Universidad del Valle de Guatemala
2021

Marco Tulio Trujillo Lara
18069
Digital ll

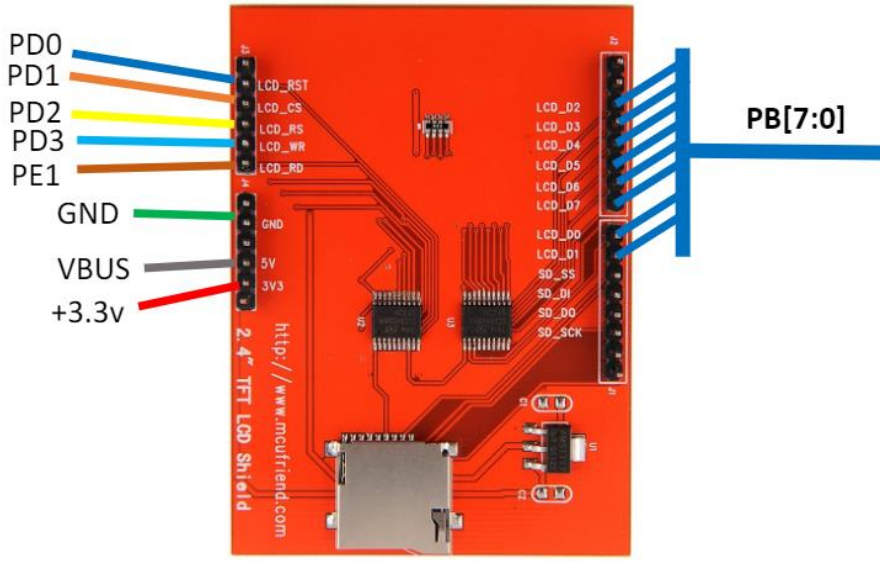
Proyecto #3

30/04/2021

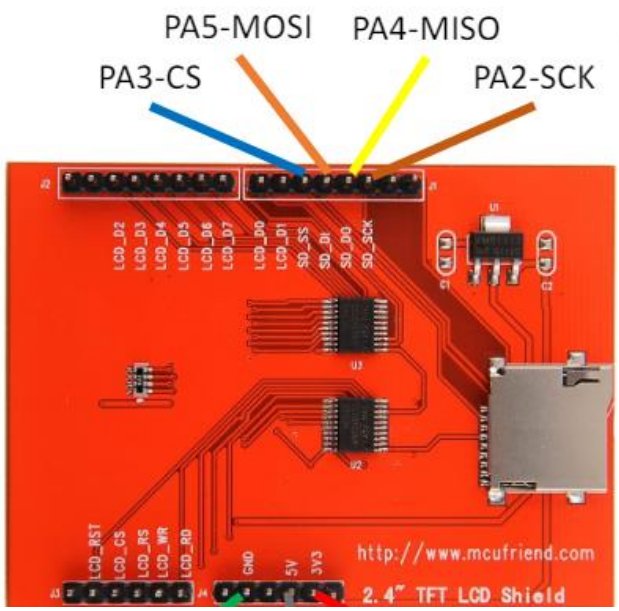
Circuito

Para este proyecto se utilizó únicamente la Tiva-C y una pantalla TFT LCD de 320x240 pixeles. Los pines utilizados en las conexiones corresponden a los proporcionados en las presentaciones de clase; para la pantalla de utiliza el puerto PB [7:0] y se utiliza el módulo SPI (0) para la memoria micro SD.

Pines para conexión de la pantalla LCD.



Pines para utilizar módulo SPI (0) para la memoria micro SD.



Código

El código está separado en distintas secciones para que se entienda más fácilmente, de la siguiente manera:

Void Setup

```
void setup() {
  SysCtlClockSet(SYSCTL_SYSDIV_2_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
  Serial.begin(9600);
  GPIOPadConfigSet(GPIO_PORTB_BASE, 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7, GPIO_STRENGTH_8MA, GPIO_PIN_TYPE_STD_WPU);
  Serial.println("Inicio");
  LCD_Init();

  pinMode(PUSH1, INPUT_PULLUP);
  pinMode(PUSH2, INPUT_PULLUP);

  enemyctr = 1;

  SPI.setModule(0);
  Serial.print("Initializing SD card...");
  pinMode(PA_3, OUTPUT);
  if (!SD.begin(PA_3)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");
  Serial.println("La Memoria posee los siguientes archivos:");
  root = SD.open("/");
  printDirectory(root, 0);


  enemy1 = 0;
  enemy2 = 0;
  enemy3 = 0;
  enemy4 = 0;

  posx = 230;
  punteo = 0;
```

