Middle East Technical University | Department of Computer Engineering

CENG 336

Intro. to Embedded Systems Development

Spring 2022-2023

THE3

****

ULTIMATE FRISBEE

****

Due date: 26 May 2023, Friday, 23:55

Note: *This PDF uses animated images and is best viewed on PDF viewers that support javascript. On linux you can use Okular and on Windows you can use Adobe Acrobat. If your PDF viewer doesn't support javascript, animated images are also provided as .gif files.*

1 Objective

THE3 aims to help you get familiar with LCD and A/D Converter (ADC) concepts on Picsimlab with PIC18F4620.

2 Problem Definition

In this THE, you are going to make a miniature simulation of ultimate frisbee match. There will be 2 teams where each consists of 2 players. The aim for each player is either to throw the frisbee to its team mate, or to catch the frisbee thrown by any player. In this way, they will either increase the score of their own team or prevent the other team to increase their scores. You are going to simulate the players and frisbee movements on LCD, show the scores on 7-segment display, give some necessary commands from the push buttons and gamepad and adjust the speed of the game with ADC.

3 General Flow of the Game

Below, the initiation and the general flow of the game is explained. The detailed specifications are later explained one-by-one.

1. You are going to use 16x4 LCD on Picsimlab PICGenios board.

2. You are going to define the following 8 custom characters given in Figure 1 on LCD. The corresponding byte array for each character is supplied to you in the the3.h file.

3. The game starts with the initial configuration illustrated in Figure 2. The $< x, y >$ coordinates for the initial position of each character are given as follows:

   - The first Team-A player (indicated by the cursor) : $< 3, 2 >$
   - The second Team-A player : $< 3, 3 >$
   - The first Team-B player : $< 14, 2 >$
   - The second Team-B player : $< 14, 3 >$
   - The frisbee : $< 9, 2 >$

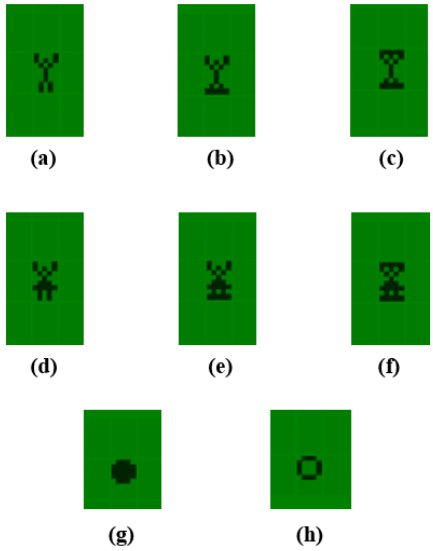

Figure 1: The figure illustrates a Team-A player (a), a Team-A player indicated by the cursor, which means to be managed by the gamepad (b) and a Team-A player holding the frisbee and indicated by the cursor (c). Similarly, you can find their Team-B correspondences in (d), (e) and (f), respectively. The frisbee itself is given in (g) and the frisbee target which is to be blinked on the target cell of the frisbee is shown in (h). The binary encodings of these custom characters are supplied to you in the3.h.

4. The game is activated with a frisbee throw. In order to throw the frisbee, there must be a player on the same cell where the frisbee is located. Initially, since the frisbee is located on the cell $< 9, 2 >$, you need to move one of the players to the cell $< 9, 2 >$. You can switch between the players by pushing the `RB1` button and choose any player that you would like to manage. Then you can move your player to the left, right, up and down by using the gamepad buttons. When you come onto the frisbee cell, you can throw the frisbee by pushing the `RB0` button.

