Embedded Systems Guitar Hero Project

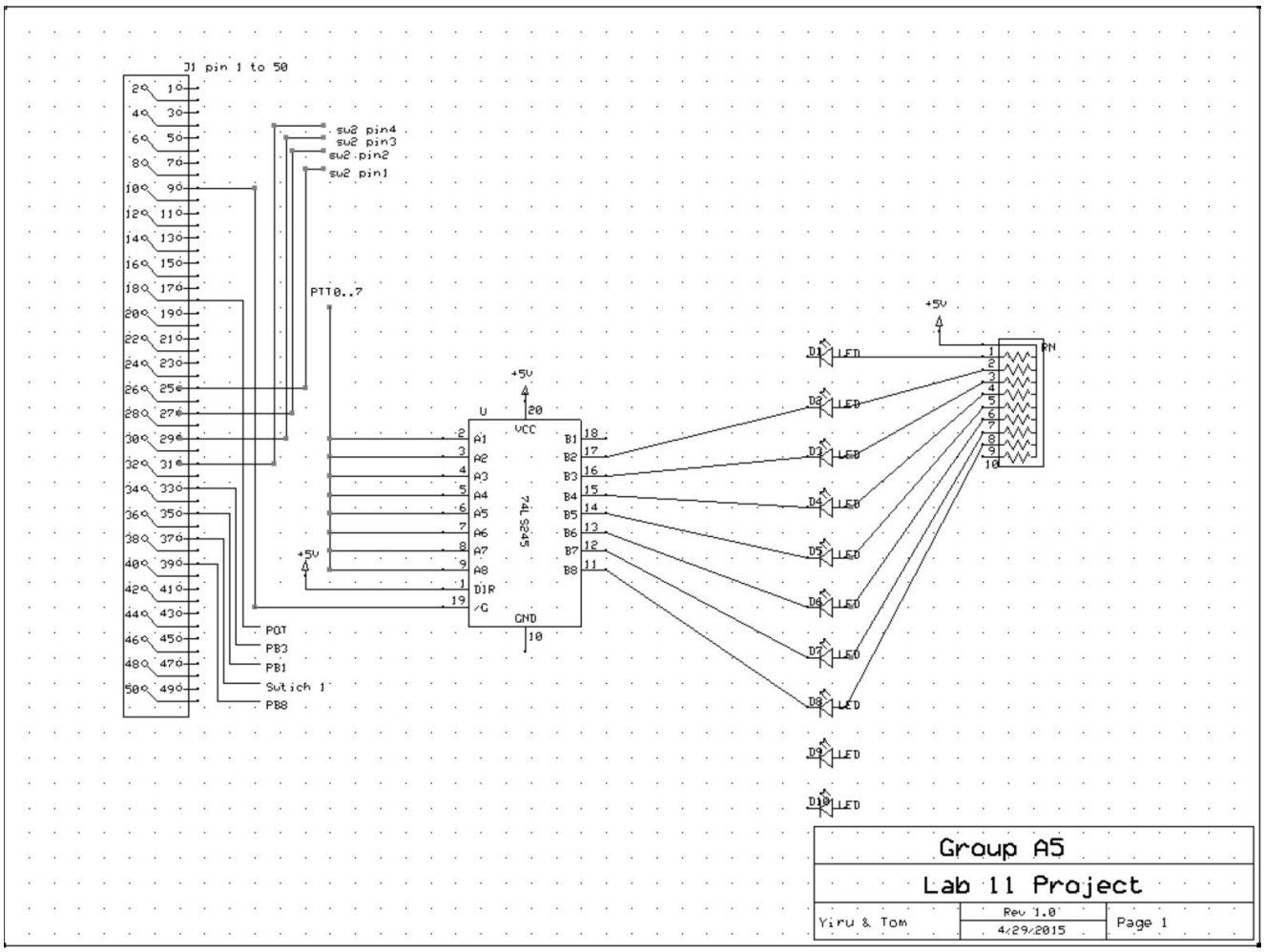
Yiru Yao yiruy@andrew.cmu.edu
Thomas Mayo-Smith tmayosmi@andrew.cmu.edu

1. **Short description**

We are implementing a simple version of Guitar Hero with only one note as opposed to four notes like the real game. Users press a button when the last LED on a "bar LED" turns on in order to score. The "note" or lighted LED will "scroll" on the bar LED with increasing brightness as the note gets closer to the position where the player needs to press a button in order to "play it". Some key features include a potentiometer which controls the "scrolling speed" and every 3 seconds and the game will enter a bonus mode for 5 "scrolling intervals". The game stops after 2 minutes.

