ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Curso académico 2010/2011

Proyecto de fin de carrera

DISPOSITIVO ELECTRÓNICO E INFORMÁTICO PARA FOTOGRAFÍA DE ALTA VELOCIDAD

Autor: Alberto Acuña Gallego

Tutor: Alexandre Wagemakers

Resumen

El proyecto que se presenta a continuación permite la realización de fotografías a alta velocidad con un equipo básico de fotografía y con la ayuda de diversos sensores y de un software diseñado para la realización de dicha tarea.

La manera de realizarlo es un poco diferente a la habitual. En lugar de disparar la cámara en el momento deseado, se toma la foto con una velocidad de disparo muy baja. Posteriormente, cuando el evento es detectado, el flash se dispara. Queda inmortalizado en la cámara únicamente el momento en el cual se disparó el flash, que era precisamente el momento que deseábamos.

Para ello se utiliza como base una placa Arduino la cual es la encargada de decidir en que momento debe de ser disparado el flash o la cámara. A su vez, también se conecta el equipo con el ordenador, que permitirá al usuario configurar los diferentes valores, así como mostrar la información necesaria sobre el proceso de realización de la fotografía.

El software tiene dos partes. Por una parte tenemos el software para la placa Arduino, el cual, en función de la información recibida por el ordenador o por los diversos sensores actuará de una manera u otra. Esa parte está programada en el propio lenguaje de la placa que tiene el mismo nombre, Arduino, el cual es bastante similar a C. La otra parte de programación es la encargada de interactuar con el usuario a través del ordenador, lo cual fue programado en java, incluyendo una interfaz gráfica desarrollada en Netbeans (versión 6.9.1).

Los resultados de las fotografías son bastante impresionantes, quedando plasmado en la pantalla de la cámara momentos que son imposibles de percibir para el ojo humano, es cómo si el tiempo se hubiese parado. Esto puede permitir el estudio de cómo se deforma un objeto al ser golpeado, cómo se produce la explosión de un globo, etc.

Índice

1.- Introducción

La fotografía de alta velocidad para obtener buenas tomas y, a su vez, buena calidad era hasta finales del siglo pasado algo imposible de conseguir. Actualmente, con los avances en cámaras digitales (tanto de vídeo como de fotografía) y de la electrónica digital hacen que cada vez esto sea más sencillo.

En estos momentos, la mayoría de las soluciones para obtener imágenes de alta velocidad suelen tener una pega principal y es el elevado coste que tienen, ya que una de las maneras más extendidas suele ser la grabación a alta velocidad. Posteriormente se obtiene el fotograma de la grabación que se desea. Esto es muy útil si uno quiere una secuencia, pero si lo que interesa es únicamente un instante (el momento del impacto, el momento que atraviesa cierto punto, etc) no merece la pena.

Este proyecto lo que nos permite es realizar fotografías de alta velocidad por un precio bajo, aproximadamente 50€, todo ello acompañado de una simple interfaz gráfica realizada en java, la cual nos permitirá tanto saber el valor de los sensores como ajustar los valores límites en los cuales el flash será disparado, así como el retraso que se desea.

1.1.- Funcionamiento general

Para entender el funcionamiento este proyecto, lo más importante es comprender en primer lugar el mecanismo de una cámara fotográfica. Lo que una cámara fotográfica capta es la luz que recibe el sensor (o el carrete en el caso de cámaras antiguas). La cantidad de luz que recibe varía en función de tres elementos.

- El primero, evidentemente es la luz exterior. Cuanta más luz haya, más luz va a captar la cámara. Por esa razón, es mucho más sencillo realizar fotos de día que por la noche.

- En segundo lugar está la velocidad de disparo. Esta velocidad es básicamente el tiempo que el sensor permanece expuesto a la luz. Como es lógico, cuanto más tiempo esté expuesto, más luz podrá captar.

- Finalmente está el diafragma de la cámara. Esto lo que permite es permitir que pase más o menos luz a través del objetivo, cerrándose para permitir el paso de

menos luz y abriéndose para permitir pasar mas luz.

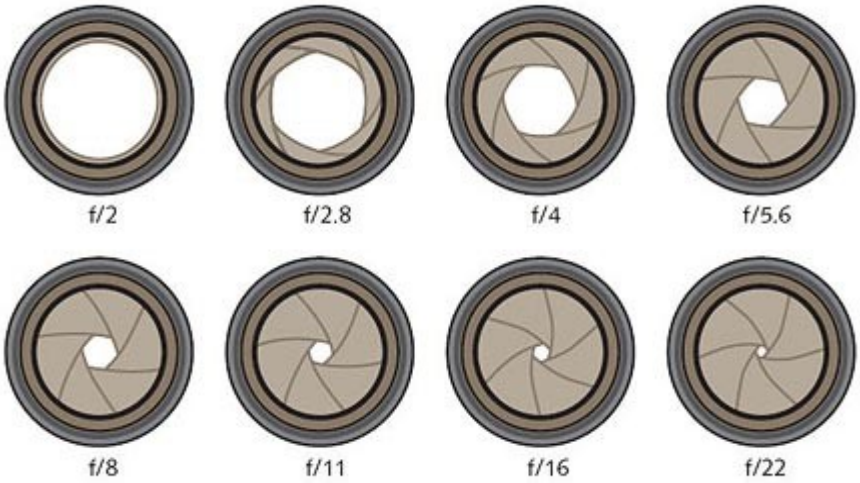


Figura 1: *Diferentes aperturas del diafragma de la cámara*

Teniendo en cuenta esos tres elementos, y sabiendo además que una cámara fotográfica, en el mejor de los casos tardará un mínimo de 40ms en comenzar a realizar la foto (tiempo que tarda en levantar el espejo y abrir el obturador la cámara), la mejor solución para realizar fotografías de alta velocidad consiste en dejar el diafragma bastante cerrado (para que entre la menor cantidad de luz), dejar el sensor expuesto en todo momento (velocidad de disparo baja o el modo BULB) y finalmente realizar la fotografía en un lugar oscuro.

Con esos elementos, al no llegar prácticamente nada de luz al sensor, la cámara no captaría ninguna imagen. Eso será así hasta que el evento que se espera (una explosión de un globo, el paso de un objeto por delante de algún sensor, etc.) haga que el flash sea disparado, iluminando la escena justo en el momento deseado. en ese instante será lo único que capte la cámara y así obtendríamos la fijada del evento.