5. Once the frisbee is thrown by pushing the `RB0` button, a random target cell is defined for the frisbee to fall. Also, movements of the frisbee is computed cell-by-cell from the starting position to the target position. A basic algorithm computing the route of the frisbee is supplied to you in the3.h. You can directly use it as well as you are allowed to do any change on it. When the route computation

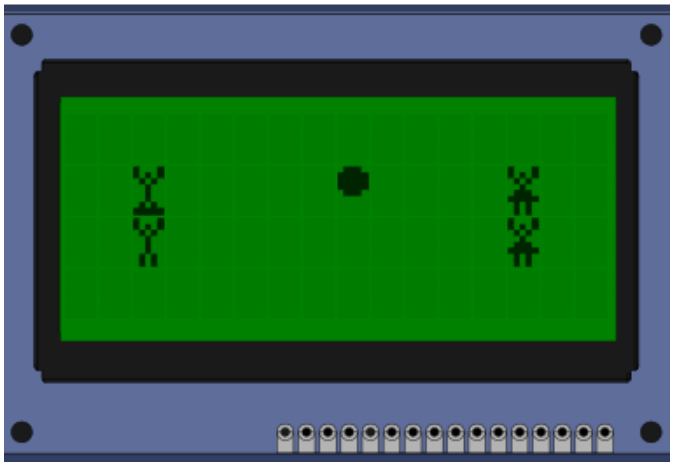

Figure 2: The initial configuration of the LCD when the game starts is illustrated. The corresponding position coordinates for each character is given in the text.

is completed, the frisbee starts to move based on its computed route at regular time intervals. At each time period, it applies one movement. This movement consists of one of the followings: 1-cell left, 1-cell right, 1-cell up, 1-cell down, one of the four crossing cells.

6. While the frisbee is moving, you can switch between the players and also move your current player by using the gamepad. At that moment, the other players that you are not managing will be moving randomly. You can program the movement of those players in a synchronous way with the movement of the frisbee. In other words, one movement of the frisbee corresponds to one random movement of the other players. On the other hand, the management to move your current player and switching between the players does not have to be synchronized with the frisbee. They should immediately react to the commands given by gamepad and `RB1` button, respectively.

7. When the frisbee completes its route and reaches to the target cell, you should check whether it has been caught by your player. If it is, then you should increase the score of the team that player belongs to by one. Otherwise, it means the frisbee fell onto the floor and scores do not change. By the way, in a real frisbee game, there occurs a score only if the frisbee is caught in the score parts locating at the ends of the game area. Yet, we adapt the scoring in the explained way to easily observe the changes on the 7-segment display.

8. When the frisbee movements are completed, you can rethrow the frisbee by following the procedure from step-4. The game continues in this way.

9. You can change speed of the game simulated on LCD over the ADC.

10. The duration between the frisbee throw and fall/catch is called as `ACTIVE_MODE` whereas the duration in which the frisbee does not move is called as `INACTIVE_MODE`.

11. A sample animation showing the flow of the game is given in the Figure 3.

4 Implementation

4.1 Gamepad Specs

- Figure 4 shows an image of the gamepad.

