EE712 Embedded System Design Course Project Report, EE Dept, IIT Bombay, April 2015
(Course Instructors: Prof. P. C. Pandey & Prof. D.K. Sharma)

Archery Game using TI-RTOS

Vishwanathan Iyer (143076004) <vishwanathan@ee.iitb.ac.in>
Piyush Manavar (P10131) <p10131@iitb.ac.in>
Ganesh Gore (13307R017) <ganeshgore@iitb.ac.in>

Abstract: Interactive archery game is implemented using TI-RTOS on 128x64 graphics LCD using two-axis joystick with push button. The de-bouncing of the push button is done in software using three state mealy state machine. The algorithm for performing various tasks such as updating target, reading ADC, updating the bow and arrow position and updating the score is scheduled periodically. In the game there are five levels which the player can select at the start. In each level the movement speed of the target changes and it increases with the level. The player is given six arrows to shoot. At the end of the level the total score of the game is displayed.

1. Introduction:

The archery game is designed and developed using Tiva C Series TM4C123G LaunchPad Evaluation Kit which is a low-cost evaluation platform for ARM® Cortex™-M4F-based microcontrollers from Texas Instruments. The design of the TM4C123G LaunchPad highlights the TM4C123GH6PM microcontroller with a USB 2.0 device interface and hibernation module.The EK-TM4C123GXL also features programmable user buttons and an RGB LED for custom applications.

In this project we have used the RGB LED for indicating shooting of the arrow along with the buzzer, the 128x64 graphics LCD to the GPIO pins of the controller, the joystick is interfaced to the ADC pins of the controller. We have used TI-RTOS to implement the various tasks of the game.

TI-RTOS is a real-time operating system that enables development by providing schedulers, protocol stacks, power management frameworks and drivers. It is provided with full C source code and requires no up-front or runtime license fees. TI-RTOS is supported directly by TI. TI-RTOS scales from a low-footprint, real-time preemptive multitasking kernel to a complete RTOS with additional middleware components including a power manager, TCP/IP and USB stacks, a FAT file system and device drivers, allowing developers to focus on differentiating their application. It provides a consistent embedded software platform across TI's embedded processing portfolio, making it easy to port existing applications to the latest devices.

The TI-RTOS Kernel (formerly called SYS/BIOS™) is optimized for use in performance and footprint-sensitive applications. TI-RTOS Kernel provides preemptive multitasking, communication and synchronization primitives and memory management. In addition, it provides standard interfaces for managing select hardware resources such as interrupts, timers, exceptions and power-saving modes. OS objects, such as tasks and semaphores, may be created either dynamically by the application or statically using a configuration tool. To support hard real-time applications, such as motor control where failure to respond to an interrupt may result in a critical

system failure, TI-RTOS Kernel supports "zero-latency interrupts" so that the most critical interrupts are never disabled.

TI-RTOS Kernal services include Task, software interrupts, clock, events, mailbox, semaphore, gate, heaps, hardware interrupts, timers and timestamps and logs. In this project we are using tasks and semaphores services for implementation.

2. Hardware and Software requirements

a. Hardware required:
   i. Tiva C series TM4C123G Launchpad
   ii. Two axis joystick with push button
   iii. 128x64 Graphics LCD
   iv. Buzzer
   v. Potentiometer
   vi. Connecting wires

b. Software required:
   i. Code Composer Studio.
   ii. Tivaware for Tiva C Series
   iii. TI-RTOS for Tiva C Series