1.2.- Método de trabajo

Para el desarrollo del proyecto se siguieron una serie de pasos en el orden siguiente:

1.-Estudio previo. En esta fase se decidió el modo de desarrollo del proyecto, como por ejemplo el tipo de placa Arduino a utilizar, los componentes necesarios y el tipo de sensores que iba a manejar.

2.- Desarrollo del hardware. Una vez que ya se habían decidido los componentes a utilizar, fueron conectados uno a uno, tomando cada sensor como un módulo individual.

3.- Prueba del hardware. Cuando ya se conectaron todos, se fue probando que el hardware funcionaba correctamente y que los componentes estaban correctamente conectados.

4.- Desarrollo del software de Arduino. Con todos los componentes ya conectados, incluimos algo de programación al Arduino, para que fuera comunicandose con los sensores. Al no haber desarrollado ningún programa para controlar Arduino, se controlaba a través del propio entorno de desarrollo de Arduino.

5.- Decisión de los comandos a detectar. En este punto se desarrolló los comando y la estructura de los mismos que debía detectar/enviar la placa Arduino y el programa a desarrollar en java.

6.- Desarrollo del software para PC. Con todos los comandos ya decididos, comenzamos con el desarrollo del software para el ordenador, incluyendo su interfaz gráfica.

7.- Inclusión de los comandos en la placa Arduino. Se finaliza de incluir todos los comandos a detectar y las operaciones que debe de hacer al recibir cada orden.

8.- Fase de pruebas. Con todo terminado, comenzamos las pruebas, así como la resolución de los errores encontrados en esta fase.

2.- Requisitos

Los requisitos, sin contar los elementos electrónicos necesarios para realizar la parte hardware los cuales se detallarán más adelante, son los siguientes:

- Ordenador con Windows:

A pesar de haber utilizado java para realizar el proyecto, es necesario que posea una versión de Windows ya que las conexiones con el puerto serie son diferentes en Windows y Linux.

Además de ello, se requiere que el ordenador cuente con al menos un puerto USB y que cuente con una versión java de 32bits. Eso es debido a que para realizar la conexión con la placa Arduino, utilizo la librería SimpleSerial, la cual no funciona a 64bits.

- Cámara:

Sirve cualquier cámara fotográfica (no importa que sea digital o analógica) siempre y cuando esta cuente con controles manuales (al menos que permita controlar la velocidad de disparo).

Si se desea usar el modo automático que se incluye en el proyecto para el disparo de la cámara, es necesario que cuente con conexión para disparador remoto, así como su cable (con tres conexiones) para ser conectado al circuito.

- Flash:

Cualquier tipo de flash que pueda ser disparado haciendo cortocircuito sirve, esto incluye tanto los flashes profesionales cómo los flashes incluidos en las cámaras analógicas compactas de un único uso.

3.- Entorno de desarrollo del software

3.1.- Entorno de trabajo

Para realizar el software de este proyecto, se han utilizado dos entornos diferentes de programación.

Por un lado está la parte del desarrollo del código para la placa Arduino, el cual ha sido desarrollado con el propio software de Arduino en su versión 0022. Este entorno, además de ser muy simple, nos permite las operaciones básicas como la de compilar código y enviarlo a Arduino para que lo ejecute. Además, proporciona una ventana simple de comunicación serial con Arduino, la cual permite probar las conexiones vía el puerto serie antes de programar la aplicación de interfaz ordenador/Arduino.

Posteriormente, para la realización del software para PC, fue utilizado NetBeans. Este programa, desarrollado por Oracle, permite programar en java y nos facilita un entorno de desarrollo bastante completo, el cual nos incluye, entre otras muchas herramientas, un creador de interfaces de usuario bastante sencillo, permitiendo crearla de una manera simple.

3.2.- Entorno de ejecución

A pesar de que java es un lenguaje de programación universal, es decir, que puede ejecutarse en cualquier máquina que tenga java instalado sin importar donde se haya compilado, en este proyecto, sólo es posible ejecutarlo en entornos de windows y con una versión instalada de java de 32bits. Esto es debido a que la conexión se realiza a través de SimpleSerial que se trata de una interfaz para realizar las conexiones a través del puerto serial la cual sólo soporta versiones de java con las características anteriormente especificadas.

4.- Elementos del proyecto

La realización de este proyecto puede dividirse en dos partes. Por un lado estaría la parte de componentes utilizados, lo que formarían la parte hardware del sistema y, por otro lado, estaría la parte software, la cual a su vez está dividida en la parte del ordenador (programado en java) y la parte de Arduino (programado en su propio lenguaje, similar a C).

4.1.-Elementos electrónicos

Los elementos electrónicos se podrían dividir en un grupo. Por un lado estaría la placa Arduino, la cual se encarga de leer los datos y actuar de una manera u otra en función de esos datos recibidos (tanto por parte del ordenador cómo por parte de los sensores). La otra parte serían los componentes que forman los diferentes sensores. En función de lo que midan, la placa Arduino realizará una acción adaptada.

4.1.1.- La placa Arduino

El principal elemento en este caso sería la placa Arduino. Dicha placa controla absolutamente todo, desde el momento en el cual va a disparar la cámara, el flash, detectar los diferentes valores de los sensores y además es el encargado de comunicarse con el ordenador.

En el mercado existen varios tipos de placa Arduino, que en función de las necesidades vendrá mejor una u otra. Existe por ejemplo la placa Arduino Mega2560, Arduino Uno, Arduino Nano, etc

Figura 2: *Placa Arduino utilizada en el proyecto*

En este caso usamos la placa Arduino Uno ya que para la realización del proyecto era el elemento más económico y cumplía de sobra las necesidades, tanto por capacidad, de 32kb (la cual se usa para guardar el código) cómo por procesador (Atmega328). El precio de dicha placa es de 24€. Además de todo ello, cuenta con 14 pins digitales (que pueden ser de entrada o de salida) y 6 pins de entrada analógicos, que permite la lectura de los datos analógicos. También un puerto USB, el cual puede servir tanto cómo método de alimentación como para enviar datos y una toma para fuente de alimentación.