En esta sección se realiza la configuración inicial, tal como las palabras de configuración para la Tiva-C, el inicio de la comunicación serial (se utiliza para monitorear que todo esté en orden), se inicializa la LCD con el comando LCD_Init (), se configuran los botones SW1 y SW2 de la Tiva-C como entradas de tipo pullup, se inicializa el modulo cero del SPI, podemos verificar si inicializo correctamente por medio del monitor de comunicación serial y podemos verificar los archivos que posee actualmente la micro SD, ponemos en cero las banderas de los autos enemigos y el punteo, también especificamos la posición inicial del jugador en el eje x.

Void loop

Pantalla de carga

```
  //DESPLIEGUE PANTALLA DE CARGA + SONIDO
  LCD_Bitmap(0, 0, 320, 240, fondo);
  thisNote = 0;
  tempo = 2500;
  notes = sizeof(melody) / sizeof(melody[0]) / 2;

//this calculates the duration of a whole note in ms
   wholenote = (60000 * 4) / tempo;
   divider = 0, noteDuration = 0;

  while(digitalRead(PUSH2)== 1 && digitalRead(PUSH1)== 1){
  divider = melody[thisNote + 1];
    if (divider > 0) {
      // regular note, just proceed
      noteDuration = (wholenote) / divider;
    } else if (divider < 0) {
      // dotted notes are represented with negative durations!!
      noteDuration = (wholenote) / abs(divider);
      noteDuration *= 1.5; // increases the duration in half for dotted notes
    }

    // we only play the note for 90% of the duration, leaving 10% as a pause
    tone(buzzer, melody[thisNote], noteDuration * 0.9);

    // Wait for the specief duration before playing the next note.
    delay(noteDuration);

    // stop the waveform generation before the next note.
    noTone(buzzer);

  LCD_Bitmap(60, 182, 200, 13, text1);
  delay(100);
  LCD_Bitmap(60, 182, 200, 13, text2);
  delay(100);
  thisNote = thisNote + 2;
  if (thisNote == 4){
    thisNote = 0;
   }
  }
```

Primero cargamos el fondo de la pantalla de inicio en la LCD, luego podemos observar que tenemos un siclo While que estará repitiéndose hasta que alguno de los dos botones de la Tiva-C sean presionados, dentro de este siclo estamos imprimiendo constantemente un fragmento de la imagen de fondo que contiene la oración “Press any button to start game” dando así un efecto de tintineo, al cual lo acompaña un sonido parecido a un pequeño “beep”. Si alguno de los botones es presionado pasamos la siguiente etapa, que es el despliegue de instrucciones.

Despliegue de Instrucciones

```
379   //DESPLIEGUE INSTRUCCIONES
380   LCD_Clear(0x605F);
381   LCD_Print("How to play?" , 60, 30, 2,    0xFFFF, 0x605F);
382   LCD_Bitmap(60, 70, 186, 43, btns);
383   LCD_Print("Move to the left with SW1" , 20, 130, 1,    0xFFFF, 0x605F);
384   LCD_Print("Move to the right with SW2" , 20, 150, 1,    0xFFFF, 0x605F);
385   LCD_Print("Have Fun :)" , 60, 190, 2,    0xFFFF, 0x605F);
386  //sonido de inicio
387   thisNote = 0;
388   tempo = 100;
389   notes = sizeof(melody) / sizeof(melody[0]) / 2;
390
391 //this calculates the duration of a whole note in ms
392    wholenote = (60000 * 4) / tempo;
393    divider = 0, noteDuration = 0;
394    while (thisNote != 30){
395    divider = melody[thisNote + 1];
396     if (divider > 0) {
397       // regular note, just proceed
398       noteDuration = (wholenote) / divider;
399     } else if (divider < 0) {
400       // dotted notes are represented with negative durations!!
401       noteDuration = (wholenote) / abs(divider);
402       noteDuration *= 1.5; // increases the duration in half for dotted notes
403     }
404
405     // we only play the note for 90% of the duration, leaving 10% as a pause
406     tone(buzzer, melody[thisNote], noteDuration * 0.9);
407
408     // Wait for the specief duration before playing the next note.
409     delay(noteDuration);
410
411     // stop the waveform generation before the next note.
412     noTone(buzzer);
413
414     thisNote = thisNote + 2;
415    }
416   LCD_Clear(0x605F);
```

