Special Project Computer architecture

Roberth Reynel Rosero S251640
Carlos Jose Peña Agreda S250953

June 20, 2018

Objectives

The general objective of the project is to initiate the work with the new boards LandTiger V2.0 with the aim of generating the foundations for new laboratories using this board. The project is focused in the use of some peripherals and services routines that help the student understand how to interact with a board at a low level. In this document we present the use of the following peripherals:

- LCD display
- Joystick
- Buttoms
- Timers
- Potentiometer

Libraries

In order to make the display work, a library that was already working was used. This library was a modification of one of the board KEIL MCB1700 that was found on the Internet. Additionally, we also used libraries that were already created and implemented on the laboratory of Computer Architecture like:

- The libraries of the timer
- The libraries of the buttons

Communication with the Processor

In order to reduce the load on the processor we used interruptions for communication as much as we could. Unfortunately, we had to use polling for the joystick. Due to the architecture of this board, GPIO interruptions are only supported for the ports P0 and Ports P2 and the joystick are in port P1.

Implementation

For the implementation of all the peripherals and libraries, we decided to make a Snake Game which can be controlled with the Joystick. This peripheral controls the direction in which the snake moves. The goal of the game is to control the snake avoiding to crash with the borders of the screen and to eat the 10 dots in order to win. Furthermore, the user can control the speed of the snake with the potentiometer selecting one of the different levels of the game.

Movement function

This is the principal library of the game and was created based on the behavior of the Joystick and the actual direction of the snake, the directions are shown in the figure 1.

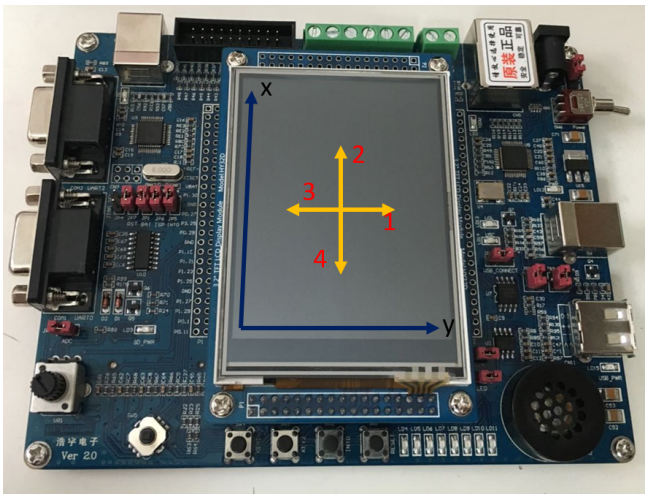


Figure 1: Axes and directions

In the function, the snake follows always a direction if the joystick is not pressed and, if the joystick is pressed, the snake goes to the direction pressed by the joystick, this behavior is managed with a switch function as shown in the following piece of code:

```
switch(direction){
 case 1:
  joys = LPC_GPIO1->FIOPIN;
    if ((joys & (1<<29))==0){//left
```

The case 1: in the code represents the actual direction of the snake, and always reads the value of the Joystick and if pressed in a different direction there is a change in the direction of the movement of the snake.

When the snake reaches the limit of the LCD display, the functions end and call the function findeljuego() that consist in a menu where the user can select to restart the game.//

In order to know the state of the joystick we used Polling, so every time the function is called, the processor checks the state of the joystick. It wasn't the most efficient way but as previously explained, the port of the joystick is not supported with Interruption communication.

For the process of drawing the snake, the function movement() prints always 3 squares, so the snakes moves by printing and erasing at the same time. Once the joystick is pressed, the direction of the snake changes and so does the direction of the printing. In the figure 2 we can see how it works.

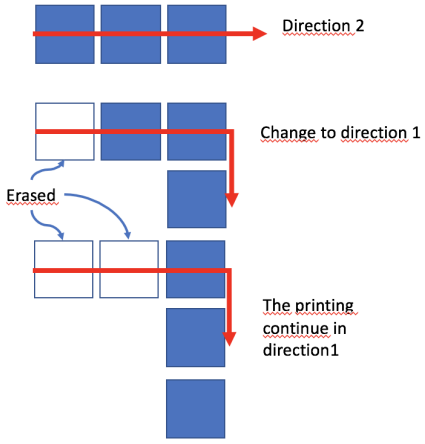


Figure 2: Movement function

Potentiometer

For the use of the potentiometer, first we needed to initialize and configure it, for this we used a different kind of register as shown in the following piece of code:

```
void Potenciometer_init(void){
 LPC_PINCON->PINSEL3 |= 0xC0000000; /* P1.31, A0.5, function 11 */
 LPC_SC->PCONP |= (1<<12); // Enable power to ADC block
 LPC_ADC->ADCR = (1<<5) | //                  select AD0.5 pin
                 (4<<8) | // ADC clock is 25MHz/5
                 (1<<21); // enable ADC
 LPC_ADC->ADCR |= (1<<24);//|(1<<16); // starts the A/D converter


 LPC_ADC->ADINTEN &= (1<<8); // global enable interrupt

```

```
12  NVIC_EnableIRQ(ADC_IRQn); // enable ADC Interrupt
13 }
```

To obtain the value of the potentiometer and control the difficulty of the game, we used the ADC conversion from the potentiometer. The behavior flow chart of the ADC conversion is show in the next part.