Las características completas son las siguientes

Controlador	ATmega328
Voltaje con el que trabaja	5V
Voltaje de entrada recomendado	7-12V
Límites del voltaje de entrada	6-20V
Pins digitales de entrada/salida	14 (6 de ellos con salida PWM)
Pins de entrada analógicos	6
Corriente continua para cada pin de entrada/salida	40mA

Corriente continua para los pins de 3.3V	50mA
Memoria flash	32KB (de ATmega328), de los cuales 0.5KB son usados por el sistema de arranque
SRAM	2KB (de ATmega328)
EEPROM	1KB (de ATmega328)
Velocidad del reloj	16MHz

Tabla 1: *Características de Arduino Uno. Datos sacados de http://arduino.cc/en/Main/ArduinoBoardUno*

4.1.2.- Resto de componentes

La lista completa de componentes se detalla a continuación, así como su precio y la cantidad de ellos utilizados en cada parte.

Componente	Cantidad	Precio por unidad
Barrera láser		
Láser rojo 5mW	x1	US$3.95 (2.8€ aprox)
Resistencia 390Ohm	x1	0.07€
Fotorresistencia	x1	0.50€
Sensor de impacto		
Altavoz piezoeléctrico	x1	1.50€
Resistencia 10KOhms	x1	0.07€

Sensor de sonido		
Micrófono electret	x1	1.50€
Resistencia 10KOhms	x1	0.07€
Condensador 10nF	x1	0.80€
Condensador 10µF	x1	0.20€
Amplificador LM386N-3	x1	1.50€
Barrera infrarroja		
Led infrarrojo	x1	0.70€
Receptor infrarrojo	x1	1.00 €
Resistencia 82 Ohms	x1	0.07€
Resistencia 10KOhms	x1	0.07€
Otros componentes		
Placa de inserción	x1	10,00 €
1 metro de cable	x5 (diferentes colores)	0.30€
Tiristor BRY55-4	x2	1.50€
Arduino Uno	x1	24,00 €

Tabla 2: *Lista de componentes utilizados*

Con estos componentes ya es posible de hacer funcionar el proyecto de una manera básica, aunque a parte, se puede añadir el cable para conectar la cámara (no se incluye dado que el cable depende del modelo de cámara que se utilice)

4.2.- Descripción informática

Cómo ya se ha ido comentando anteriormente, hay tanto una parte de programación para el ordenador, realizada en java, y otra para la placa Arduino, realizado en su propio lenguaje de programación.

4.2.1.- Programación Arduino

Esta parte, a pesar de ser la más sencilla de las dos, es la más importante dado que dependiendo del buen funcionamiento de esta parte permitirá o no que el proyecto cumpla con los objetivos pedidos.

.- Especificación

Es muy importante tener en todo momento presente que es lo que se quiere conseguir en el proyecto. Por ese motivo, en la parte de especificación se enumera que es lo que debe de hacer y de que manera lo debe de hacer. Además, nos enfrentamos en este caso a que es una placa Arduino, tiene unos recursos limitados (tanto de memoria como de procesador), lo cual hace esta parte más importante aún.

- La respuesta desde que el evento que se espera hasta que el flash es disparado debe de ser rápida, prácticamente inmediata.

- Debe de actuar en función de la solicitud del usuario

- Se comunicará con el usuario a través del puerto serie y los comandos deben de ser lo más simples posibles para mejorar la velocidad y simplicidad.

- Debe de permitir al menos 4 sensores (sonido, impacto, barrera láser y barrera infrarroja).

- El usuario debe poder decidir el sensor a detectar, el límite que se desea y el retraso.

- El usuario podrá cancelar en todo momento.

- Una vez compilado, el código no puede ocupar más de 32KB (limitación de memoria flash de Arduino Uno) ni utilizar más de 2KB de memoria (SRAM).

- Debe ser flexible con el tipo de cámara, con el tipo de flash, así como con el tipo de sensor a utilizar (que no dependa de un modelo concreto de sensor)

- Debe de ser autónomo a la hora de realizar las fotografías. Una vez que el usuario da la orden y los datos de realizar la fotografía, es él que debe decidir el momento de realización de la foto.

.- Implementación.

Para la programación de la placa se ha utilizado el software Arduino 0022, que ofrece un Entorno de desarrollo para su placa. Las funciones que utilizamos en el código de Arduino son las siguientes:

1.- **Función setup**: Es la primera de las dos funciones obligatorias para el correcto funcionamiento de la placa Arduino. Todo programa que se realice con Arduino debe de incluir esta función ya que es la encargada de configurar los diferentes pins digitales para que sean de entrada o salida. Además, también debemos incluir los baudios a los que vamos a utilizar el puerto serial (en este caso a 9600)

```
void setup()
{
    Serial.begin(9600);
    pinMode(FLASH, OUTPUT);
    pinMode(CAMARA, OUTPUT);
}
```

2.- **Función loop**: Es la otra función necesaria para hacer funcionar cualquier programa en Arduino. Como su nombre indica, esta función realiza un bucle infinito. Cuando le llega corriente a la placa Arduino comienza a ejecutarse desde el principio y, si llega al final, volvería a empezar, y así continuamente hasta que se desconecte la placa. En este proyecto, esa función lee continuamente los datos que recibe del puerto serie y, en caso de que reciba algo, comienza a realizar la tarea demandada. El funcionamiento de esta función se verá más detenidamente en el apartado de conexión Arduino-PC.

3.- **Función hacerFotos**: Esta función no recibe ningún valor ni devuelve ningún valor. Cuando se ejecuta la función, lo primero que realiza es la lectura del puerto serie para ver el resto de datos que se necesitan, tales como el retraso, número de fotos, valor al que se disparará el flash, etc. A su vez, esta función llama a la función hacerFoto tantas veces cómo número de fotos se haya pedido hacer (salvo que el usuario lo cancele antes)

La siguiente parte de código muestra el funcionamiento del bucle principal de la

función, el cual se estará ejecutando continuamente hasta que, o bien haya realizado todas las fotos que se solicitaban o bien el usuario pida finalizar la realización de fotos.

```
if (nFotos==0)
    nFotos=-1;
while(!fin){
    digitalWrite(CAMARA,HIGH);
    hacerFoto(sensor,retraso,limite,&fin);
    digitalWrite(CAMARA,LOW);
    if(nFotos>0)
        nFotos--;
    if(Serial.available()>0){
        fin=true;
        Serial.flush();
    }
    else{
        if (nFotos==0){
            fin=true;
        }
        else{
            Serial.print((byte) 1);
            delay(1000);
        }
    }
}
```