- **Main Components**
  - **i.** Serial Communications → send "note" data
  - **ii.** PWM → make LED's about to be mimicked brighter
  - **iii.** Timer → used to scroll LED's
- **Hardware components list**
  - **i.** Project board and Module board
  - **ii.** USB to Serial cable
  - **iii.** USB cable
  - **iv.** Bar LED
  - **v.** LS245 chip

Hardware block diagram



- **Interrupts**
  - **i.** Timer Interrupt
  - **ii.** ATD interrupt
- **Inputs**
  - **i.** Button1: registers users reaction to a data bit. (PRESSED or UNPRESSED)
  - **ii.** Reaction data: (serial interface input): indicates whether the individual LED's on the bar LED are on or off
  - **iii.** Potentiometer_Value: controls "scrolling speed". Ranges from 0 to 15
  - **iv.** Switch 2 value: controls the brightness of individual LEDs. Ranges from 0 to 15
  - **v.** Button8: indicates the user would like to start the game
  - **vi.** Switch1 "bit1": used to indicate the user would like to reset the game

- **Outputs**
  - **i.** scrollPeriod(LCD display): the speed of LED scrolling on the LED Bars. Ranges from 0 to 15
  - **ii.** score(LCD display): the current score of the player. Range from 0 to 31
  - **iii.** timeElapsed(LCD display): the time elapsed for the current round of the game. Ranges from 0 to 120 in decimal seconds
  - **iv.** data: first 7 bits in data will be displayed on on the LED row. Range from 0 to 0xFFFFFFFE
  - **v.** "brightness" (PWMDTY0): indicates how bright each LED will be. Range from 0 to F
  - **vi.** timeOut: indicates whether there is still time left in the current round of the game and causes the LCD to display "Time Out" when it is true. Range: 1, 0 (True or False)
- **List of state variables**

  timeOut - indicates if the game is running or if time is out for the current round and causes a transition from state GOING to waitForReset Range from 1 to 0

  buttonVal- indicates if the players score has been incremented after reacting to a particular LED and causes a transition from state GOING to SCORE Range from 1 to 0

  Button 8 (referenced in the program as PORTA_BIT0) - indicates the user is done choosing settings and causes a transition from chooseSettings to GOING Values are "Pressed and Unpressed"

  Switch1 "bit1" (referenced in the program as PORTA_BIT1) - indicates the user is done observing score from previous game and causes a transition from state waitForReset to chooseSettins Either switched or unswitched

  PCB[1].NextTime - indicate if it is the next time during which we update the data and causes a transition from GOING to UPDATE Range: 0 to 2^32-1
- **List of high level requirements**
  - **i.** R1. The controller shall implement the following states: chooseSettings, GOING, UPDATE, SCORE, waitForReset
  - **ii.** R2. User shall set speed and brightness in the chooseSettingState until he/she indicates he/she is finished.
  - **iii.** R3. After user indicates he/she is finished choosing settings the user shall start the game.

iv. R4. The game shall scroll a pattern of LED's at the user's chosen speed and brightness.

v. R5. The user shall be rewarded for reacting correctly to an LED.

vi. R6. After two minutes the user shall be shown how many times he/she reacted correctly.

