Fall 2014

ECGR 4101/5101

Lab 7: Creating a Virtual Pet

Objective:

In this lab, we will create a virtual pet using the LCD and various other peripherals on the board. For those of you who remember the virtual pet toys from the 1990's, we will be creating something very similar. Using everything that we have learned throughout the labs we have completed this semester.

Some helpful labs to reference for this lab are:

Lab 3 – "Top Gun"

Lab 6 – "Oscilloscope"

Assignment:

1. Begin by creating the tutorial project workspace under the YRDKRX63N Simple Demos option. Consult the supplemental instruction from lab 2 if you need help setting up the workspace.
2. We will be following a very similar process to the previous for creating functions to manipulate the LCD graphics. If you wish, you may use the previous lab and simply edit the main file. If you do, you may skip to step 5.
3. Once we have created a tutorial workspace, we will begin by removing most of the functions in tutorial_main.c.
    a. Remove flash_led(); from main()
    b. Remove timer_adc() from main()
    c. Remove statics_test() from main()
4. We are going to need to write custom graphics to the screen, so we need to prepare functions that allow us to set individual pixels on the LCD. We will do this by setting the LCD front to bitmap and writing one large, square character to the entire screen and setting the individual bits within the character array.
    a. Begin by adding bitmap_font.c to your workspace. When you create the tutorial workspace, this file is created for you, but not included in the project. It is located in "your workspace root dirctory"/r_glyph/src/glyph/fonts/.
    b. Then, you will need to open config.h and uncomment the following lines of code

```
//#define USE_GLYPH_FONT_BITMAP
//#define USE_DEFAULT_FONT Bitmaps_table
```

If you did this correctly your config.h should look like this

```
#define USE_GLYPH_FONT_BITMAP
//#define USE_GLYPH_FONT_HELVR10
//#define USE_GLYPH_FONT_8_BY_16
#define USE_GLYPH_FONT_8_BY_8
//#define USE_GLYPH_FONT_WINFREE
//#define USE_GLYPH_FONT_5_BY_7
//#define USE_GLYPH_FONT_6_BY_13

#define USE_DEFAULT_FONT Bitmaps_table
//#define USE_DEFAULT_FONT FontHelvr10_table
//#define USE_DEFAULT_FONT Font8x16_table
#define USE_DEFAULT_FONT Font8x8_table
//#define USE_DEFAULT_FONT FontWinFreeSystem14x16_table
//#define USE_DEFAULT_FONT Fontx5x7_table
//#define USE_DEFAULT_FONT Fontx6x13_table
//#define USE_DEFAULT_FONT FontHelvr10_table
```

c. You will also need to edit glyph.h by uncommenting and adding “extern”:

```
extern const uint8_t * Bitmaps_table[256];
```

d. At this point we are ready to start modifying the LCD code. Open up lcd.c and in the function InitialiseLCD add

```
GlyphWrite(Lcd_handle, GLYPH_FRAME_RATE, 137);
GlyphWrite(Lcd_handle, GLYPH_CONTRAST, 255);
```

Inside the if statement before the

```
GlyphNormalScreen(lcd_handle)
```

Command.

e. In lcd.c create a new function called Set_Font_8_by_8() that has the following function call inside

```
GlyphSetFont(Lcd_handle, GLYPH_FONT_8_BY_8)
```

Note: be sure to add a function prototype in lcd.h for Set_Font_8_by_8()

f. In lcd.c create a new function called Set_Font_Bitmap() that has the following function call inside

```
GlyphSetFont(Lcd_handle, GLYPH_FONT_BITMAP)
```

Note: be sure to add a function prototype in lcd.h for Set_Font_Bitmap ()

g. In lcd.c create a new function called Set_LCD_Pos(int x, int y) that has the following function calls inside

```
GlyphSetX(Lcd_handle,x)
GlyphSetY(Lcd_handle,y)
```

Note be sure to add a function prototype in lcd.h for Set_LCD_Pos(int x, int y)

5. Once the lcd.h file is prepared, overwrite your bitmap_font.c file with the one provided on the course website. This file contains custom images for our virtual pet.
6. Now that we have functions and graphics for our screen, we need to add a timer for refreshing the screen and evaluating game events. In the tutorial project there is a function called timer_adc(). This function reads analog channel 2 (the potentiometer) and uses that value to set a timer period which calls a callback function called cmt_callback_readADC(). Add a global variable to timer_adc.c called draw and initialize it to 0. Inside of cmt_callback_readADC(), set draw to 1. Now, inside of your tutorial_main.c, add extern int draw; as a global variable. We will use this draw flag to refresh the screen.
7. Inside of your main, create a function called updateScreen(). It should return nothing and have nothing passed to it. Inside of your while loop, add:

```
If(draw)
    updateScreen()
```

Now, updateScreen will be called every time the draw flag is set by the timer. However, inside of updateScreen we need to set the draw flag back to 0. We will also draw the image of our pet inside of this function as well. Inside of updateScreen include the following code for testing:

```
lcd_clear();
Set_LCD_Pos(20,20);
Set_Font_Bitmap();
// Select Animation Frame
if(animate == 1)
        animate = 2;
else
        animate = 1;
Set_LCD_Char(animate);
draw = 0;
```

You will also need to add a global int variable called animate, and initialize it to 1. If you compile your code at this point, you should see a blobby creature on your screen, bouncing up and down at the rate of timer. Turning the potentiometer should increase the rate at which the blobby creature bounces or “breathes.”