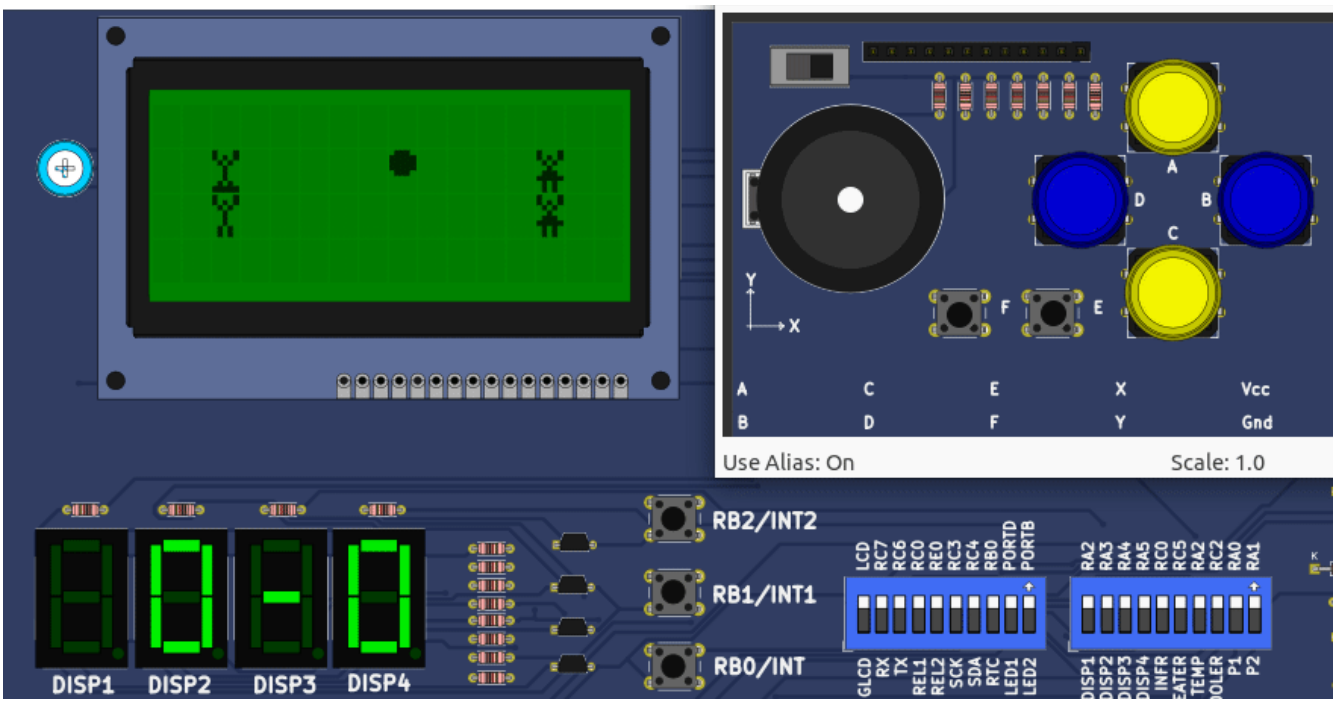

Figure 3: A sample scenario is illustrated. Note that the movements of the player indicated by the cursor is managed by gamepad. Also, depending on which user catches the frisbee, the score table changes. There does not exist any flickering or discontinuity on LCD and 7-segment display. Actually, it is very important to satisfy this feature in terms of grading. Also, take attention that the frisbee target character is blinked until the frisbee reaches to the target cell. Lastly, there occurs some player switches at some points of the time. Some of them are caused because `RB1` button is pushed whereas some of them are caused because the turn for throwing the frisbee passes to the other team.

- The gamepad is used in order to move the selected player.
- It should operate based on PORTB On-Change interrupt mechanism.
- You should connect the up button, right button, down button and left button to RB4, RB5, RB6 and RB7 pins, respectively. The corresponding pin numbers for those pins are 37, 38, 39 and 40.. The given .pcf file is adjusted in this way too.
- You can also operate the buttons by pushing I, L, K and J keys on the keyboard. There is nothing special that you need to do for activating this feature. It is default. Note that there may appear some differences depending on the keyboard language. For instance, since the correspondence of the capital of the "i" letter is "I" in English, you may need to push either on "i" key on the keyboard or on "I" key while the Caps Lock is turned on. The keyboard usage may be practical, yet if you have a problem because of the explained language features, you can always use mouse to push the buttons on gamepad.
- It doesn't matter at which edge of the pushing operation the interrupt occurs (rising or falling). You should accept the interrupts either when the button is pressed or released.
- Pushing any one of the buttons results in the currently selected player to move one single cell in the corresponding direction. If the player is in an end cell of that direction, then the movement is invalid (that is the button push results in no operation). For instance, if the player is on the cell $< 16, 2 >$ of the 16x4 LCD, then pushing the right button does not affect the position of the player.

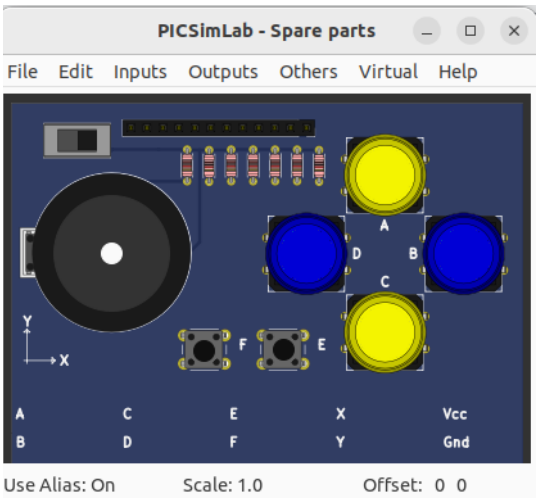

Figure 4: The gamepad to manage movements of the selected player is given. The buttons tagged by A, B, C and D are connected to `RB4`, `RB5`, `RB6` and `RB7` pins and used to go 1-cell up, right, down and left, respectively. They operate based on PORTB On-Change interrupt.