Primero realizamos una limpieza de la pantalla y se le asigna un color determinado de fondo parecido al azul. Se imprime la cadena de texto “How to play?” y luego se muestra una pequeña imagen de los botones que están en la Tiva-C, después se imprimen dos cadenas de texto más, las cuales explican que se debe presionar el botón izquierdo para moverse hacia la izquierda y el derecho para moverse hacia la derecha. Por ultimo se reproduce una pequeña melodía para cumplir la función de “delay” y dar tiempo suficiente de leer las instrucciones antes de comenzar el juego.

Generar escenario

```
//Generar escenario de juego
LCD_Bitmap(0, 0, 320, 240, fondo);
FillRect(170, 0, 320-170, 240, 0x6B4D);
FillRect(140, 35, 20, 20, 0x1C59);
LCD_Print("Puntos:" , 2, 190, 2, 0xFFFF, 0xCB26);
```

Se genera el escenario del juego imprimiendo el mismo fondo de la pantalla de inicio y rellenando con rectángulos que coincidan con el color de la carretera. También se imprime el texto “Puntos” donde se ira sumando la puntuación del jugador.

Valores iniciales de las variables

```
//Valores iniciales
int posplayer = 2;
int coin = 0;
check1 = 0;
check2 = 0;
enemy1 = 0; enemy2 = 0; enemy3 = 0; enemy4 = 0;
enemy1x = 0; enemy2x = 0; enemy3x = 0; enemy4x = 0;
enemy1y = 0; enemy2y = 0; enemy3y = 0; enemy4y = 0;
enemyctr = 0;
punteo = 0;
LCD_Bitmap(230, 170, 18, 25, player);
while(coin != 1){
```

Antes de comenzar el juego se establecen los valores iniciales de las variables que están involucradas en su funcionamiento y también se imprime la posición inicial del jugador. Por ultimo entramos a un ciclo While que se estará repitiendo hasta que el jugador pierda.

Sistema anti rebotes

```
//revisar si se han presionado los botones (anti-rebote incluido)
if (check1 != 0 && digitalRead(PUSH1) == 1){
         check1 = 0;
        }

 else if (check2 != 0 && digitalRead(PUSH2) == 1){
          check2 = 0;
        }

 else if (check1 == 0 && digitalRead(PUSH1) == 0 && posplayer != 1){
      posplayer = posplayer - 1;
      check1 = 1;
      changepos = 1; //bandera de cambio de posicion
     }

 else if (check2 == 0 && digitalRead(PUSH2) == 0 && posplayer != 3){
      posplayer = posplayer + 1;
      check2 = 1;
      changepos = 1; //bandera de cambio de posicion
     }
```

Este sistema anti rebote evita que una vez presionado cualquiera de los dos botones se registre más de un movimiento del jugador. La variable “posplayer” nos indica en que carril debe estar el jugador. Izquierda, centro o derecha (1, 2 o 3 respectivamente).

Cambio de posición del jugador

```
//Animacion, posicion del jugador
if (changepos == 1){
 switch(posplayer){
  case 1:
    posx = 182;
    LCD_Bitmap(posx, 170, 18, 25, player);
    FillRect(230, 170, 18, 25, 0x6B4D);
    FillRect(280, 170, 18, 25, 0x6B4D);

    tone(buzzer, melody[12], 40);
    delay(40);
    noTone(buzzer);
  break;

  case 2:
    posx = 230;
    LCD_Bitmap(posx, 170, 18, 25, player);
    FillRect(182, 170, 18, 25, 0x6B4D);
    FillRect(280, 170, 18, 25, 0x6B4D);

    tone(buzzer, melody[12], 40);
    delay(40);
    noTone(buzzer);
  break;

  case 3:
    posx = 280;
    LCD_Bitmap(posx, 170, 18, 25, player);
    FillRect(182, 170, 18, 25, 0x6B4D);
    FillRect(230, 170, 18, 25, 0x6B4D);

    tone(buzzer, melody[12], 40);
    delay(40);
    noTone(buzzer);
  break;
 }
 changepos = 0;
}
```

Si se ha detectado que han presionado uno de los botones se procede a realizar el cambio de posición del jugador. Se imprime el carrito del jugador en el carril izquierdo si la variable posplayer es igual a 1, en el centro si es igual a 2 y a la derecha si es igual a 3. En cada caso también se procede a “borrar” la posición anterior del carrito del jugador. Asi mismo se genera un pequeño tono cuando el carrito cambia de carril.