4.- **Función hacerFoto**: Se encarga de realizar una única foto. La función recibe por valor el sensor, el retraso y el valor límite, mientras que por referencia recibe el valor fin. Eso es así ya que el usuario podría cancelar en cualquier momento la acción y al terminar la función que le llamó (en este caso hacerFotos) no sabría si acabó por cancelación del usuario o por realización de la foto.

```
void hacerFoto(int sensor, int retraso, int limite, boolean *fin){
    if(sensor==LASER || sensor==IR){
        while(analogRead(sensor)>limite){
```

```
            if(Serial.available()>0){
                *fin=true;
                Serial.flush();
                break;
            }
        }
    }
{…}
    if(!*fin){
        delay(retraso);
        digitalWrite(FLASH,HIGH);
        delay(TIEMPO_FLASH);
        digitalWrite(FLASH,LOW);
    }
}
```

También, a parte de las funciones, incluye algunas constantes como por ejemplo LASER, IMPACTO, SONIDO, IR, las cuales contienen un valor numérico con el pin analógico asignado y el número de sensor (cuando se recibe datos a través del puerto serie) que le corresponde a cada uno.

Las constantes FLASH y CAMARA contienen el número de pin digital asignado para activar el flash o la cámara mientras que TIEMPO_FLASH es el tiempo aproximado que debe de estar el flash en cortocircuito para que sea disparado correctamente.

4.2.2.- Programación Java

Esta parte es la encargada de configurar todo lo necesario para que la placa Arduino pueda realizar todo su trabajo. Además, es una ayuda visual para el usuario ya que permite saber en todo momento tanto las fotos realizadas/por realizar, así como saber a la hora de la configuración el valor en ese momento del sensor.

.- Especificación.

- Debe de ser simple e intuitivo.

- Tiene que contener todas las opciones de configuración necesarias para el correcto funcionamiento (valores límites, sensor a utilizar, retraso y número de fotos)

- Tiene que poder comunicarse con la placa Arduino a través del puerto serie.

- Tiene que mostrar la información más importante, ya sea el valor del sensor para poder configurar correctamente los límites o el número de fotos realizadas/por realizar por la placa Arduino.

- Tiene que poder cancelar cualquier operación en cualquier momento, volviendo a la pantalla inicial.

- Debe de ocupar el menor espacio posible y no consumir demasiados recursos del ordenador.

- Todos los datos de la placa deben de ser mostrados en tiempo real.

- Diseño / implementación

La parte de programación en java fue realizada creando dos clases principales. Por un lado está la clase InterfazGrafica que, como su nombre indica, es la encargada de generar la interfaz gráfica para interactuar con el usuario. Esta clase comunica a su vez con la clase Arduino, la cual encarga de mandar las señales a la placa Arduino a través del puerto serie. Para ello hace uso de la clase SimpleSerial que es la interfaz de comunicación con el puerto serie utilizada en este proyecto.

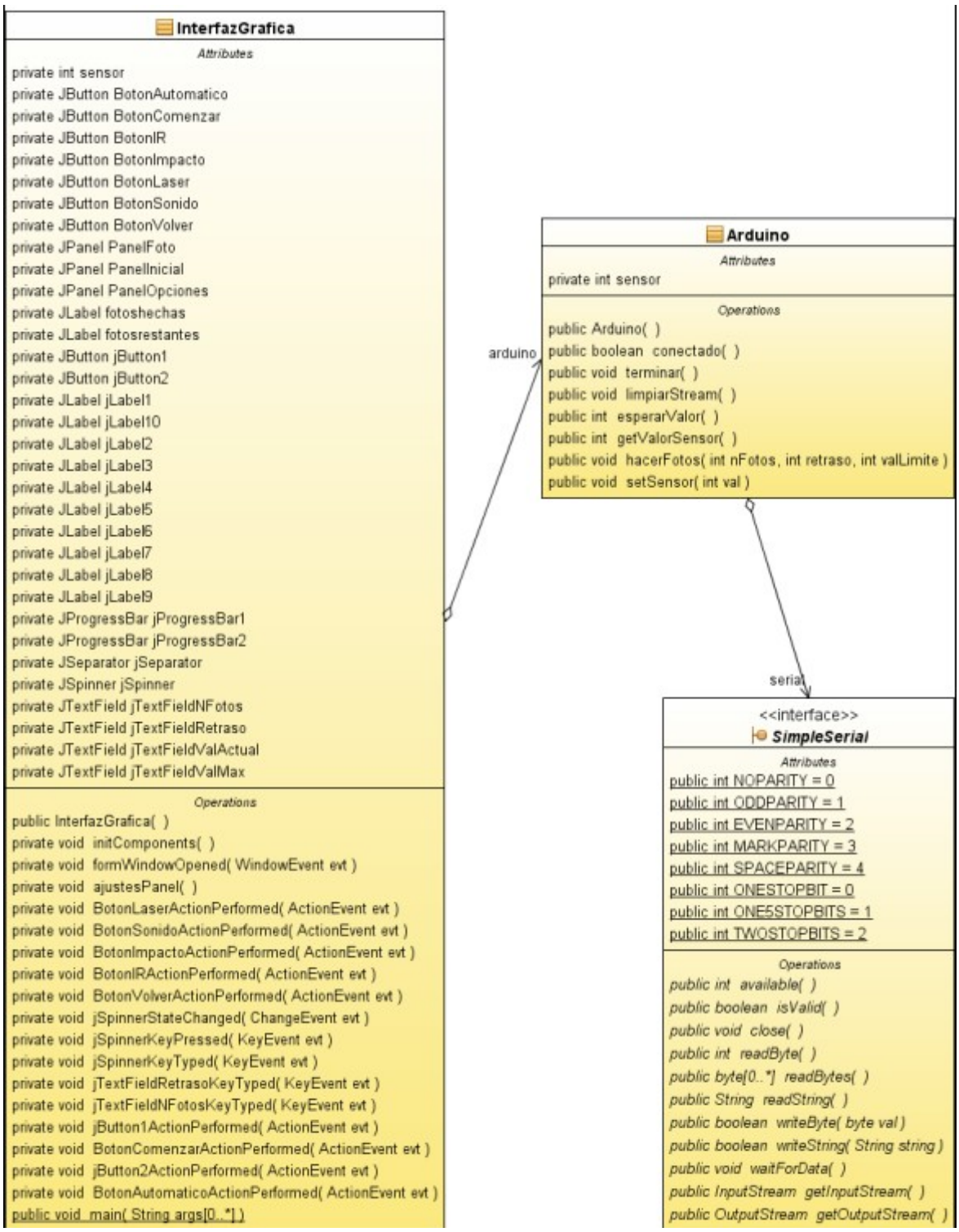


Figura 3: *Diagrama de clases*

-Clase InterfazGrafica: Esta clase, al ser la interfaz gráfica que usa el usuario, posee numerosos componentes, tales como el JButton, JLabel, JTesxtField, etc. Existen 3 tipos de JPanel diferentes, cada uno de ellos se encargará de mostrar sus elementos asociados. De esta manera, cuando por ejemplo estamos mostrando el JPanel panel inicial y se pulsa un botón, el