- Gamepad is always active both in `ACTIVE_MODE` and `INACTIVE_MODE` of the game whether or not the player holds the frisbee.
- Whenever a valid command is given by gamepad, you should immediately reflect the corresponding movement on the LCD.

4.2 `RB0` Button Specs

- This button is used to throw the frisbee.
- It should operate based on the INT0 external interrupt mechanism.
- It doesn't matter at which edge of the pushing operation the interrupt occurs (rising or falling). You should accept the interrupts either when the button is pressed or released.
- This button operates only if there is a player on the same cell that the frisbee locates and the game is in `INACTIVE_MODE`. In other cases, pushing this button does not mean anything (No operation).
- When the button is pushed in a valid mode, the game is switched to `ACTIVE_MODE`. You should initiate the computation of frisbee target and movements. Similarly, you should activate the random movement computations for the players who are not managed by gamepad.

4.3 `RB1` Button Specs

- This button is used to switch between the players to choose the player that you want to manage by gamepad.
- It should operate based on the INT1 external interrupt mechanism.
- It doesn't matter at which edge of the pushing operation the interrupt occurs (rising or falling). You should accept the interrupts either when the button is pressed or released.

- If the game is in `INACTIVE_MODE` and the currently selected player already holds the frisbee, the player switching should not operate. In other words, pushing this button does not mean anything (No operation). This is because there is no character defined to indicate a player holding the frisbee yet not managed by you (gamepad).
- As long as the frisbee is not hold, you can switch between any one of the four players by pushing this button. The switching order does not matter, yet it should loop through all the four players one by one at each push and return to the beginning of the loop at each 4th push.
- A sample switch player action is illustrated in Figure 5.

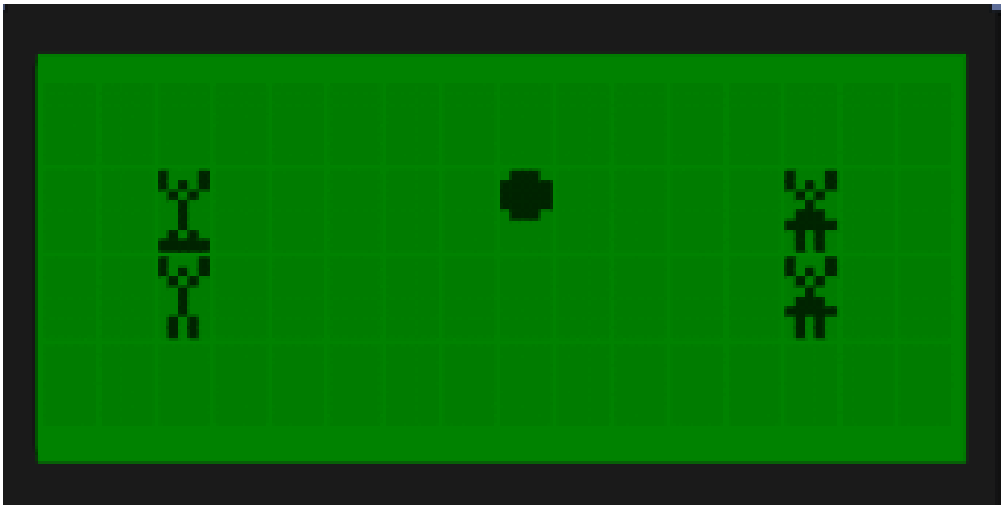

Figure 5: Switching between the players at the beginning of the game is illustrated.