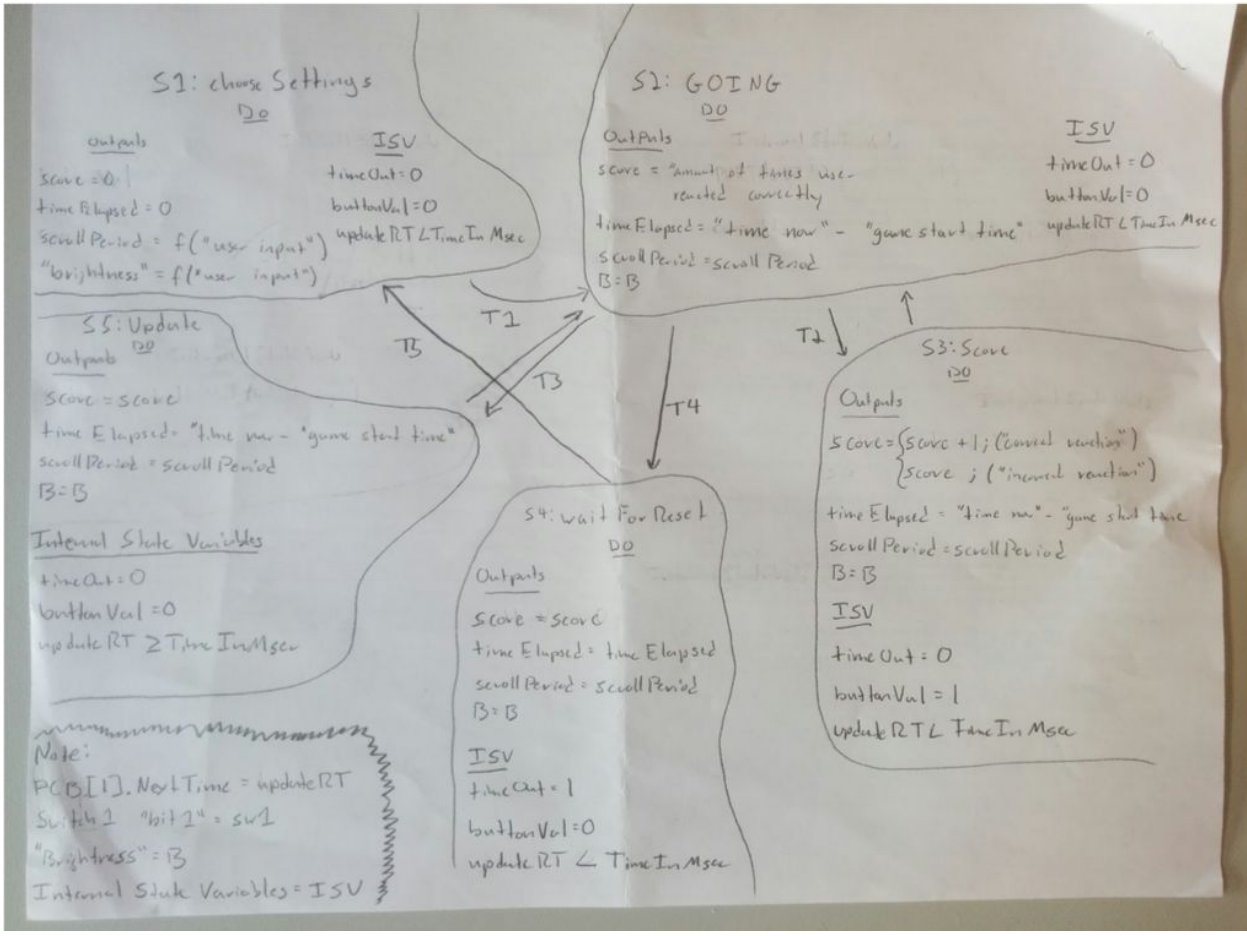
vii. R7. After the user indicates he/she is done viewing results user shall return to chooseSettingState

viii. R8. The LCD shall display…

ix. R8A. the time elapsed and the player's score when in the GOING state.

x. R8B. the brightness setting and the speed in the chooseSettings state.

- **State Chart**



- **State-chart to requirements traceability table**

State / Transition Requirement	S1: chooseSettings	S2: GOING	S3: SCORE	S4: waitForReset	S5: UPDATE	T1	T2	T3	T4	T5
R1.	**X**	**X**	**X**	**X**	**X**					
R2	**X**									
R3						**X**				
R4		**X**						**X**		
R5			**X**				**X**			
R6				**X**					**X**	
R7										**X**
R8A		**X**								
R8B	**X**									

Table 1 - state-chart to code traceability table

State/Transition	Implementing code
S Update	line 186
S chooseSettings	line 269
S Going	line 312
S Score	line 336
S waitForReset	line 369
T Going -> update	line 180
T chooseSetting -> Going	line 273
T Going -> Score	line 336