panel, junto con todos sus elementos asociados, quedaría oculto para que posteriormente se muestre el JPanel PanelOpciones que correspondería a la siguiente pantalla que se quería mostrar.

Esta clase InterfazGrafica contiene también numerosos métodos que son en su mayoría funciones oyentes (listeners) que esperan a que el usuario realice alguna acción para ser ejecutados, tales como pulsar un botón, escribir texto, etc.

Por ejemplo, el método ajustesPanel() se encarga de mostrar un panel u otro en función del sensor seleccionado, y de ajustar los textos.

```
private void ajustesPanel(){
    if(sensor==0){
        PanelInicial.setVisible(true);
        PanelOpciones.setVisible(false);
        PanelFoto.setVisible(false);
    }
    else{
        PanelOpciones.setVisible(true);
        PanelInicial.setVisible(false);
    }
    switch(sensor){
        case 0:this.setTitle("Selecciona sensor");
            if (t!=null)
                t.detener();
            break;
        case 1:jLabel2.setText("Valor mínimo");
            jLabel6.setText("Valor mínimo detectado:");
            t=new actualizaBarra();
            t.start();
            this.setTitle("Sensor láser");
            break;
        {…}
    }
}
```

A su vez, también contiene otras dos subclases, una se llama fotosRealizadas y la otra actualizaBarra. Ambas heredan de la clase Thread, lo cual permite ejecutarse de manera concurrente, sin bloquear el programa principal.

La clase **fotosRealizadas** permite que mientras está en el estado de realizar fotografías, esté continuamente esperando a recibir una señal, la cual podría ser un 1, lo cual indicaría que realizó la fotografía por lo que actualiza la interfaz indicando que se ha realizado una foto, o también podría valer 0, lo que indicaría que se ha terminado de realizar las fotos y puede volver al panel inicial. Sus métodos son run() (método que permite comenzar la ejecución del hilo) y detener (método que permite salir del bucle que se inició con el método run()). A su vez, contiene dos atributos, uno se llama fin, el cual es un valor booleano que cuando se pone a true indica que debe finalizar, y valor, el cual es un entero que contiene el valor recibido a través del puerto serie.

```
public void run() {
    while(!fin){
        valor=arduino.esperarValor();
        if (valor==0){
            fin=true;
        }
        else if(valor==1){
            fotoshechas.setText(""+(Integer.parseInt(fotoshechas.getText())+1));
            if(!fotosrestantes.getText().equalsIgnoreCase("Infinitas")){
                int res=Integer.parseInt(fotosrestantes.getText())-1;
                fotosrestantes.setText(""+res);
                if (res<=0){
                    fin=true;
                }
            }
        }
    }
}
```

La clase **actualizaBarra** actualiza constantemente el valor del sensor de Arduino, permitiéndonos saber en todo momento en que nivel se encuentra para así poder ajustar nuestro valor límite. Contiene cuatro métodos:

- run(): Permite comenzar a ejecutar el hilo,
- detener(): permite salir del bucle iniciado en run(),
- finalizado(): Devuelve true si ha terminado de ejecutarse y false si todavía sigue ejecutándose.
- resetValmin: Resetea el valor mínimo detectado hasta ese momento.

También contiene tres atributos:

- valorLimite: Se representa con un entero el valor máximo o mínimo (dependiendo del sensor seleccionado) detectado hasta ese momento,
- fin: Es un valor booleano que indica si debe finalizar o no el bucle
- finalizado: Es un valor booleano que se pone a true cuando el bucle ha terminado de ejecutarse, actuando de semáforo para evitar problemas de concurrencia como podría ocurrir en caso de que dos procesos diferentes enviaran una solicitud a Arduino, ya que posteriormente no se sabría de quien es el valor recibido.

```
public void run() {
    {…}
     while (!fin) {
        i=arduino.getValorSensor();
        jProgressBar1.setValue(i);
        jTextFieldValActual.setText(i+"");
        if (sensor==2 || sensor==3){
            if (valorLimite<i){
                valorLimite=i;
                jTextFieldValMax.setText(i+"");
            }
        }
        else{
            if (valorLimite>i){
```

```
                valorLimite=i;
                jTextFieldValMax.setText(i+"");
            }
        }
    }
}
```

-Clase Arduino: Esta clase es la encargada de realizar la comunicación con la placa Arduino. Es capaz de leer los datos del puerto serie (con ayuda de la interfaz SimpleSerial), así como enviar los datos necesarios. También es la clase que contiene todo el protocolo, es decir, la estructura que debe de seguir para poder realizar fotos, obtener el valor del sensor, etc.

Los métodos más importantes de dicha clase son dos, getValorSensor y hacerFotos, especialmente este último que es el que indica a la placa Arduino como debe de hacer la foto..

