong approach for completing this project.

First prong: Hand draw an FSM that shows the farm plots states and transitions. You will eventually draw this FSM for your report when you are done with your project. Furthermore, write the pseudo code of your main function on paper or as a comment in CCS. Think what your high-level functions are. Then, think about how you can break down those high-level functions into smaller functions. As you develop your code, go back to your paper and improve your pseudo code and the list of your functions and their hierarchy, etc. If you are unsure, it is OK, you will get better at this as you write more and more code. ***I am asking all the GTAs and ULAs to check your handwritten notes before answering any questions you have.***

Second prong: I suggest that you simplify this project and implement the simple form first. Here are some suggestions on how to simplify the project while building your solution.

1) Assume the farm has 1 plot only.
2) Instead of graphics, use characters to represent the state of the plot. For example draw "e" for Empty and "s" for Seeding, etc.
3) Do not incorporate time. In other words, no age, no hydration or health decrease per month. Instead, implement the state transitions based on user inputs of 'p', 'r', 't', and 'h'.
4) Disregard money, farm health, difficulty, and time.

Even the simple farm should not be built in one step. Write your code in small steps and test it as you go.

Third prong: while you are building the simplified farm, you should build bits and pieces of the large project and test them independently without being part of the simple project. Here are some suggestions on different pieces:

1) Build a timer that shows months and days where each month is 15 seconds and display them.
2) Draw the farm plots.
3) Draw the graphics for each of the plot's states for the top left plot(one function for each state.)
4) Tweak the functions of item 3 such that they can draw the state for any plot of the farm based on the input parameter.
5) Implement cycling through baudrates by pressing 'l'.
6) Draw the gameover graphics
7) Implement and test all the HAL functions you need.

At some point, you should start incorporating the elements you have developed in the third prong into the simplified farm model you developed in the second prong. ***The changes to your simplified farm should always be small and incremental.*** You should always keep track of the last change you made that suddenly made things crash!!! It is never a good idea to sit down and write code for an hour before pressing the compile and debug buttons. ***I am asking all the GTAs and ULAs to ask you what worked before you made what set of changes that made things stop working. You cannot ask for help by simply saying "It does not work and I have no idea what is going on with my code."***

In order for you to have an official place where you test small functions independent of your main project before incorporating them into your main project, I am including a second project in the starter repository. I call this project playground. The advantage of this playground is that any time you push, this playground is also saved on GitHub and you do not have to worry about backing up your local disk for small pieces of work that you have done but is not included in your project yet.

5. Programming notes

Please pay attention to the following programming guidelines:

1. No global variables are allowed.
2. With the exception of 1 and 0 and the parameters used for graphical interfaces, you are not allowed to use hard-coded numbers in the code. All numbers should be used as macros.
3. You are required to use a Hardware Abstraction Layer to make your farm game portable to any other platform if you wish to. UART, Timer, and Display each should have their own set of HAL functions placed in a separate file. We have provided empty files for you to place your functions inside. You can look at ButtonLED_HAL.h and ButtonLED_HAL.c for guidance. No Driverlib function should be used in higher level functions. In other words, all the Driverlib and Graphics library functions should be kept within HAL files.

4. You are required to use two structures: one for the farm and one for the plot. The two structures in farm.h starter code are provided as a suggestion. You are welcome to add or remove members to these structures.

5. The whole project should be reasonably commented. If you are unsure what level of commenting is expected, check with your professor or your TAs. All functions should have a header comment. At the minimum, this comment should include the purpose (general functionality) and input/output of the function. For more complicated functions, a simple pseudo code is recommended but is not needed if the body of the function is sufficiently commented.
6. The main function should not be longer than 100 lines. This 100 lines should include comments and the typical empty lines as well as the typical line indentation and breakages. For example your code should look like this

```
 if (farm.DaysPassed == 31) {

              farm.DaysPassed = 0;

              PassTime(&farm);

}
```

and not this:

```
if (farm.DaysPassed == 31){farm.DaysPassed =
0;PassTime(&farm);}
```

6. Report

You need to create a formal technical report (an example will be provided) that has the following sections:

1. **Report Summary:** In less than 100 words summarize the purpose and the content of the report.
2. **Project description**: Use between 100-300 words to describe what your project can achieve. Ideally, this should match the project description, but you will have to write it in your own words and you do not have to include all the details. If your project is not completed describe what is different from the golden solution. If you have implemented any bonus parts, you need to briefly describe them here as well. Include a few images that show the state of the game under different scenarios. Remember to number your figures and refer to your figures.
3. **Farm plot's states**: Draw an FSM (mealy machine) for farm plot's states. Use powerpoint or some other software. Hand-drawn plots receive no points. Remember to number the figure that includes the graph. Describe the graph in a short paragraph.
4. **Debug folder:** Study the contents of the Debug folder. Ignoring the folders inside, organize the folder by file type and take a screenshot for your report, making sure to number your figure. Create a table with the list of all the file types you have found in the folder. For each file type, write a short paragraph on what that file type represents and what its purpose is in that folder. Describe the role of the compiler and linker in relationship to any of the file types if needed.
5. **Bonus features:** If you have implemented bonus features, clearly describe how to play the bonus features. Include images of your advanced graphics or any bonus parts.

7. Submission

You will have to put your report in your original starter repository and submit all your code to GitHub by pushing from CCS. Submitting report or any part of the project outside GitHub incurs at least 200 points penalty.

8. Bonus features

You can get up to 200 bonus points by implementing the following:

- Add animals to your farm. For example, consider paying to buy a cow and placing it in one of the plots, consider feeding the cow (it should not be free), consider using harvest to provide feed for the cow, consider making money by milking the cow or selling the cow after it has gained weight. Consider showing the age of cow and it weight, etc. Of course, you can choose any other farm animal. This is your game and you define your rules. Make sure Section 5 of your report describes clearly how to play this custom part of the game. 100-200 points based on complexity.
- Add propagation of dead plants, if your plants are left dead in one field make the death propagate to healthy fields if left unattended after a month. 50 points
- Add more advanced graphics compared to the golden solution. 50-100 points based on the complexity
- Propose new features. You can add features that you like but you need to email Jon Bunting (buntingj@vt.edu) to get approval for your feature and get a confirmation for the number of points you will receive for your proposed features.