T Going -> waitForReset	line 364
T waitForReset -> chooseSettings	line 372

- **Scheduling = Prioritized Preemptive**
  Note: deadlines were calculated via averaging multiple runs of the code and observing TimeInMsec.
  Note: List of tasks and their periods (1st bullet = 1st priority etc.)
- Task0
  - Scheduler
  - Period = 50ms - based on Lab 9/ works perfectly after experimentation
- Task1
  - Updates bits to be displayed on LED
  - 67ms <Period<1000ms - most people have reaction times in this range also Task2 will be able to update the time accurately
  - Deadline = Period1
- Task2
  - Display Time
  - Period = 1000ms - good to update time at this interval because want give players a good idea of how much time they have left
- Task3
  - Bonus Flag Setter
  - Period = 10000ms - if too short people won't get nervous when its about to come... the point is to test how players perform when it counts
- Task4
  - Game Loop
  - Period = 900000 - essentially supposed to constantly run (assume player will quit after many hours of play) Making this a task with a period gets rid of complex code.
- **Watchdog use**
  i. Time out period = 2^24/4e6 = 4.194s
  ii. Leaves room for adjusting reaction time for a particularly slow player (currently reaction time is max 1s)
  iii. When the COP resets the program it is not handled and proceeds to reset the board.
- **Worst case timing analysis explained**

    i. worst case execution → two break points (analyze specific situation for each task)
    ii. worst case latency → make task output a “spike” signal upon starting → observe spikes over a long period of time until “cloud” is formed → measure time from spike to “end of cloud”
    iii. Note: maximum latency for each of these (aside from Task4 which has no “latency” since it is only run once) is 0 to the nearest ms. This is likely a result of the fact that the execution time of these tasks is negligible compared to their periods.

- Task0
    - Worst Case Execution Time: 77us
- Task1
    - Worst Case Execution Time: 7us
- Task2
    - Worst Case Execution Time: 8.47ms
- Task3
    - Worst Case Execution Time: 1.125us
- Task4
    - Worst Case Execution Time: inf (game task never terminates)
- TaskW
    - Worst Case Execution Time: 3.25us

White Box Tests

	Pass/Fail	Initial State	Input 1 (State after input)	Step 2 (State after input)	Step 3 (State after input)	Step 3 (State after input)
Test 1.1		1	Button8 (2)	Button1 *when there is an LED* (3)	*Wait scrollPeriod seconds* (5)	*Wait 2min* (4)
Test 1.1 (Continued)			Step 4 (State after input) Switch 1 "bit1" (1)			

White Box Test Traceability Table

	T1	T2	T3	T4	T5
Test 1.1	X	X	X	X	X

Black Box Tests

	Pass /Fail	Test Description	Result
Test 2.1		Verify chooseSettings, GOING, UPDATE, SCORE, waitForReset by observing behavior of program after inputting appropriate arguments to make transitions.	
Test 2.2		Verify user is able to set speed and brightness in the chooseSettingState.	
Test 2.3		Verify that after user indicates he/she is finished choosing settings the game starts.	
Test 2.4		Verify the scroll a pattern of LED's at the user's chosen speed and brightness by	

		testing multiple settings of speed and brightness and comparing these respectively.	
Test 2.5		Verify the player is rewarded for reacting correctly to an LED by observing how score changes after correct button presses.	
Test 2.6		Wait two minutes to see if the score of the last game is displayed for the player's convenience.	
Test 2.7		After the user indicates he/she is done viewing results verify the user is able to re-choose the settings and start a new game.	
Test 2.8		Verify time elapsed and the player's score are displayed when in the GOING state.	
Test 2.9		Verify the brightness setting and the speed setting are displayed in the chooseSettings state.	

Black Box Test Traceability Table

	R1	R2	R3	R4	R5	R6	R7	R8A	R8B
T3.1	x								
T3.2		x							
T3.3			x						
T3.4				x					
T3.5					x				
T3.6						x			

T3.7							x		
T3.8								x	
T3.9									x

Code Review Bug List.
Developer: Thomas Mayo-Smith
Reviewer: Yiru Yao
Filename: main
Date: 5/4/15
Length: 40min
Metrics: 1165 (lines reviewed/hr) 6 (defects found/hr)

Defect #	Line #	Description of Defect
1	188	data should be updated with more data (or data copy once user has gone through all of it)
2	192	timeIntervalNotUp needs to be set to FALSE otherwise game will not end (inf loop…)
3	410	clearStrBuff was “clearing” memory indexes beyond the len of strBuff
4	809	attempt to recognize null byte terminating the will not work