El primer método (getValorSensor), pregunta a la placa Arduino el valor del sensor, para ello simplemente envía el número del sensor del que desea saber el valor, y espera a recibir una respuesta por parte de la placa Arduino, el cual será el valor que se devuelva. Se usa para poder obtener en tiempo real la información del sensor seleccionado y saber que valores se deben aplicar. En ese método, y debido a que Java controla los valores en bytes como enteros con signo (lo cual nos daría valores entre -128 y +128), hacemos también una pequeña conversión para que el valor del primer bit no cuente como signo, si no como valor.

```
public int getValorSensor(){
    serial.writeByte((byte) sensor);
    serial.waitForData();
    int val=serial.readByte();
     if (val<0){ //Aquí comprueba si el valor es menor que cero, en ese caso, evitamos que trate el primer bit como signo.
        val=(256+val);
    }
    return val;
}
```

El método hacerFotos es el más importante. Envía a la placa Arduino un total de 6 bytes y no recibe ningún tipo de respuesta ni devuelve ningún valor. La placa Arduino, únicamente con esos 6 bytes que se le envían es capaz de realizar todo los tipos de fotografía que tenemos disponibles en este proyecto, ya que incluye tanto el sensor que debe de atender, el valor límite, el retraso desde que el evento es detectado y salta el flash, etc. El formato de envío de datos se explicará más detenidamente en la sección “conexión Arduino-PC”

El resto de métodos no son tan importantes, aunque evidentemente también tienen su utilidad. El método setSensor() configura el atributo sensor que es un entero con el número de sensor que le corresponde, el cual es posteriormente utilizado por getValorSensor() para conocer el valor del sensor seleccionado. El método esperarValor() se queda bloqueado en ese punto hasta recibir algún byte a través del puerto serie, limpiarStream() elimina todos los datos que fueron recibidos hasta ese momento en el puerto serie y terminar() indica a Arduino que debe terminar la acción que está realizando (en el caso de este proyecto, la única acción que se puede anular es el proceso de realizar fotos)

5.- Conexión Arduino-PC

La conexión entre el PC y Arduino se realizó de una manera simple pero efectiva, de tal manera que cubriera todos los posibles casos de uso y a su vez fuera fácil de desarrollar.

Para ello, lo que se hizo fue dividir las señales en dos partes, por un lado aquellas señales que se realizan antes de comenzar a hacer las fotos (cuando se pide valores del sensor), es decir, cuando no ha entrado en ninguna función y está en la función loop() inicial, y por otro el momento en el que se va a comenzar a realizar las fotografías y se están realizando.

Para el primer caso, es bastante simple. A cada sensor se le dio un valor numérico entero, de la siguiente forma

Sensor	Comando correspondiente
Láser	1
Impacto	2
Sonido	3
Infrarrojo	4

Tabla 3: *Relación sensor/número asignado*

Además, a parte de ser el valor que transmitimos por el puerto serie para hablar de ese sensor a Arduino, también coincide con el valor del pin analógico al cual va conectado la salida del sensor. Por tanto, estando la placa Arduino en el modo inicial (esperando algún dato del serial), si el ordenador desea conocer el valor del láser, únicamente enviaría "1" y Arduino automáticamente devolvería el valor que lee del pin analógico número 1 que corresponde a la fotoresistencia.

Aquí se muestra un pequeño trozo del código arduino que se encarga de esa parte. Tanto LASER como SONIDO son constantes con el valor de su pin analógico (1 y 2 respectivamente)

```
if(Serial.available() > 0){ //Si hay algún byte disponible entramos, si no esperamos.
    switch(Serial.read()){ //Obtenemos el valor que hay en el puerto serie
        case LASER:
            val = analogRead(LASER);
            Serial.print((byte) (val/4));
            break;
        case IMPACTO:
            val = analogRead(IMPACTO);
            Serial.print((byte) (val/4));
            break;
    {…}
    }
}
```

La razón de dividir el valor que se va a enviar entre 4 es debido a las características de Arduino y del puerto Serie. Arduino lee los valores del puerto entre 0 y 1023, mientras que el puerto serie, al enviar byte por byte los datos acepta valores entre 0 y 255. Como realmente es

un margen de error más que aceptable para lo que se busca, no es necesario mayor precisión. Se envía únicamente un byte y así se pueden actualizar los datos más rápidamente.

La segunda parte correspondería al momento en el que se desea realizar las fotos y se están realizando. Si como se puede comprobar anteriormente, antes de comenzar a realizar las fotografías el ordenador es el que manda las órdenes mientras que la placa Arduino actúa de una manera pasiva enviando los datos según lo que le pidan, una vez que comience a realizar las fotografías cambiará de tal manera que la placa Arduino será la que controle todo mientras que el ordenador únicamente espera las órdenes de la placa Arduino (salvo en el caso que se cancele).

A la hora de realizar las fotos, es necesario que el ordenador no tenga ningún poder de decisión, principalmente por el motivo de que eso ralentizaría mucho el proceso por la comunicación con el puerto serie. Por ello, cuando se quiere comenzar a realizar las fotos, el ordenador debe de enviar a la placa toda la información que el usuario había realizado previamente en el ordenador de tal manera que la placa pueda ser 100% independiente. El sistema para enviar los datos fue el siguiente (cada X representa un byte)

255	X	X	X	X	X
Indica modo foto	Valor sensor	Número de fotos a realizar	Retraso desde que el sensor pasa el limite		Límite del sensor
	Entre 1 y 4	0 = infinitas			

Tabla 4: *Trama del comando para realizar foto*

Con estos seis bytes enviados Arduino ya conoce absolutamente toda la información necesaria y los parámetros configurados por el usuario.

Cómo se puede observar, el cuarto y quinto byte corresponden al retraso. La razón de utilizar dos bytes es que en este caso era necesaria bastante precisión puesto que se trata de los milisegundos de retraso con los que se realizará la fotografía. Al quererlo expresar en milisegundos, un único byte nos daría un retraso máximo de 255ms, es decir, 0.255 segundos. Al ser esto insuficiente para ciertas fotografías se optó por añadir un segundo byte, lo que permite un retraso de hasta 65025ms, o lo que es lo mismo, unos 65 segundos, más que suficiente para este tipo de fotografía. Por ello, el primero de esos dos bytes lo que contiene es

la división entera del valor entre 255 y el segundo byte contiene el módulo del valor entre 255. Por tanto, el código de envío en java quedaría así.

```
public void hacerFotos(int nFotos, int retraso, int valLimite){
    serial.writeByte((byte) 255);
    serial.writeByte((byte) sensor);
    serial.writeByte((byte) nFotos);
    serial.writeByte((byte) (retraso/255));
    serial.writeByte((byte) (retraso%255));
    serial.writeByte((byte) valLimite);
}
```

Finalmente, una vez que ya se configuró la placa y comienza el proceso de realizar fotografías, la placa Arduino envía al ordenador un "1" en caso de que haya realizado una fotografía y un "0" en caso de que haya finalizado de realizar fotografías. Además, el ordenador en ese modo puede enviarle a la placa cualquier byte (el código en java envía un "0", pero el código de Arduino aceptaría cualquier valor) para finalizar el proceso de realización de las fotografías.

