Seminario: Introducción a la Programación de Microcontroladores con Tecnologías Libres

Ardu-Pong!

Integrantes:

- Emanuel Dubor
- Nahuel Benitez
- Marcelo Garzón
- Martin Carniello

Índice

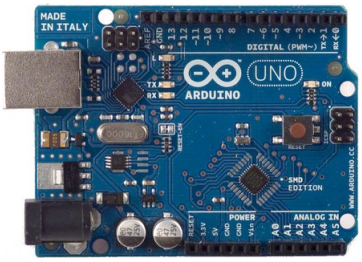
Descripción

Ardu-Pong! es una nueva versión del ya conocido juego “Pong”, el cual consiste en un tablero de dos dimensiones que simula ser un tenis de mesa. Se juega de a dos jugadores, los cuales controlan una paleta, la cual puede moverse en dirección ascendente o descedente. El objetivo del juego es que cada jugador le pegue a la pelota con su paleta. Si un jugador falla al devolver la pelota, instantaneamente se le dará un punto al jugador oponente y el que falló, deberá hacer un nuevo saque. El jugador que tenga más puntos al final de la partida es el victorioso.

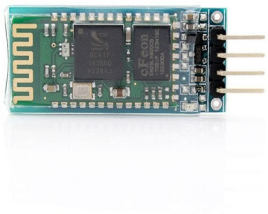
En nuestro caso se juega a travez de un celular con Bluetooth y se puede observar la partida a traves de una televisor con un conector RCA.

Materiales

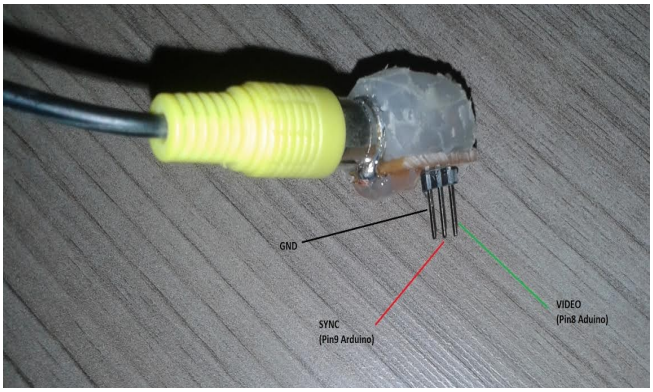
Arduino Uno:



Módulo Bluetooth:
(HC-06)

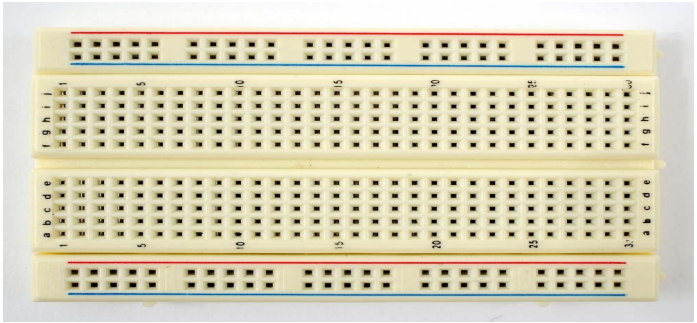


Módulo RCA:
(Casero)

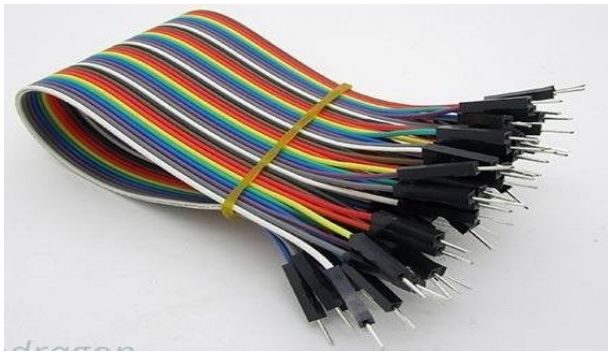


Televisión con conexión RCA:

Protoboard:



Cables:

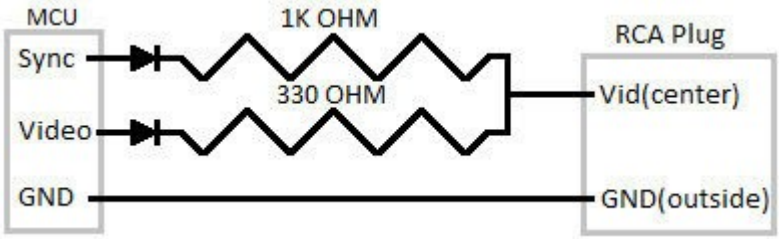


Celular con Bluetooth:

