Práctica 10 - Jugando al *Pong* con un sensor de distancia

Medina Medina, David A.

17 de abril de 2019

Índice general

Introducción

Arduino es el primer proyecto de hardware en open-source que facilita el diseño de productos digitales que requieren de un microcontrolador.

El objetivo de este proyecto (Medina Medina, s.f.) consiste en utilizar la placa *Arduino UNO R3* para leer los datos que nos llegan del sensor de distancia *GP2D12*. Este sensor será utilizado para mover el jugador derecho del juego *Pong* que se implementó en la práctica 1.

Este documento describe los cambios más importantes que se han realizado en el diseño del proyecto de la práctica 1 además del código que se ha diseñado para que la placa *Arduino Uno R3* lea y transmita la información del sensor de proximidad a *Processing*.

Método y materiales

2.1. Materiales

El desarrollo de este proyecto se ha llevado a cabo utilizando el IDE de desarrollo de aplicaciones *C/C++* y *Java* de *JetBrains*, *CLion* e *IntelliJ*, respectivamente. El resto de materiales hardware/software utilizados en este proyecto son enumerados a continuación:

- Placa Arduino Uno R3.
- Sensor de distancia *GP2D12*.
- Librería `Arduino.h` (Arduino, s.f.-b).
- Archivo `CMakeLists.txt` para la construcción automatizada de las dependencias de la librería `Arduino.h`.
- Librería `Serial` de *Processing*.

El proyecto también puede cargarse en la placa usando el el IDE oficial de *Arduino*.

2.2. Método

El desarrollo de este proyecto se divide en dos categorías importantes:

- Lectura de medidas del sensor de proximidad.
- Adapatación del código del *Pong*.

2.2.1. Lectura de medidas del sensor de proximidad

El código que se cargará en la placa *Arduino Uno R3* se encuentra el el archivo `Practica10.ino` el cual se estructura en las siguientes partes:

- Rutina `setup()`.
- Rutina `loop()`.
- Rutina `signalScaling()`

Rutina `setup()`

Esta primitiva realiza los ajustes iniciales de la placa. Se establece la velocidad de transmisión de datos de la placa a 115200 baudios.

El fabricante indica el tiempo de duración de la primera medida con datos espúrios que obtenemos del sensor (ver figura 2.2.1). El tiempo de duración esta primera lectura es de $38{,}3ms \pm 9{,}6ms$. El tiempo transcurrido entre el final de la primera lectura y el comienzo de la siguiente es como máximo de $5ms$.

Por este motivo, se establece un tiempo de espera inicial con la primitiva `delay()` de $43ms$.

```
1 void setup() {
2     Serial.begin(115200);
3     delay(43);
4 }
```

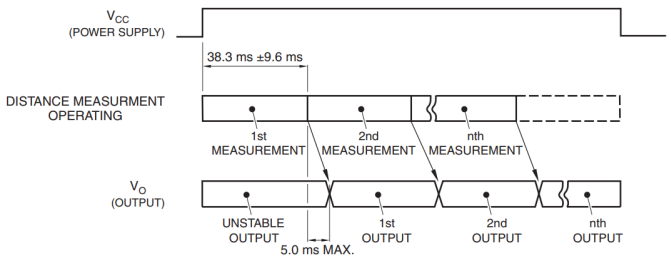

Figura 2.1: Diagrama de tiempo del sensor *GP2D12*

Rutina `loop()`

Esta rutina primitiva define el bucle principal que se ejecuta en la placa *Arduino*.

La variable `val` contiene el valor de las medidas leídas del sensor tras llamar a la rutina `signalScaling()`.

Por último, se manda el valor de `val` al puerto serial llamando a la primitva `Serial.println()` y se establece un tiempo de espera de $100ms$ entre lecturas.

```
1 void loop() {
2     int val = (int) signalScaling();
3     Serial.println(val);
4     delay(100);
5 }
```

Rutina `signalScaling()`

En esta rutina se extraen las medidas tomadas del sensor de proximidad utilizando la primitiva `analogRead()` y el pin analógico `A0`. Según la documentación oficial de *Arduino* (Arduino, s.f.-a), la resolución máxima de lectura analógica es de 10 bits para voltajes máximos de $5V$. Esta resolución permite valores en el intervalo $[0, 1023]$.