(Código Arduino)

```
while(analogRead(sensor)>limite){
    if(Serial.available()>0){
        *fin=true;
        Serial.flush();
        break;
    }
}
```

6.- La interfaz gráfica

Para hacer más simple el manejo de la placa Arduino y además poder tener un mayor control, se ha desarrollado una interfaz gráfica en java que permite tanto ver la información que recogen los sensores (muy útil a la hora de poner los valores límites) o el número de fotos realizadas o por realizar como poder configurar los valores, el número de fotos, el retraso y el sensor que se va a utilizar.

Dicha interfaz gráfica fue realizada con la ayuda del programa Netbeans, el cual incluye un creador de interfaces. El creador de interfaces crea los componentes con la ayuda de la librería de java para diseño de interfaces llamado Swing, que incluye numerosos componentes.

Cuando se ejecuta la aplicación nos encontramos en primer lugar con una pantalla que nos pide seleccionar el sensor que deseamos utilizar.

Figura 4: *Captura de pantalla de la interfaz incial*

Como se puede comprobar consta únicamente de cuatro botones que, sin importar el botón que pulses, lleva a la misma pantalla pero con una configuración distinta para cada caso.

Entre los 4 sensores, se pueden distinguir dos grupos y los sensores que estuvieran en cada grupo tendrían un comportamiento similar. En uno de los grupos estaría la barrera láser y la barrera infrarroja y en el otro el sensor de impacto y el de sonido. Lo que cada grupo tiene en común es el valor límite que, en el primer caso, para que se dispare el sensor el valor que se lea en ese momento debe de ser inferior que el del límite y en el otro grupo tiene que ser superior.

Como se comentó, sin importar a que botón se pulse se iría a la segunda pantalla que sería la misma en todos los casos pero con diferentes configuraciones. El aspecto de la segunda pantalla sería el siguiente

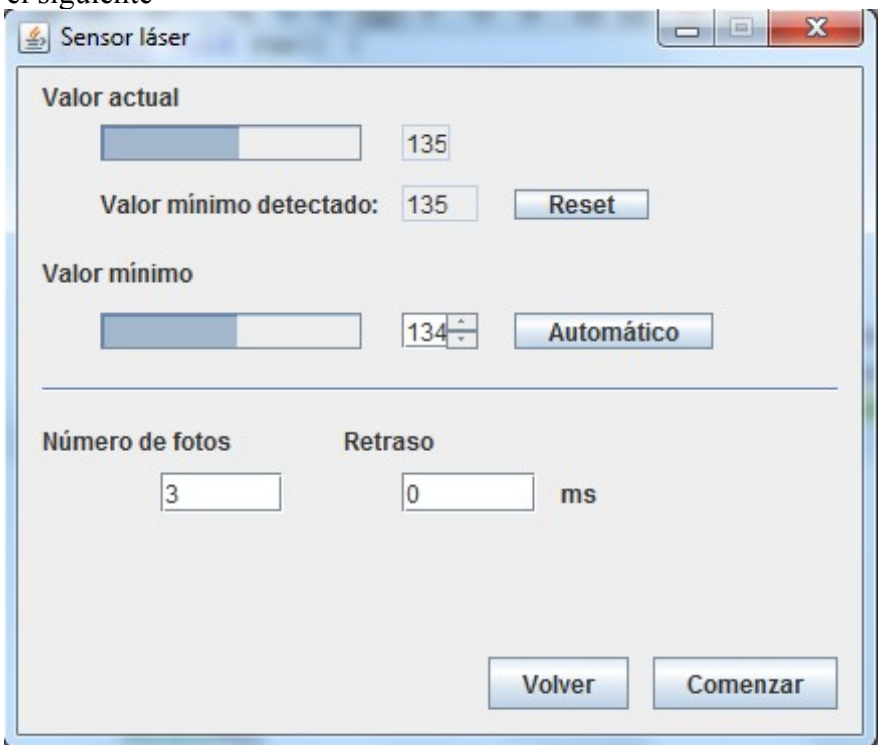


Figura 5: *Captura de pantalla del modo configuración*

En primer lugar tenemos una barra de progreso justo debajo de valor actual. Ese es el valor de la tensión que lee la placa Arduino en el pin correspondiente al sensor seleccionado (en este caso el sensor láser). Además, justo en la parte inferior nos indica cual ha sido el valor mínimo detectado hasta ese momento, junto a un botón de reset para volver a calcular dicho valor. En caso de que el sensor seleccionado hubiese sido el de sonido o el de impacto, en lugar de mostrar "valor mínimo detectado" mostraría "valor máximo detectado".

Justo debajo comienza la parte configurable por el usuario. En primer lugar tendríamos el valor mínimo (máximo en caso de ser el sensor de sonido o de impacto). Nos permite seleccionar el umbral que, en caso de ser pasado, sería disparado el flash. El botón automático permite poner dicho valor de manera automática. Su funcionamiento es bastante simple, se obtendrían tres valores del sensor, se quedaría con el más bajo de los tres (el más alto en caso de ser el sensor de sonido o de impacto) y le restaría 1. De esta manera, es simple la selección del

valor, ya que hace todo de manera automática.

Finalmente, tenemos otras dos opciones, por un lado estaría el número de fotos, el cual indica cuantas fotos desearías hacer (si se indica 0 serían ilimitadas, hasta que el usuario cancele) y por otro, el retraso, el cual indica (en milisegundos) el retraso que se desea desde que se pasa el límite hasta que el flash es disparado.

También disponemos de un par de botones para comenzar el proceso de realizar las fotos. En tal caso, iríamos a la siguiente pantalla y enviaríamos a Arduino la información correspondiente, mientras que si pulsamos el botón de volver, volveríamos a la pantalla anterior para seleccionar un nuevo sensor.

Figura 6: *Captura de pantalla del modo realizar fotos*.

En esta última pantalla indica únicamente el número de fotos realizadas hasta ese momento y las fotos que quedan por realizar. En este momento el programa en java no hace ninguna acción, únicamente se queda en espera mostrando esa pantalla mientras que un thread se está ejecutando escuchando los valores que recibe de la placa Arduino para saber si ha realizado una foto o ha terminado.