3. Hardware design:

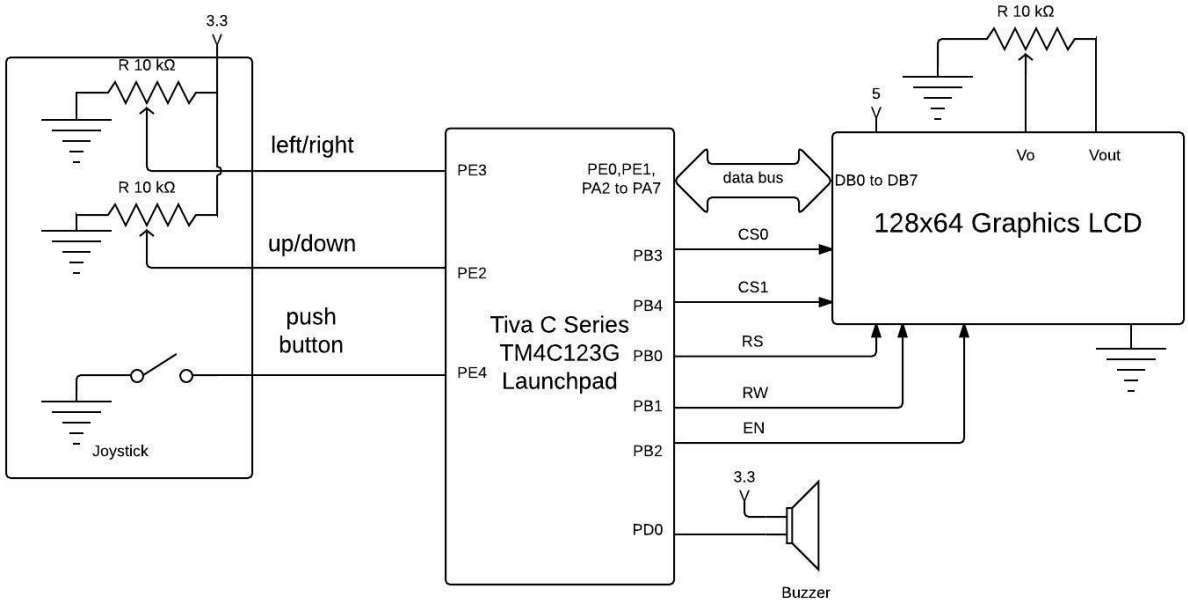

Fig. 1 Block Diagram

- The system consists one input device and two output devices.
- The joystick consists of two potentiometers and one push button. The two potentiometers are connected to PE3 and PE2 pins of the Launchpad.
- The left/right pin is connected to PE3 and up/down is connected to PE2. The switch is connected to PE4 which is configured in weak pull-up mode.
- The buzzer is connected to PD0 and it is used to give feedback to the user during the game.
- The LCD data lines are connected to PE0, PE1 and PA2 to PA7 as DB0 to DB7 respectively. CS0 is connected to PB3, CS1 is connected to PB4 and RST is connected to PB5. RS pin is connected to PB0, RW pin is connected to PB1 and EN is connected to PB2.
- The LCD works on 5V. V_{out} of the LCD is connected to potentiometer and the variable terminal is connected to V_o.