Según el fabricante, el valor voltaje máximo de salida es de 2,6V para una distancia de 10cm (ver figura 2.2.1). Esto requiere ajustar la salida que obtenemos con `analogRead()` para un voltaje máximo de 2,6. Se ha obtado también por reducir la resolución máxima original de 10 bits a 1 byte (valores en el intervalo $[0, 255]$), coincidiendo así el tamaño con el utilizado en la codificación *ASCII*.

Se aplica la siguiente fórmula para realizar las transformaciones pertinentes,

$$x' = x \cdot \frac{1023}{2{,}6} \cdot \frac{5}{1023} \cdot \frac{255}{1023}$$

siendo x' el nuevo valor de salida con los escalados pertinentes y x la medida obtenida en crudo del sensor.

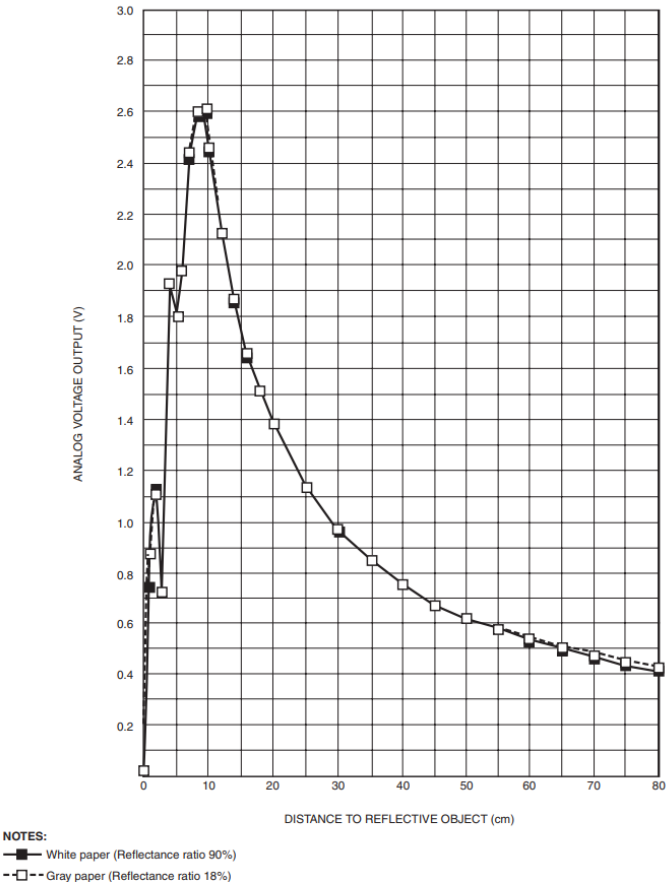

Figura 2.2: Curva de distancia del sensor a la superficie reflectante frente al voltaje de salida

```
float signalScaling() {
    return analogRead(A0) * (1023. / 2.6) * (5. / 1023.) * (255. /
    1023.);
}
```

2.2.2. Adapatación del código del *Pong*

En este apartado se describe como se integra las medidas obtenidas del sensor de proximidad en *Processing* haciendo uso de la librería `Serial`. Los pasos realizados se describen a continuación:

1. Modificación de la primitiva `setup()`.
2. Generación de eventos en el puerto serie.

Modificación de la primitiva `setup()`

Justo al final de la primitiva `setup()`, se inicializa un puerto serie de lectura a 115200 baudios (ver 2.2.1) creando un objeto `Serial`.