Animación de la carretera

```
//animacion de la carretera
i = 0;
while (i != 11){
  j = 0;
  posy = 0;
  while (j != 5){
    LCD_Sprite(210, posy, 9, 48, middle, 11, i, 0, 0);
    LCD_Sprite(260, posy, 9, 48, middle, 11, i, 0, 0);
    if (i < 8){
      LCD_Sprite(160, posy, 14, 48, borde, 8, i, 0, 0);
      LCD_Sprite(320-14, posy, 14, 48, borde, 8, i, 1, 0);
    }
    else{
      LCD_Sprite(160, posy, 14, 48, borde, 8, i-8, 0, 0);
      LCD_Sprite(320-14, posy, 14, 48, borde, 8, i-8, 1, 0);
    }

    posy = posy + 48;
    j = j + 1;
  }
  i = i + 1;
}
```

Para la carretera se utilizaron dos sprites, uno que corresponde a las lineas de los bordes y otro que corresponde a las lineas de los carriles de la carretera, cada uno tenía un total de 48 pixeles de alto, por lo que se necesitaba imprimir 5 copias de cada sprite por línea de borde o carretera. El sprite de la carretera tenía un total de 11 cuadros, mientras que el de los bordes solamente tenía 8. El ciclo While permite realizar una secuencia completa del sprite de las lineas de la carretera y mostrar una secuencia más 3 cuadros del borde, lo cual afectaba a la animación del borde, pero era poco perceptible.

Asignar el carril a los enemigos de forma aleatoria

```
//Asignar carril aleatorio a cada enemigo
  if (enemy1 == 0 && enemyctr == 10){
    enemy1x = enemyrand();
    enemy1y = 0;
    enemy1 = 1;
  }

  else if(enemy2 == 0 && enemyctr == 15){
    enemy2x = enemyrand();
    enemy2y = 0;
    enemy2 = 1;
  }

  else if(enemy3 == 0 && enemyctr == 25){
    enemy3x = enemyrand();
    enemy3y = 0;
    enemy3 = 1;
  }

  else if(enemy4 == 0 && enemyctr == 30){
    enemy4x = enemyrand();
    enemy4y = 0;
    enemy4 = 1;
  }
```

```
int enemyrand(void){
  posenemy = rand() % 3;
  switch(posenemy){
   case 0: enemyx = 182; break;
   case 1: enemyx = 230; break;
   case 2: enemyx = 280; break;
  }

  return enemyx;
}
```

Si la bandera del enemigo 1 está en cero (enemy1 = 0) esto quiere decir que aún no ha aparecido dentro del juego, si esto es así entra a la función para asignarle un carril de forma aleatoria con la función enemyrand, la cual utiliza la función integrada rand para generar un numero aleatorio entre 1 y 3 y asignar la posición en x del enemigo. La variable enemyctr sirve para generar un pequeño delay antes de la aparición del enemigo en la pantalla. Lo mismo aplica para el enemigo 2, 3 y 4.

Animación de la aparición del enemigo

```
//Spawn de los enemigos
  if (enemy1 == 1 && enemy1y == 0){
    k = 7;
    while(k != 11){
    LCD_Sprite(enemy1x, enemy1y, 20, 27, enemy, 11, k, 0, 0);
    k = k + 1;
    delay(30);
    }
    enemy1y = 2;
  }
```

Se utiliza un sprite para que la aparición del enemigo se vea más natural, de forma que va apareciendo poco a poco y se asigna la posición inicial en y.

Movimiento y desaparición de los enemigos

```
//Movimiento y despawn de los enemigos
  if (enemyly !=0 && enemyly < 210 ){
    LCD_Sprite(enemylx, enemyly, 20, 27, enemy, 11, 1, 0, 0);
    enemyly = enemyly + 7;
  }

  else if(enemyly >= 210 && enemyl == 1){
    k = 0;
    while(k != 6){
    LCD_Sprite(enemylx, 212, 20, 27, enemy, 11, k, 0, 0);
    k = k + 1;
    delay(30);
    }
    enemyl = 0;
    punteo = punteo + 1;
  }
```

Vemos que entramos al if si se cumple que la bandera del enemigo no sea cero y su posición sea menor a 210 pixeles en el eje y lo que hace es simplemente aumentar 7 pixeles a la posición anterior del carrito enemigo, lo que hará que la próxima vez que se imprima lo haga una posición más abajo. Luego, si la bandera del enemigo es uno y su posición en y ha superado los 210 pixeles procedemos a realizar la animación de “desaparición”, desactivamos la bandera del enemigo y sumamos un punto a la puntuación del jugador. Esto se repite exactamente igual para el enemigo 2, 3 y 4.