4.4 Timer Interrupt Specs

You should use timer interrupts for various activities explained below:

- You should set the clock frequency to 10 MHz.
- After a frisbee throw, there is defined a random target cell for the frisbee to reach. In order to inform the user about which cell is the target, you should blink the "frisbee target" character on the target cell. The blinking starts when the target is defined and continues until the frisbee reaches to the target cell. For the blinking operation you can use timer interrupt. Showing and clearing the "frisbee target" character at intervals of approximately 100 ms gives a valid blinking.
- After a frisbee throw, the frisbee route is computed for the randomly defined target cell. Next each step of the route is simulated on the LCD with some regular time intervals. In order to simulate with the exact time intervals, you should use timer interrupts. The default period for the movement simulation is 400 ms. However, we will also use ADC to change the speed of the simulation. Depending on the value that ADC dial is set to, the simulation could be slowed down either to the periods of 800 ms, 1200 ms or 1600 ms. Note that you should also compute random movements of the other players within those intervals.
- The given timings above are not highly strict, yet note that they are determined within a sufficient amount sensibility that can be measured by human eye.

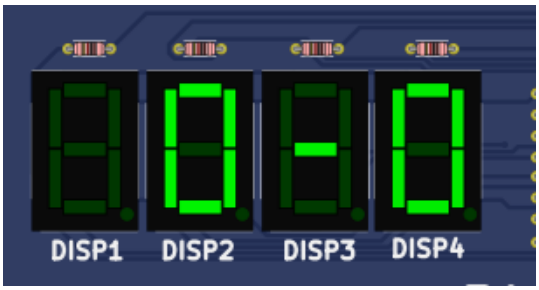

Figure 6: The score table is given. DISP2 shows the score of Team-A whereas DISP-4 shows the score of Team-B.

- You can also use timer interrupts for the other operations depending on your need. However, it is not mandatory to use timers for those operations and there does not exist a definite time period defined by the author. You should configure them by yourself to provide conformal results with the explained specs.

4.5 ADC Specs

You should use ADC interrupt to adjust the speed of the simulation.

- You should divide the numerical scale (10-bits) defined by the ADC dial into 4 equal intervals. Each interval corresponds to a different period for the random movements (those of the frisbee and players (except the one managed by you)). The scale of 1024 should be separated into the following intervals [0, 255], [256, 511], [512, 767] and [768, 1023] and they should correspond to the periods of 400 ms, 800 ms, 1200 ms and 1600 ms for the simulation, respectively.
- Any change by the ADC dial should be reflected on the next movement of the currently simulated `ACTIVE_MODE`.

4.6 7-Segment Display Specs

While you simulate everything on the LCD, you are expected to simultaneously show the scores on the 7-Segment Display.

- The score table on the 7-segment display is illustrated in Figure 6.
- When a player catches the thrown frisbee, the score made by the corresponding team increases by 1. This should be immediately reflected on the 7 segment display.
- We will not test the game upto there occurs high scores. Therefore, you can assume the maximum score will be 5 while implementing.
- You should use DISP2, DISP3 and DISP4 only. Use the default pin connections which are RA3, RA4 and RA5 for DISP2, DISP3 and DISP4 respectively. DISP2 and DISP4 will be used to show the score of Team-A and Team-B, respectively and DISP3 will be used to show a dash character ("-").
- There should not be any flicker on the 7-segment display. Also, you need to be very careful while using it simultaneously with the LCD. Since they use the same data port which is PORTD. There should not be any discontinuity neither on the LCD nor on the 7-segment display.

4.7 Miscellaneous Specs

Below, there are specifications that are given to make the game consistent. However, you do not have to implement them as exactly as they are stated.