Developer: Yiru Yao
Reviewer: Thomas Mayo-Smith
Filename: main
Date: 5/5/15
Length: 1hr
Metrics: 825 (lines reviewed/hour) 6 (defects found/hr)

Defect #	Line #	Description of Defect
1	285	watch dog Alive(0x10) will not work (must use kick())
2	365	same issue…
3	370	setting timeIntervalNotUp to true here has potential to cause inf

		loop
4	338	need to set scored to true inside if statement so there user has longer than one iteration of the while loop to react to an LED
5	458	need to keep interrupts disabled in timeNow() function
6	439	need to run task4() (task which updates state machine) first before any other and must do this manually in timer interrupt

White Box Test Results.

	Pass /Fail	Initial State	Input 1 (State after input)	Step 2 (State after input)	Step 3 (State after input)	Step 3 (State after input)
Test 1.1	**Pass !!!!!!**	1	Button8 (2)	Button1 *when there is an LED* (3)	*Wait scrollPeriod seconds* (5)	*Wait 2min* (4)
Test 1.1 (Continued)			Step 4 (State after input) Switch 1 "bit1" (1)			

Bug List

None!

Black Box Test Results

	Pass /Fail	Test Description	Result
Test 2.1	Pass	Verify chooseSettings, GOING, UPDATE, SCORE, waitForReset by observing behavior of program after inputting appropriate arguments to make transitions.	5 separate states did seem to be implemented.

Test 2.2	Pass	Verify user is able to set speed and brightness in the chooseSettingState.	Potentiometer set speed switches set brightness in chooseSettinsState
Test 2.3	Pass	Verify that after user indicates he/she is finished choosing settings the game starts.	Game started immediately after Button8 or “start” was pressed
Test 2.4	Fail	Verify the scroll a pattern of LED’s at the user’s chosen speed and brightness by testing multiple settings of speed and brightness and comparing these respectively.	Watch Dog interrupted and reset the board after 4 seconds of observing the scrolling.
Test 2.5	Pass	Verify the player is rewarded for reacting correctly to an LED by observing how score changes after correct button presses.	Button presses at the right time did cause an increase in score while those at the wrong time had no effect.
Test 2.6	Pass	Wait two minutes to see if the score of the last game is displayed for the player’s convenience.	After a 2min wait upon pressing Button8 in the the score was displayed.
Test 2.7	Pass	After the user indicates he/she is done viewing results verify the user is able to re-choose the settings and start a new game.	Once the final score was being displayed a flick of the Switch1 brought the player back into a state where settings for the next game could be chosen.
Test 2.8	Fail	Verify time elapsed and the player’s score are displayed when in the GOING state.	Score not initially displayed in GOING state. Instead PWM info was displayed.

Test 2.9	Pass	Verify the brightness setting and the speed setting are displayed in the chooseSettings state.	These two variables are displayed.