“Delay” de aparición

```
//contador para generar enemigos (delay)
  enemyctr = enemyctr + 1;
  if (enemyctr == 31){
    enemyctr = 0;
  }
```

La variable enemyctr es la encargada de generar un pequeño delay entre la aparición de los carritos enemigos, cuando este llegue al número 31 simplemente vamos a resetearlo para que los tiempos de aparición sean los adecuados.

Mostrar punteo

```
//Mostrar punteo
  puntostxt = String(punteo);
  LCD_Print(puntostxt , 120, 190, 2,   0xFFFF, 0xCB26);
```

Para el punteo simplemente se transforma la variable “punteo” de tipo numero a tipo texto y se guarda en “puntostxt”, luego se imprime con la función LCD_Print.

Lógica para el game over

```
//Logica para el game over
  if (enemyly >= 150 && enemyly <= 195 && enemylx == posx){
    coin = 1;

    tone(buzzer, 55, 300);
    delay(300);
    noTone(buzzer);
    tone(buzzer, 31, 300);
    delay(300);
    noTone(buzzer);
  }
```

Para el game over solamente comparamos la posición en y del enemigo 1, 2, 3 y 4 para saber si es igual a la del jugador, así mismo si la posición en x es la misma. Si estos dos valores son verdaderos quiere decir que un vehículo enemigo a tocado al jugador; por lo que el valor de la variable coin cambia a 1 y saldremos del siclo while que empezó desde la sección “Valores iniciales de las variables”, también se reproducirá un sonido que indica que hemos chocado.

Punteo final y pantalla de game over

```
//Game over y punteo final
  LCD_Clear(0x605F);
  LCD_Print("GAME OVER" , 80, 30, 2,   0xFFFF, 0x605F);
  LCD_Print("Your score: " , 70, 110, 1,   0xFFFF, 0x605F);
  LCD_Print(puntostxt , 180, 110, 2,   0xFFFF, 0x605F);
  LCD_Print("Thanks for playing" , 20, 190, 2,   0xFFFF, 0x605F);
  //sonido
  tempo = 100;
  wholenote = (60000 * 4) / tempo;
  divider = 0, noteDuration = 0;
  thisNote = 32;
  while (thisNote != 90){
   divider = melody[thisNote + 1];
    if (divider > 0) {
      // regular note, just proceed
      noteDuration = (wholenote) / divider;
    } else if (divider < 0) {
      // dotted notes are represented with negative durations!!
      noteDuration = (wholenote) / abs(divider);
      noteDuration *= 1.5; // increases the duration in half for dotted notes
    }

    // we only play the note for 90% of the duration, leaving 10% as a pause
    tone(buzzer, melody[thisNote], noteDuration * 0.9);

    // Wait for the specief duration before playing the next note.
    delay(noteDuration);

    // stop the waveform generation before the next note.
    noTone(buzzer);

    thisNote = thisNote + 2;
   }
```

Luego de salir del ciclo While limpiamos la pantalla con un color especifico, mostramos al jugador el total de sus puntos obtenidos en la partida y como delay utilizamos una pequeña melodía.

Guardar en la SD

```
//Guardar puntaje en la sd
  myFile = SD.open("PTS.txt", FILE_WRITE);
  if (myFile) {
    Serial.print("Writing to PTS.TXT...");
    myFile.println("Puntuacion: " + puntostxt);
    // close the file:
    myFile.close();
    Serial.println("done.");
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
```

Por último, guardamos el punteo en la memoria micro SD insertada en la pantalla.

Bibliografía

Para la pantalla y la memoria micro SD se utilizaron las librerías proporcionadas por el catedrático del curso.

Para la música se utilizó como base el proyecto de Robson Couto el cual se encuentra en GitHub en el siguiente enlace: https://github.com/robsoncouto/arduino-songs.

Este proyecto está basado en el juego Road Fighter desarrollado por Konami.