- A player should not move onto the cell including another player.
- It is possible for the frisbee to be caught only by the player that you manage since we do not have an LCD character defining the other players while holding the frisbee (We only defined our controlled player character on LCD to hold the frisbee). Therefore, the random movements of the other players should be designated not to conflict with the frisbee. In other words, they should never be in the same cell with the frisbee.
- Neither frisbee nor players can move out of the 16x4 LCD. If the movement operation (caused by gamepad or random generator) requires an object locating on a border cell to go out of the LCD, this operation is simply neglected (result is "no operation").
- You have to implement a random algorithm for the movements of the players that are not managed by the gamepad. You can implement the following recommended algorithm for it: Design the random movement of the players to go to 1-cell neighbor. For instance, you can define 9 different movements such that if the random generator produces 0, the player moves up; if it is 1, the player moves right, for 2 move down, for 3 move left, for 4 move up-right crossing cell, for 5 down-right crossing cell, for 6 down-left crossing cell, for 7 up-left crossing cell and for 8 no move. Of course, the movements occur only if it is possible.
- When a player stands on the cell of frisbee target, you can simply show any one the two objects (either frisbee target or the player).
- A player should not move while holding the frisbee. Therefore, if you give move command to another cell while you are on the same cell with frisbee, you leave the frisbee on the current cell and move the player alone.
- If the frisbee was thrown by a member of Team-A (or Team-B), and it is not caught by anyone, then right to throw the frisbee belongs to the Team-B (or Team-A respectively).

4.8 Algorithm for Pseudo Random Generator

An algorithm defining the movements of the frisbee is already supplied to you in the3.h file. However, both the given algorithm and the one generating the movements of the players who are not managed by the gamepad call a random generator. You should implement your own pseudo random generator based on the algorithm given below:

1. Use one of the timers in 16 bit mode. Enable it at the beginning of the game. You do not need to use any interrupt of it.
2. When you need a randomly generated value, read the number in the data registers of timer and apply the necessary modulo operation on that number to obtain a valid value for your need. You can directly use the resulting value as the random number that you are looking for.
3. For the future random values, configure the timer by rotating the value which was read before the modulo operation and putting on the timer data registers. In other words, continue the timer increments over the rotated value. You can use different rotation amounts from 2-bit to 15-bit for different random number needs. Choose your own shifting amounts.

4. Example over an 8-bit timer value: Assume the value read on the timer is 10011010 (which is 154). If you need a random number between 1 and 4, Take the modulo 4 of 154 (which is 2) and add 1 to make the value on the interval [1,4]. The resulting value becomes 2+1=3. You can choose 3 as the needed random value. For the next random values, re-configure the timer. For instance you can choose the rotating amount as 2-bits and rotating direction as the right. Then the rotation applies as 10011010 –> 01001101 –> 10100110. You can fill the timer data register with 10100110 and let it continue its increment.

5 Hints

- If it is not possible to use buttons and gamepad appropriately due to bouncing effect, you can apply the following: Take only the first signal into consideration and wait some amount of time for the bouncing to disappear.
- In order to obtain a continuous and not-flickering display on LCD and 7-segment display, you can call your function that activates the seven segment display as frequently as possible and shorten the LCD delays sufficiently.

6 Regulations

- You must code your program in C and compile with MPLAB XC8 C compiler.
- Your program should be written for PIC18F4620 working at 10MHz.
- There is already a discussion forum for THE3, please ask your questions there. Unless you have really specific question about your code use the forum! If you think that your question is too specific to ask on the forum or private you can ask your questions via e-mail to `asiler@ceng.metu.edu.tr`.
- You should submit your files by compressing as group-number.tar.gz. **Please submit everything including LCD files, the3.h etc. even if you did not change them.**
- You have a total of 3 days for late submission. A penalty of $10 * day$ is applied.
- In case of cheating, the university regulations will be applied.

7 Grading

- Showing initial game configuration on LCD ..... 5 points
- Blinking the frisbee target on LCD within regular time intervals ..... 10 points
- Randomness Algo. (show random movements of frisbee and players on LCD smoothly) ... 10 points
- `RB0` button usage (throwing the frisbee)) ..... 5 points
- `RB1` button usage in `INACTIVE` + `ACTIVE MODE` (show the effects on LCD smoothly) .... 5 + 10 points
- Gamepad usage in `INACTIVE` + `ACTIVE MODE` (show the effects on LCD smoothly) ..... 5 + 10 points
- 7-Seg. Disp. in `INACTIVE` + `ACTIVE MODE` (show the score without any flicker) ..... 10 + 10 points
- ADC Operation `INACTIVE` + `ACTIVE MODE` (show the speed change on LCD) ..... 10 + 10 points

NOTE: Any "flickering", "discontinuity", "irregular intervals of timing", **etc.** issues will result in 5-10 points of penalty from the corresponding grading item.