Bug List

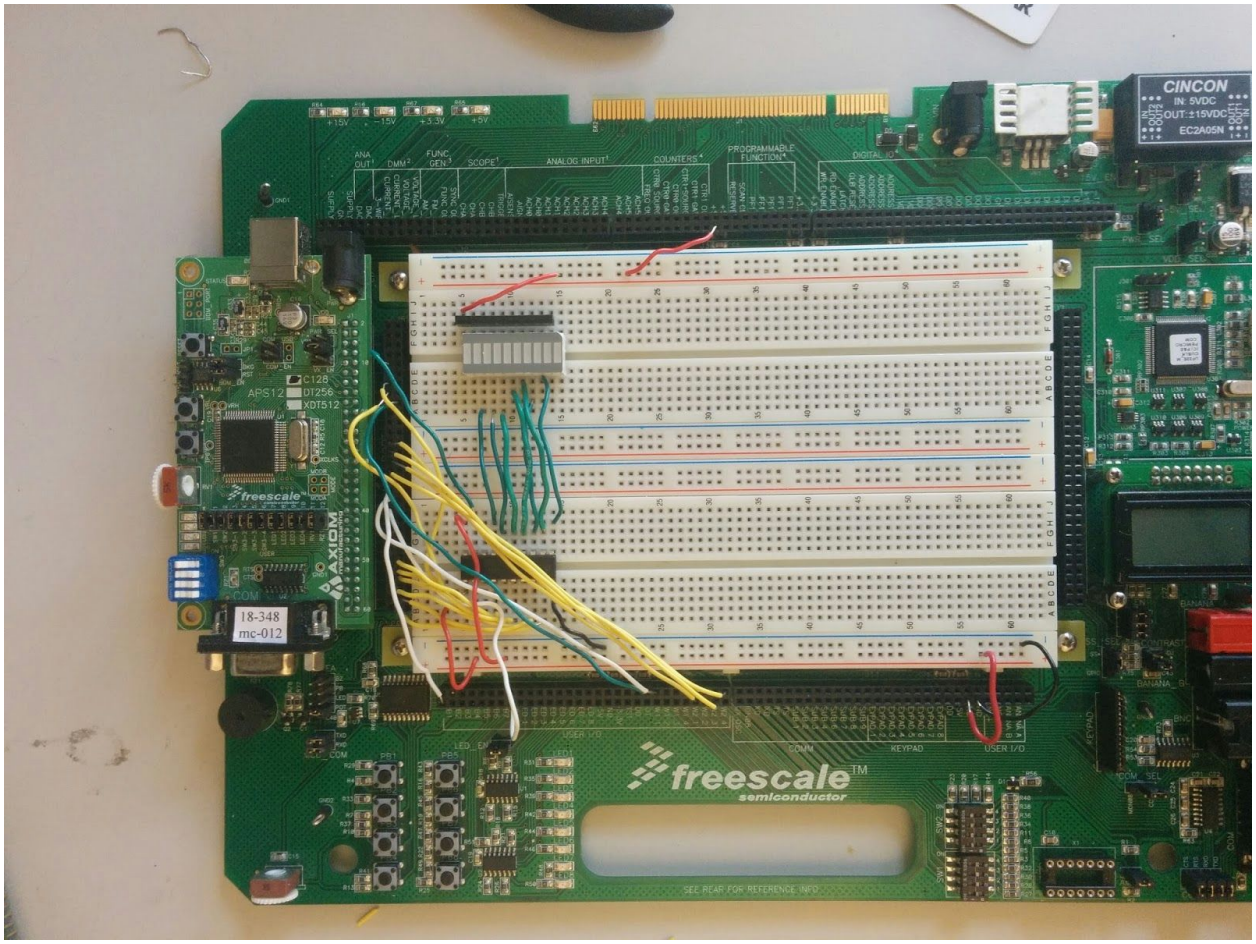
Bug #	Test #	Requirements(s)	Description
#1	2.4	R4	Bonus task with a period of 10sec did not call “Alive” function before watchdog timed out.
#2	2.8	R8A	Before the player scores for the 1st time the score is not displayed. Instead the remanence of the cooseSettings state display of PWM info is on the LCD.

- **Acceptance test plan**
  i. Send a series of 1s and 0s with 9 at the end via PC terminal and see if the LCD changes from “Loading” to speed and PWM.
  ii. Adjust the potentiometer on the project board and see if the “sp: 0xXX” changed and if the LED scrolls at a different speed.
  iii. Change the switch 2 and see if the “PWM: 0xXX” is changing on the LCD and if the brightness of LED changes.
  iv. Press button 8 and see if the LCD starts to display current time and score and if the LED starts to scroll according to the data sent from the computer.
  v. Press buttons 1 at the right timing and see if the scores is updated on the LCD.
  vi. Time the game for 2 mins and see if it stops (the LEDs stop scrolling, and the LCD displays “Time out”, score stops to update.