Montaje:

RCA

El modulito de video tiene 3 pines:
- GND: que va al GND del Arduino/protoboard
- VIDEO: que va al pin 8 del Arduino
– SYNC: que va al pin 9 del Arduino



HC-06

Solo usamos 3 pines:
- GND: que va al GND del Arduino/protoboard
- VCC: que va al 5V / 3.3V del Arduino/protoboard
- TX (transmision serie): que va al RX (recepcion serie) del Arduino
- (el RX del modulo no se usa, creo)

Código Arduino:

```
//Attribution Non-commercial Share Alike (by-nc-sa)
//This license lets others remix, tweak, and build upon your work
//non-commercially, as long as they credit you and license their
//new creations under the identical terms.Others can download and
//redistribute your work just like the by-nc-nd license, but they
//can also translate,make remixes, and produce new stories based
//on your work. All new work based on yours will carry the same license,
//so any derivatives will also be non-commercial in nature.
//-----------------------------------------------------
//| original author: Pete Lamonica                    |
//| atari input mod: kyle brinkerhoff                 |
//|                                                   |
//-----------------------------------------------------
// Feb 2015, Porting to TVOut new library by Afentakis Andreas
//                                                  -- Several correction & bug fixies
//                                                  -- change joystick to weels
//
//

#include <TVout.h>
//#include <font6x8.h>
//#include <font4x6.h>

#define WHEEL_ONE_PIN 0 //analog
#define WHEEL_TWO_PIN 1 //analog
#define BUTTON_ONE_PIN 2 //digital

#define PADDLE_HEIGHT 12
#define PADDLE_WIDTH 2

#define RIGHT_PADDLE_X (TV.hres()-4)
#define LEFT_PADDLE_X 2

#define IN_GAME 0 //in game state
#define IN_MENU 1 //in menu state
#define GAME_OVER 2 //game over state

#define LEFT_SCORE_X (TV.hres()/2-15)
#define RIGHT_SCORE_X (TV.hres()/2+10)
#define SCORE_Y 4

#define MAX_Y_VELOCITY 3
#define PLAY_TO 7

#define LEFT 0
#define RIGHT 1

// UNQ-ARDUPONG
#define JOYSTICK_SIDE_MASK      0x40
#define JOYSTICK_POSITION_MASK  0x3F
#define BUTTON_PRESS            0x3F
#define LEFT_JOYSTICK           0x00
#define RIGHT_JOYSTICK          0x40
#define JOYSTICK_MAX_VALUE      63
// UNQ-ARDUPONG

TVout TV;
unsigned char x, y;
boolean buttonStatus = false;
int wheelOnePosition = 0;
int wheelTwoPosition = 0;
int rightPaddleY = 0;
int leftPaddleY = 0;
unsigned char ballX = 0;
unsigned char ballY = 0;
char ballVolX = 1;
char ballVolY = 1;
int initialpos = 500;
int leftPlayerScore = 0;
int rightPlayerScore = 0;
int upPin = 7;
```

```
int dnPin = 6;
int joystickstate = 2;
int dnst;
int upst;
int frame = 0;
int state = IN_MENU;


void processInputs()
{
  /* // UNQ-ARDUPONG - Desactivamos la logica original...
              wheelOnePosition = analogRead(WHEEL_ONE_PIN);
              wheelTwoPosition = analogRead(WHEEL_TWO_PIN);

              buttonStatus = (digitalRead(BUTTON_ONE_PIN) == HIGH);
    // UNQ-ARDUPONG
  */

  // UNQ-ARDUPONG - Implementamos la logica para procesar comandos enviados por los joysticks Bluetooth...
  char receivedByte;
  char joystickPosition;

  buttonStatus = false; // Por defecto, al boton lo seteamos como 'no presionado'...

  // Mientras haya comandos sin procesar enviados por el modulo Bluetooth...
  while( Serial.available() )
  {
    // Leer el siguiente comando en el buffer de entrada...
    receivedByte = (char) Serial.read();

    // Extraer el valor codificado dentro del comando...
    joystickPosition = (receivedByte & ((char)JOYSTICK_POSITION_MASK) );

    // Si el comando recibido es del tipo "boton presionado"...
    if( joystickPosition == BUTTON_PRESS )
    {
       buttonStatus = true;  // Setear el boton como 'presionado'...
    }
    else
    {   // Si en cambio el comando recibido es del tipo "posicion de joystick"...

       // Asignar el valor de posicion al 'wheelXPosition' que corresponda dependiendo del joystick que
       // haya enviado el comando.
       // El valor de posicion enviado por los joysticks Bluetooth va de 0 a JOYSTICK_MAX_VALUE