4. Software Design

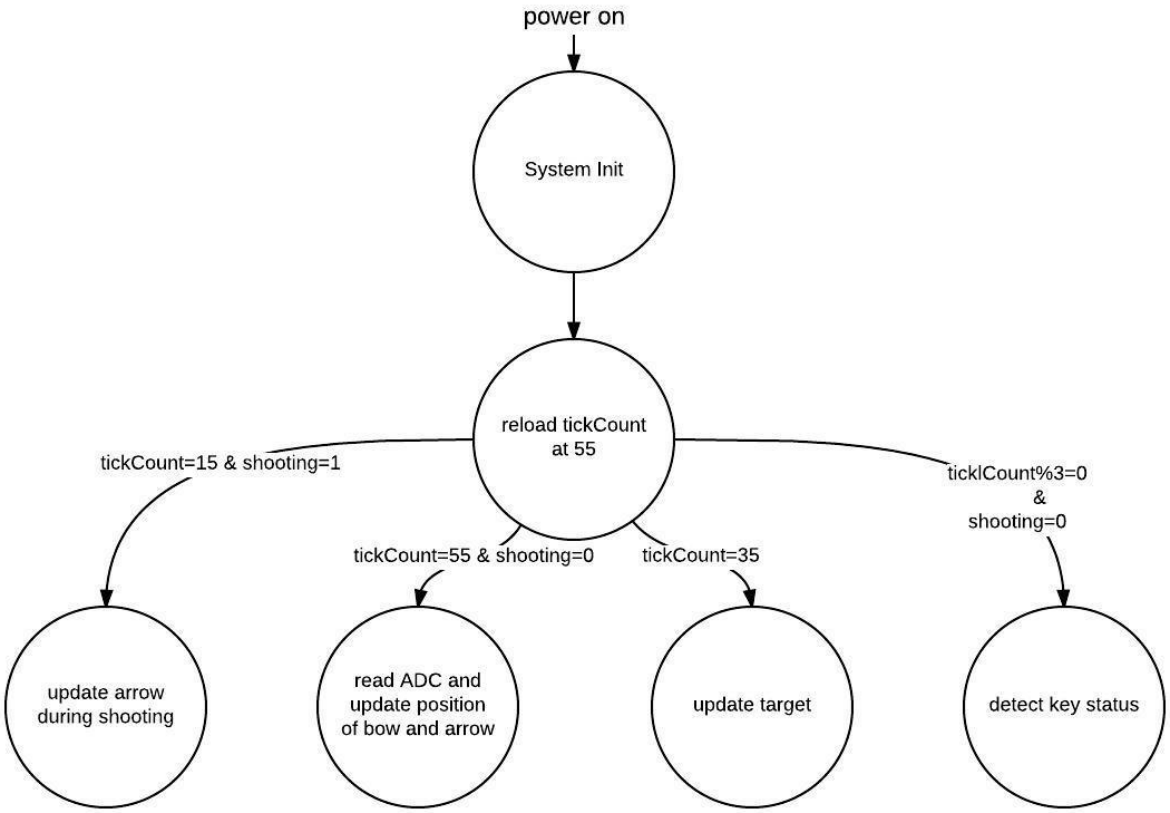

Fig. 2 System data flow diagram

- The system starts from initialization of hardware by configuring the input/output ports, ADCs and timer. Then it starts the TI-RTOS bios.
- When the bios starts all the defined tasks are created and the code which is present in their initialization gets executed.
- In our program we are saving the original priority of the tasks when they are initialized. Then all the tasks are waiting for their semaphore to be released.

- The timer is configured to a period of 3.3 ms. When the timer times out a count variable tickCount is increased by one.
- When the count reaches 55 (by this time all the tasks are executed once) the count is reloaded.
- The semaphore is released in the timer2 ISR on defined tickCount value and status of the shooting flag as shown in the above diagram.
- The shooting flag is used to indicate that the shooting of the arrow is taking place and at this time we are not releasing the semaphores of reading the ADC and detecting the shoot key press.