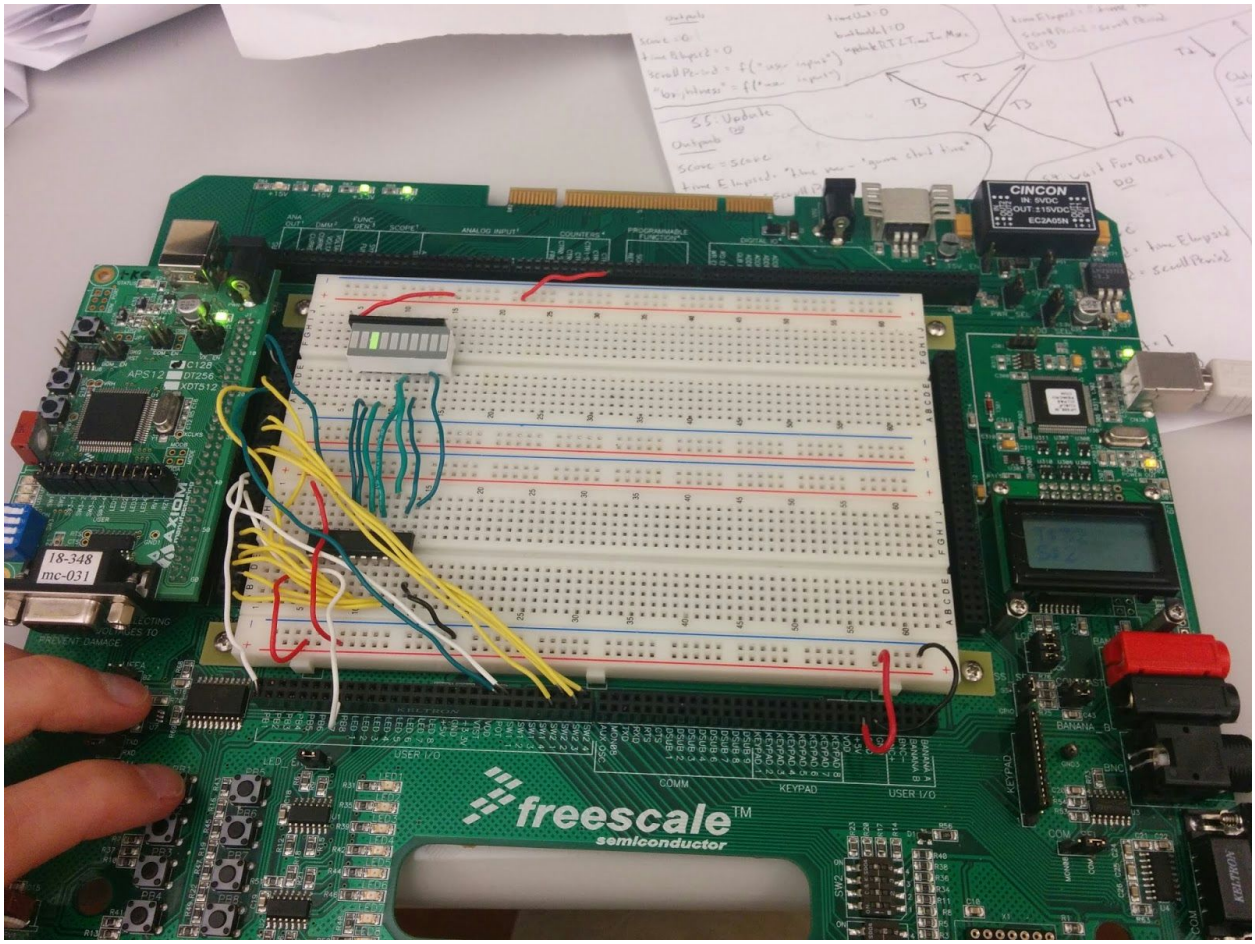
Change Log

Change #	Section	Description
1	Short Description	Change the game to use only one LEDs and take only one user input at a time.
2	List of inputs	Deleted some extra inputs that are not implemented, added the brightness input.
3	List of outputs	Changed time left to time elapsed.
4	List of state variables	Changed “match” to buttonVal since we only have one user input each time and it indicates either score or not score. Added PCB[1].nexttime to accommodate the new state “Update”.
5	List of high level requirements	Deleted the pause and reset functionality; changed the brightness from gradually changing to a setting that user can set before the game starts.
6	State-chart	Added two states: Update and chooseSettings, deleted the Pause state. Categorized the state variables to outputs and internal state variables.
7	Hardware components list	Used 245 instead of 138. Deleted extra LEDs.
8	Hardware block diagram	Newly added.
9	Acceptance test plan	Changed queries to specific series of 0s and 1s that end with a 9 in serial communication; changed to start the game with a start_button. Changed the brightness in scrolling LEDs to be the same. Deleted the pause test.
10	Scheduling	Doing serial communication through polling instead of a task. Putting score as part of the game loop instead of a task. Added
11	Watchog Use	Newly added.
12	White box test	Deleted and condensed the white box test to just one flow.

Hardware Setup



“Action Shot"


CINCON
freescale
semiconductor
18-348
mc-031
KEYPAD
COMM
USER I/O