       // por lo que hay que mapearlo al rango 0 a 1023 que espera la logica del programa original.
       // (el diseño 'cree' que cada joystick es un potenciometro conectado a una entrada analogica).
       joystickPosition = JOYSTICK_MAX_VALUE - joystickPosition; //(invertir el valor para que suba y baje igual que el joystick)
       if( (receivedByte & JOYSTICK_SIDE_MASK) == RIGHT_JOYSTICK )
          wheelOnePosition = map( joystickPosition, 0, JOYSTICK_MAX_VALUE, 0, 1023 );
       else
          wheelTwoPosition = map( joystickPosition, 0, JOYSTICK_MAX_VALUE, 0, 1023 );
    }
  }
  // UNQ-ARDUPONG
}

void joystick_processInputs()
{
              joystickstate = 2;

              pinMode(upPin, INPUT);
              pinMode(dnPin, INPUT);
              dnst = digitalRead(dnPin);
              upst = digitalRead(upPin);
              if(dnst == HIGH)

              {
                            initialpos = initialpos + 10;
                            dnst = 0;
              }
              if(upst == HIGH)
              {
                            initialpos = initialpos - 10;
                            upst = 0;
              }
              if (initialpos >= 1000)
              {
                            initialpos = 999;
              }
```

```
		if (initialpos <= 0)
		{
			initialpos = 1;
		}

		wheelOnePosition = initialpos;
		wheelTwoPosition = initialpos
;
		buttonStatus = (digitalRead(BUTTON_ONE_PIN) == HIGH);
}

void drawGameScreen()
{
		TV.clear_screen();
//draw right paddle
		rightPaddleY = ((wheelOnePosition / 8) * (TV.vres() - PADDLE_HEIGHT)) / 128;
		x = RIGHT_PADDLE_X;
		for(int i = 0; i < PADDLE_WIDTH; i++)
		{
			TV.draw_line(x + i, rightPaddleY, x + i, rightPaddleY + PADDLE_HEIGHT, 1);
		}

//draw left paddle
		leftPaddleY = ((wheelTwoPosition / 8) * (TV.vres() - PADDLE_HEIGHT)) / 128;
		x = LEFT_PADDLE_X;
		for(int i = 0; i < PADDLE_WIDTH; i++)
		{
			TV.draw_line(x + i, leftPaddleY, x + i, leftPaddleY + PADDLE_HEIGHT, 1);
		}

//draw score
		TV.print_char(LEFT_SCORE_X, SCORE_Y, '0' + leftPlayerScore);
		TV.print_char(RIGHT_SCORE_X, SCORE_Y, '0' + rightPlayerScore);

//draw net
		for(int i = 1; i < TV.vres() - 4; i += 6)
		{
			TV.draw_line(TV.hres() / 2, i, TV.hres() / 2, i + 3, 1);
		}

//draw ball
		TV.set_pixel(ballX, ballY, 2);
}

//player == LEFT or RIGHT
void playerScored(byte player)
{
		if(player == LEFT) leftPlayerScore++;
		if(player == RIGHT) rightPlayerScore++;

//check for win
		if(leftPlayerScore == PLAY_TO || rightPlayerScore == PLAY_TO)
		{
			state = GAME_OVER;
		}

		ballVolX = -ballVolX;
}

void drawMenu()
{
		x = 0;
		y = 0;
		char volX = 1;
		char volY = 1;
		TV.clear_screen();
//TV.select_font(font8X8);
//		TV.select_font(font6x8);
		TV.print_str(10, 5, "     Pong");

//TV.select_font(_5X7);
//		TV.select_font(font6x8);
		TV.print_str(5, 19, "by pete lamonica");
		TV.print_str(22, 35, "& kyle brinkerhoff");
		TV.print_str(50, 45, "press the red");
		TV.print_str(1, 80, "button to start");
		delay(1000);
		while(!buttonStatus)
		{
```

```
                processInputs();
                TV.delay_frame(3);
                if(x + volX < 1 || x + volX > TV.hres() - 1) volX = -volX;
                if(y + volY < 1 || y + volY > TV.vres() - 1) volY = -volY;
                if(TV.get_pixel(x + volX, y + volY))
                {
                        TV.set_pixel(x + volX, y + volY, 0);

                        if(TV.get_pixel(x + volX, y - volY) == 0)
                        {
                                volY = -volY;
                        }
                        else if(TV.get_pixel(x - volX, y + volY) == 0)
                        {
                                volX = -volX;
                        }
                        else
                        {
                                volX = -volX;
                                volY = -volY;
                        }
                }
                TV.set_pixel(x, y, 0);
                x += volX;
                y += volY;