Una vez establecida la comunicación serial, se inicializa un buffer a partir del cual se cargan las lecturas recibidas por el canal serial hasta alcanzar el caracter de salto de línea (`0x0A`), en cuyo caso se lanza un evento que llama a la rutina `serialEvent()`.

```
        playerR.y1 = height / 2 - playerR.x2 / 2;

        playerL = new Player(this, 0, 0, playerWidth, playerHeight,
     playerMargin, playerSpeed, playerColor);
        playerL.x1 = playerL.x2 + playerL.margin;
        playerL.y1 = height / 2 - playerL.x2 / 2;

        ball = new Ball(this, ballPos, diameter, ballSpeed,
    ballSpeed, 1, 1, ballColor);
        ball.spawn();

        hitWallSound = new SoundFile(this, "res/ping_pong_8bit_plop
    .wav");
        hitPaddleSound = new SoundFile(this, "res/
    ping_pong_8bit_beeep.wav");
        pointSound = new SoundFile(this, "res/
    ping_pong_8bit_peeeeeep.wav");

        myPort = new Serial(this, Serial.list()[0], 115200);
        myPort.bufferUntil('\n');
    }
```

En esta rutina, se lee la *string* obtenida por el puerto serie y siempre y cuando lo que se reciba sea un número, se ajusta el valor de la componente vertical de la raqueta de la izquierda a un valor entre 0 y el alto máximo de la ventana. Este mapeo entre el dato obtenido por el puerto serie y la coordenada Y que le corresponde a la raqueta de la izquierda se consigue llamando a la primitiva `map()` de *Processing*.

```
    public void serialEvent(Serial p) {
        String s = p.readString();
        if(s != null) {
            s = s.replace("\r\n", "");
            if (s.matches("(\\d+)")) playerR.y1 = (int) map((float)
     Integer.parseInt(s), 0.f, 255.f, 0.f, (float) HEIGHT);
        }
    }
```

Conclusiones

Hemos observado cómo es posible establecer una comunicación entre la información obtenida de un sensor de proximidad y un computador utilizando la comunicación serial para ello.

La integración de lectura del puerto serie con *Processing* abre las puertas a un abanico enorme de posibilidades para utilizar cualquier sensor hardware para interaccionar directamente con el ordenador a partir de una interfaz gráfica que nosotros diseñemos.

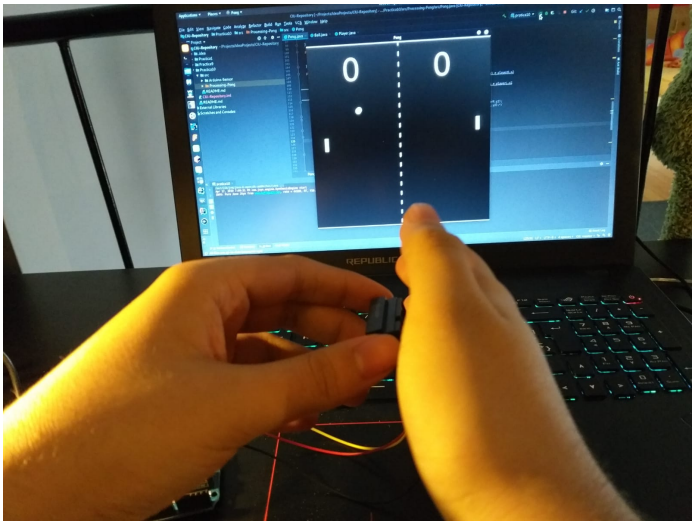

Figura 3.1: Integración de un sensor de proximidad con *Processing*

Referencias

Arduino. (s.f.-a). *analogRead().* https://www.arduino.cc/reference/en/language/functions/analog-io/analogread. Autor. (Accessed: 2019-04-17)

Arduino. (s.f.-b). *Language Reference.* https://www.arduino.cc/reference/en/. Autor. (Accessed: 2019-04-17)

Medina Medina, D. A. (s.f.). *Repositorio CIU - Practica 10.* https://github.com/david00medina/CIU-Repositorio/tree/master/Practica10. Autor (Accessed: 2019-04-17)