8. Now we have the base for drawing a creature to the screen. At this point I will not provide any more detailed instruction. Instead, I will only provide a requirements list with hints as to where you can find additional resources to help with the project.

Requirments:

1. During an idle state, the virtual pet's idle animation should be displayed.
2. The virtual pet should have at least 2 attributes: hunger and happiness
    a. As the game is running, the pet's hunger should increase slowly and the happiness level should decrease. (The purpose of the potentiometer adjusting the games speed is so we can decrease these attributes quickly over timer)
    b. if the pet's hunger stays at 100% for too long, the pet should die of starvation
        i. When the pet dies of starvation, the animation should be replaced by the "dead" bitmap image in the bitmaps_font.c file. The user should no longer be able to interact with the pet through the menu options.
    c. if the pet's happiness stays at 0% for too long, the pet should "run away"
        i. When the pet runs away, it should no longer appear on the screen and the user should not be able to interact with it through the menu options
    d. when the game begins, the pet should start with minimum hunger and maximum happiness
    e. HINT: use global variables to represent hunger and happiness
3. Pressing switch 1 brings up the menu.
    a. The first line of the LCD should display "MENU"
    b. The second, and third lines should be used to select different actions for the pet
    c. The fourth line should have an exit option, to return the user back to the game animation
    d. The last two lines of the screen should display the hunger level and happiness level of the pet as a percentage
    e. Add a ">" symbol in front of the line that is currently selected
    f. Pressing switch 2 should move the selection arrow up
    g. Pressing switch 3 should move the selection arrow down
    h. Press switch 3 to select the currently selected action
    i. HINT: Use a global variable as a "flag" to toggle between drawing the game graphics, and displaying the menu options
4. There should be at least 2 different actions for the user to interact with the pet.
    a. There must be a "feed" pet option. Feeding the pet should reduce the pet's hunger level.
        i. The feeding process should take some time; While the pet is feeding, the user should be able to see the feeding animation.
        ii. HINT: for feeding, set some number of timer cycles for the pet to feed. For each cycle, you can reduce the hunger variable by some amount
        iii. HINT: for the feeding animation, use "eat1" and "eat2" in the bitmap font array
    b. One other action must be available for the pet. You may choose to implement any of the following:
        ii. Walk the pet: Use the accelerometer to take the pet on a "walk." As the user tilts the board, move the location of pet on the screen. As the pet moves, its happiness level should increase, and it's hunger level should increase.
        iii. Talk to the pet: Use the on-board microphone to talk to the pet. Talking to the pet should increase the pet's happiness.
        iv. Play "Catch" with the pet: use the switches to play a game of catch with the pet. Switch 2 and Switch 3 will be used to "throw" a ball at the pet. The pet will randomly move between two

positions on the screen (top and bottom) and the switch 2 and 3 will throw a ball in the top and bottom position, respectively.

5. Creativity: For this section, you must add your own, creative aspect to the game. You may create your own action for the pet, create new graphics for the pet, or anything you want to do, no matter how ridiculous that might be.

Hints:

- For interfacing with a new sensor, check out the demo applications in hew. Chances are, there is a library already written for all the sensors on the board.
- This lab will be graded on how you manage to meet the requirements. There isn't one specific "correct" answer. There are many ways to complete this lab.
- For creating your own graphics, I have included on the lab handout page the bitmap images and the software for converting the bitmaps to the monochrome bitmap used by the LCD. However, the software does not do a perfect job of rendering the image, and you will need to make a few slight modifications by hand.

To Submit:

- A pdf containing your project report. Paragraphs describing the objective, what you did to complete that objective, and code snippets throughout the description as they are relevant. When describing a library function you created, be sure to explain any design decisions you made.
- Your tutorial_main.c
- A zip file containing the 2 files mentioned above
- Your lab check-off sheet at the demonstration

Grading will be changed such that 40% is your demonstration, 20% will be your code structure, and the remaining 40% will be your report. Your code will be graded based on comments, proper alignment (indentations), function design, proper capitalization of definitions and variables, and design of custom functions.

Embedded Systems Lab Demonstration Validation Sheet

This sheet should be modified by the student to reflect the current lab assignment being demonstrated

Lab Number:	Lab 6 – Oscilloscope
Team Members	Team Member 1 Name Here Team Member 2 Name Here
Date:	MM/DD/YYYY

Lab Requirements

Obtain a list of the Lab requirements from the end of the lab handout and type them here, perform a self-review and indicate with an X if you met each requirement or not.

REQ Number	Objective	Self-Review	TA Review
1	The pet is animated and displayed on the screen properly. Adjusting the potentiometer modifies the game's speed. As the game continues idly, the pet become more hungry and more unhappy.		
2	The pet should die or run away as its hunger remains at 100% and it's happiness remains at 0%.		
3	Pressing switch 1 displays the menu. Pressing switch 2 and 3 move the selection arrow between options. Pressing switch 1 again selects an action.		
4	The pet is able to be fed. The feeding animation is shown, and the pet's hunger variable is properly increased.		
5	A second action is available to the user. It is able to perform properly as described in the requirements.		
6	A creative aspect is applied to the game in some way outside of the requirements specified.		

Comments

If you were unable to meet a particular requirement or if a requirement required a custom implementation describe briefly why you were unable to meet the requirement or what your custom implementation was in the box below, also if you have any additional considerations that you would like to be taken into account while grading you may type them here as well.