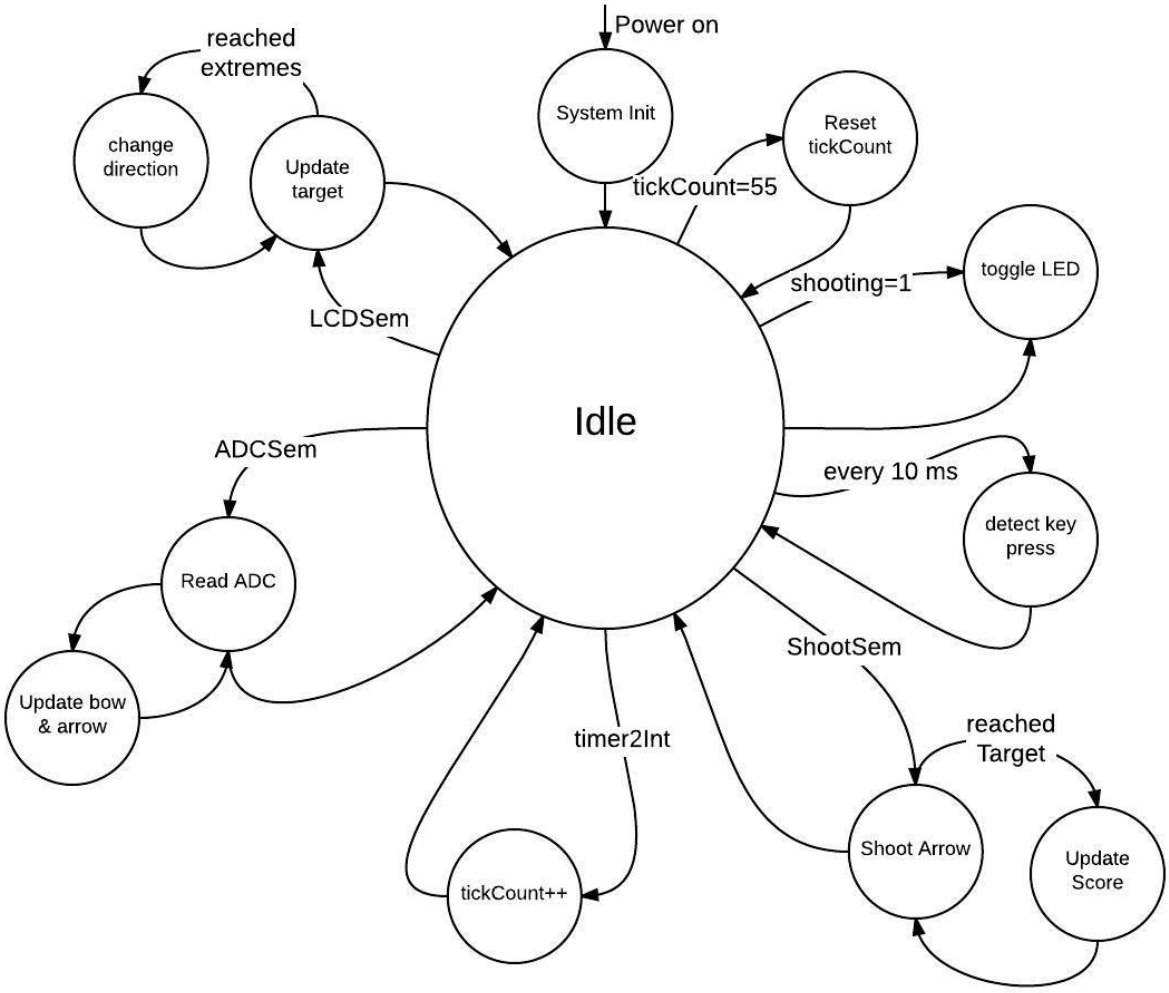

Fig. 3 Detailed flow diagram

- The system is normally in idle state. When a particular semaphore is released the tasks corresponding to that semaphore is executed.
- The ShootSem is released most frequently at every 10 ms. The state diagram to detect key press is show below.
- The LEDSem is released whenever the shooting flag is one. This tasks toggles the LED when the shooting is taking place.

- When the ADCSem is released the two potentiometers readings are taken corresponding to up/down and left/right and the corresponding position of the bow and arrow are updated.
- The user can decide the speed at which the arrow will be released by using the left/right motion of the joystick and position of the bow by using the up/down motion.
- When the LCDSem is released the position of the target is updated according to the level in which the game is being played. In this routine it is checked that if the target has reached the extremes or not and on reaching the extremes the direction of movement is changed.

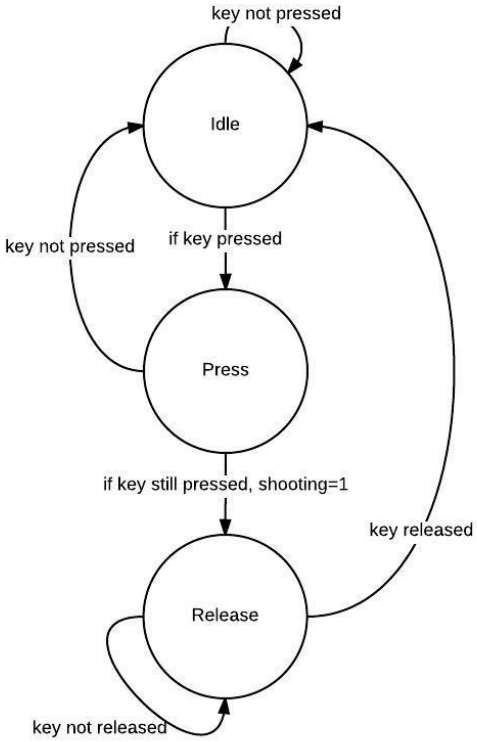

Fig. 4 State diagram for software de-bouncing of shoot button

- In order to detect the key press a state diagram is implemented as shown above.
- The state transitions take place every 10 ms.
- Each time detectKeyPress() is called it maintains its previous state and based on previous state and condition of the switch the next state is determined.
- Only if the state of the key is pressed for at least 10 ms the shooting flag is raised indicating that key press is confirmed.
- It remains in release state until the key is released. Only if the key is released it enters idle. Hence if the key is pressed multiple shoots will not be detected.
- When the system is in a particular state during the 10 ms it can perform other tasks and it does not waste time waiting in a state.