                TV.set_pixel(x, y, 1);
        }
//TV.select_font(_5X7);
//      TV.select_font(font6x8);
        state = IN_GAME;
}

void setup()
{
        Serial.begin(9600);
        x = 0;
        y = 0;
//      TV.begin(PAL);//for devices with only 1k sram(m168) use TV.begin(_NTSC,128,56)
TV.begin(_NTSC,128,56);

        ballX = TV.hres() / 2;
        ballY = TV.vres() / 2;
}

void loop()
{
        processInputs();

        if(state == IN_MENU)
        {
                drawMenu();
        }
        if(state == IN_GAME)
        {
                if(frame % 3 == 0)   //every third frame
                {
                        ballX += ballVolX;
                        ballY += ballVolY;

                        if(ballY <= 1 || ballY >= TV.vres() - 1) ballVolY = -ballVolY;
                        if(ballVolX < 0 && ballX == LEFT_PADDLE_X + PADDLE_WIDTH - 1 && ballY >= leftPaddleY && ballY <=
leftPaddleY + PADDLE_HEIGHT)
                        {
                                ballVolX = -ballVolX;
                                ballVolY += 2 * ((ballY - leftPaddleY) - (PADDLE_HEIGHT / 2)) / (PADDLE_HEIGHT / 2);
                        }
                        if(ballVolX > 0 && ballX == RIGHT_PADDLE_X && ballY >= rightPaddleY && ballY <= rightPaddleY +
PADDLE_HEIGHT)
                        {
                                ballVolX = -ballVolX;
                                ballVolY += 2 * ((ballY - rightPaddleY) - (PADDLE_HEIGHT / 2)) / (PADDLE_HEIGHT / 2);
                        }

//limit vertical speed
                        if(ballVolY > MAX_Y_VELOCITY) ballVolY = MAX_Y_VELOCITY;
                        if(ballVolY < -MAX_Y_VELOCITY) ballVolY = -MAX_Y_VELOCITY;

                        if(ballX <= 1)
                        {
```

```
                                        playerScored(RIGHT);
                                }
                                if(ballX >= TV.hres() - 1)
                                {
                                        playerScored(LEFT);
                                }
                        }
                        if(buttonStatus) Serial.println((int)ballVolX);

                        drawGameScreen();
                }
                if(state == GAME_OVER)
                {
                        drawGameScreen();
//                      TV.select_font(font6x8);
//TV.select_font(_8X8);
                        TV.print_str(29, 25, "GAME");
                        TV.print_str(68, 25, "OVER");
                        while(!buttonStatus)
                        {
                                processInputs();
                                delay(50);

                        }
//                      TV.select_font(font6x8);
//TV.select_font(_5X7); //reset the font
//reset the scores
                        leftPlayerScore = 0;
                        rightPlayerScore = 0;
                        state = IN_MENU;
                }

                TV.delay_frame(1);
                if(++frame == 60) frame = 0; //increment and/or reset frame counter
}
```

Problemas

1.)
El objetivo principal del proyecto fué poder hacer que dos jugadores jueguen al juego con distintos dispositivos móviles, lo cual trajo un problema. El módulo BT que teníamos sólo soportaba una conexión en paralelo, por lo que teníamos dos opciones:

1- Conseguir otro módulo BT.

2- Hacer que los dos jugadores jueguen en un mismo dispositivo móvil.

Por temas económicos elegimos la segunda opción.

2.)
Otro de los problemas que tuvimos fue que quisimos usar un Arduino Mega 2560, pero la librería TVOut no soporta este tipo de Arduinos

3.)
La aplicación Android sólo corre con versiones de Android menores a 4.2. Luego de hacer varios intentos con un celular que tenía una versión más nueva, nos dimos cuenta del problema.

Referencias:

Instructables:
http://www.instructables.com

Ardu-Pong! con joystick:
http://www.instructables.com/id/Ardu-pong-the-Arduino-based-pong-console/?ALLSTEPS

TVOut:
http://code.google.com/p/arduino-tvout/
http://playground.arduino.cc/Main/TVout