General link Potentiometer and Timer

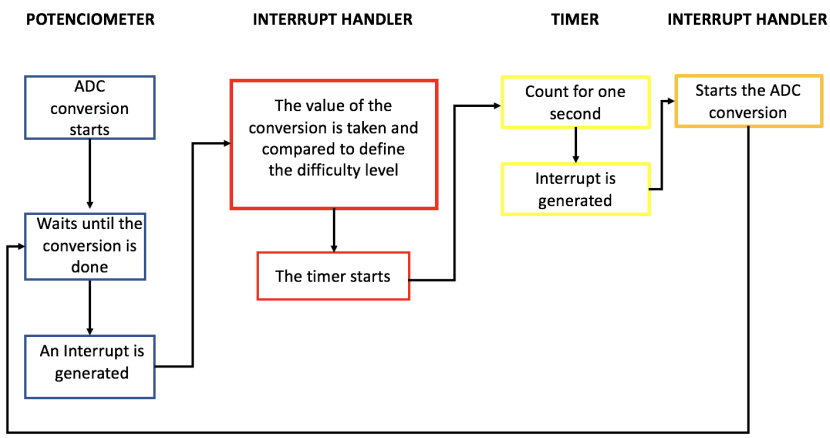


Figure 3: Potentiometer and Timer

We can see in the image above how the potentiometer and the timer are linked. The potentiometer starts the conversion, and once the conversion is done, an interrupt is generated. In the interrupt handler of the potentiometer the value of the conversion is taken and compared to define the difficulty level.

Once the comparison is done, the timer starts. The timer is set to count for 1 second after which an interrupt is generated. In the interrupt handler of the timer, the ADC conversion is initiated once again.

This conversion is reached through the timer interrupt, because this timer fires the ADC conversion and when the conversion is complete, it fires again the timer.

Graphical Interface

For the graphical interface we used a function of the GLCD library that prints a string of characters together whit a while loop that allows us to know the state

of the joystick. Depending on the joystick, we print the arrow that points to the desired window and when the user press the joystick, selects the next window. Each window has it's own function. We can see a piece of code bellow.

```
void instructions(){
  GLCD_DisplayString(4,0,0,(unsigned char*)"1. Use the joystick to
    move the snake and avoid crash");
  GLCD_DisplayString(5,3,0,(unsigned char*)"against the wall.");
  GLCD_DisplayString(7,0,0,(unsigned char*)"2. The game have 4
    difficulty levels:");
  6,3,0,(unsigned char*)"ving to the left, you decrease it.");
  /*In this part, wait for the user's selection, play or back*/
  while (inside == 1){
    delay_ms(95);
    key= joystick();
    GLCD_SetTextColor(White);
    GLCD_DisplayString(selection_line,13,0,(unsigned char*)"->");
    switch (selection){
      case 1:
        if (key== 'S'){
          play_game();/*Starts the game*/
          inside = 0;
          break;
        }
        else{break;}
      case 2: ....
        }
  }
```

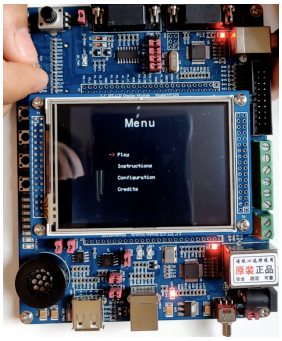
Main Code

In the main code we have the different initialization of the functions for the potentiometer, the system in general or the GLCD. Once we have done this, we use the interrupt of the system that counts the ticks every 10ms. This allow us to use certain functions after some amount of time has passed. Using this type of interrupt, we created different variables using a different function that allows us to have different flags depending on the time that has passed. Once we have those flags, we can control the time in which we call the function movement. This translates into controlling the velocity of the snake.

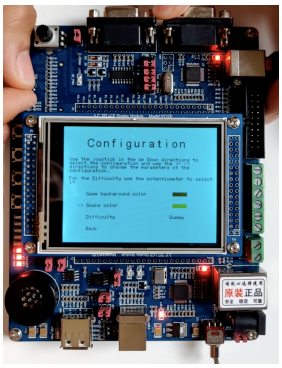
```
int main (void) {
  SysTick_Config(SystemCoreClock/100);  /*Every 10ms will be an
    interruption (SCC = 10kHz)*/
  while (1) {                           /* Loop forever
                */
    if(ClockTicks){
      ClockTicks=0;
      direction_change();/*Allows the snake to change the
      dorections*/
    }
    /*With the colition we evaluate if the snake have "eated" a
    piece of "food in the game"*/
    colition();
```

```
12      switch(lvl){// depending on the level we use diferent
        ClockRefresh to call the function more frequently or less
        frequently
13        case 0: if(ClockRefresh0){ClockRefresh0=0;movement();}
14        case 1: if(ClockRefresh1){ClockRefresh1=0;movement();}
15        ...
16      }
17    }
18  }
```

1 Results



(a) Menu



(b) Configuration

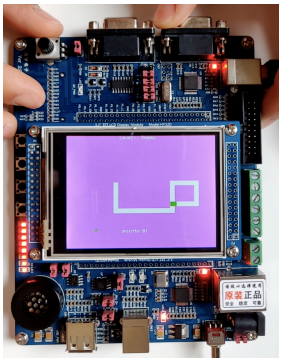


Figure 5: Snake game

We can see in the images above the results of the project like the menu screen with the different options that the user has before starting the game. There is also an image that shows the different configuration that can be changed. And in the last image we can see how the game looks. //

For the laboratory we propose that the students use all of the libraries given to recreate the game. In this way, they can get a more profound understanding of how the interruption works and how they can link them with each other.