5. Game rules and scoring

1. When the system starts it shows start page with an option to select the level
    a. In this section user can select level of difficulty to play archery game form level 1 to level 5.
    b. Speed of moving target will increase with difficulty level.
    c. Selection of the level can be increased by moving joy stick up and can be decreased by moving it down.
    d. Difficulty can be confirmed by pressing shoot button. Program will enter in play mode
2. In play mode target will move according to selected difficulty level.
    a. In play mode, position of bow can be adjust by moving joystick up-down
    b. Speed of the arrow can be set by pulling it back or forward by using joystick left-right.
    c. Arrow can be shot by pressing push button of the joystick.
3. Scoring
    a. If arrow will hit in centre then display will show a message “Bull’s Eye” and 300 points will be awarded.
    b. For hitting the adjacent areas (3 pixels) in both side of center, 250 points will be awarded.
    c. For hitting the adjacent areas (6 pixels) in both side of center, 200 points will be awarded.
    d. For hitting the adjacent areas (9 pixels) in both side of center, 100 points will be awarded.
    e. For hitting at the edges of the target 50 points will be awarded.
    f. The display will show “missed” message when the arrow misses the target.
4. Total 6 arrows will be given in one game.
5. Score will be added after every chance in a game.
6. Total score will be shown after completion of 6 chances.

6. Results

The photos of the implementation are shown as follows:

Fig. 5 Joystick and buzzer

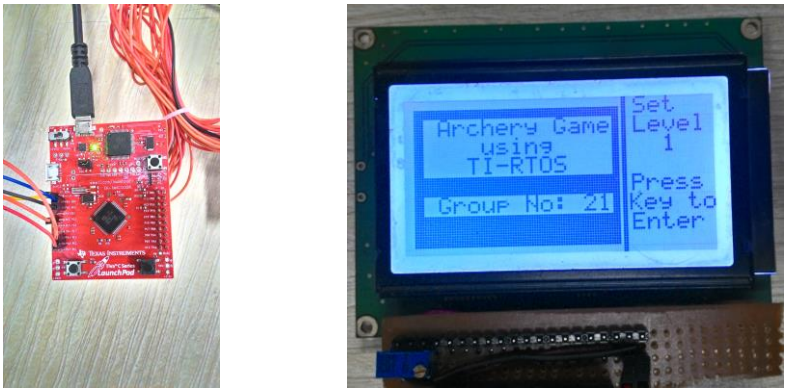

Fig. 6 Tiva C Series Launchpad and Graphics LCD

7. Conclusion

The archery game is implemented using TIVA C Series Launchpad, graphics LCD and joystick using TI-RTOS. The various tasks are scheduled periodically by releasing their respective semaphores.

8. Achievements Suggestions for further improvement

Achievements

- Implemented archery game using TI-RTOS with a fixed scheduler for multiple periodic tasks.
- Implemented selective modification of the graphics LCD on a particular section instead of writing to entire LCD for showing any change.

Suggestions

- Implementation using event based RTOS
- Increasing the complexity of the game algorithm by providing the facility to have angle control of the arrow.

References

[1] Tiva™ TM4C123G Development Board User's Guide

http://www.ti.com/lit/ug/spmu357b/spmu357b.pdf

[2] TI-RTOS training workshop

http://processors.wiki.ti.com/index.php/Introduction_to_the_TI-RTOS_Kernel_Workshop?keyMatch=ti%20rtos&tisearch=Search-EN

[3] TI-RTOS: A real-time operating system for TI devices

http://www.ti.com/lit/ml/sprt646a/sprt646a.pdf

[4] 128X64 Graphics LCD datasheet

http://www.agspecinfo.com/pdfs/J/JHD12864.PDF

[5] Tiva™ TM4C123GH6ZRB Microcontroller datasheet

http://www.ti.com/lit/ds/symlink/tm4c123gh6zrb.pdf

[6] Embedded Systems - Shape the World

https://www.edx.org/course/embedded-systems-shape-world-utaustinx-ut-6-02x