En caso de que se desee finalizar con antelación, podemos pulsar en cualquier momento sobre el botón terminar y enviará una señal a Arduino indicándoselo. Por tanto, hay dos maneras de terminar, bien por acción del usuario o bien por que se hayan realizado todas las fotografías pedidas. En ambos casos volvería de nuevo al panel inicial que permite la selección del sensor.

7.- Circuito

El circuito del proyecto fue pensado de tal manera que cada módulo fuera independiente, estando los módulos conectados entre sí únicamente a través de la placa Arduino. De esa manera, quitar por ejemplo la fotorresistencia únicamente se tendrían problemas a la hora de realizar fotos con dicho sensor, pero no afectaría, por ejemplo, si tratáramos de realizar una foto con el sensor de sonido.

-**Barrera láser**: El circuito de dicha barrera se compone de dos partes. Por un lado estaría la parte del emisor láser y por otra la de la fotorresistencia. El primero de ellos, es bastante simple ya que se decidió conectar el láser a una fuente de corriente externa (en este caso una pila de petaca de 4,5V). El motivo por el que se separó de la placa Arduino fue principalmente por comodidad ya que, al ser una fuente que estará constantemente encendido, no necesita ningún tipo de control y además de esa manera evitamos tener que cruzar cables por delante del espacio a fotografiar. Por todo ello y dado que el láser que utilizamos está construido para ser usado sin ningún tipo de resistencia, únicamente hay que conectarlo a la fuente de alimentación.

La fotorresistencia en cambio funciona de una manera diferente. La utilidad de dicha fotorresistencia no es otra que permitirnos saber la cantidad de luz que llega hasta dicho sensor. Eso se sabe en función de la tensión que mida la placa Arduino. La tensión es necesario medirla entre dos puntos, uno de ellos será tierra (GND) y el otro sería el pin analógico (en el caso del láser el pin 1). El funcionamiento de la fotorresistencia es bastante simple, cuanta más luz reciba menor será la resistencia mientras que si recibe poca luz la resistencia que causará será bastante alta. Por tanto, el esquema quedaría así:

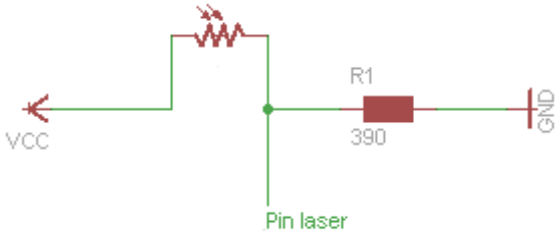


Figura 7: *Circuito del sensor láser*

Cómo se puede observar, la tensión que se mide es la tensión que cae en una resitencia que está colocada en serie con la fotorresistencia. En este caso, el valor de la resistencia es de 390Ohms, pero podría tener otros valores. Cuanto mayor sea dicho valor, mayor será la precisión con la que pueda medir (se hicieron pruebas con una resistencia de, por ejemplo, 10KOhms, dando valores de tensión mucho más altos), aunque la precisión que nos da la resistencia de 390Ohms es suficiente para la detección de la caída de un objeto.

-**Sensor de impacto**: Para poder detectar el momento en el cual un objeto golpea una superficie se utiliza un micrófono piezoeléctrico. Este elemento genera una tensión cuando vibra, ya sea por causas del sonido o bien por ser golpeado. En este caso, el esquema es algo diferente que en la fotorresistenca, pero igualmente es bastante simple.

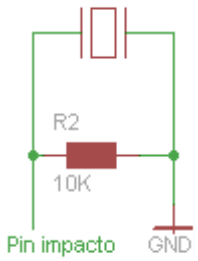


Figura 8: *Circuito sensor de impacto*

Como se puede comprobar, no tiene ningún tipo de entrada de corriente ya que, como se ha comentado, el propio micrófono piezoeléctrico es el que genera la corriente en función del impacto. Además, se coloca una resistencia en paralelo que es la que nos permite medir la caída de tensión poniendo en uno de sus extremos el pin analógico de Arduino (en este caso correspondería al pin analógico 2) y el otro extremo a tierra (GND). La elección de una resistencia de 10KOhms fue debido a que la tensión generada por el micrófono es bastante baja, por lo que una resistencia mayor nos permite mayor precisión en la lectura de los datos.

-**Sensor de sonido**: Para realizar este sensor utilizamos un sensor de sonido del tipo electret. En teoría, dado que es un micrófono del tipo electroestático, su circuito sería similar al de la fotorresistencia. El problema es que el micrófono genera muy poca variación, por lo que es prácticamente imperceptible ya que siempre tiene valores muy bajos. Es por ello que el circuito se complica un poco al tener que añadir un amplificador.

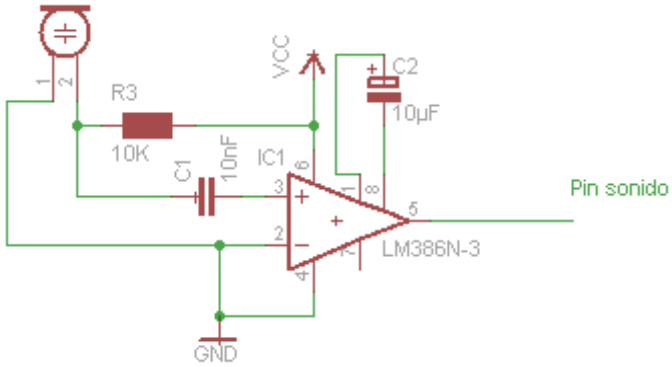
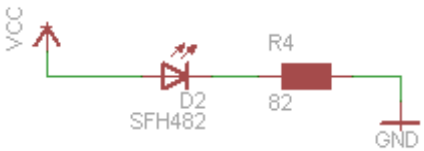


Figura 9: *Circuito del sensor de sonido*

Dicho amplificador, requiere de un amplificador operacional (en este caso utilizamos el LM386N-3), así como de dos condensadores (uno de 10µF y otro de 10nF) y una resistencia, que en este caso es de 10KOhms. De esta manera, es mucho más preciso a la hora de poder medir el sonido que capta dicho micrófono. El amplificador se pone con una configuración de comparador, es decir, sin retroalimentación.

-**Barrera infrarroja**: Al igual que en el caso de la barrera láser, esta barrera consta de dos elementos. Por un lado el emisor, que en esta ocasión se trata de un diodo led infrarrojo, y por otro lado el receptor, que se trata de un fotodiodo infrarrojo.



Circuito del emisor infrarrojo

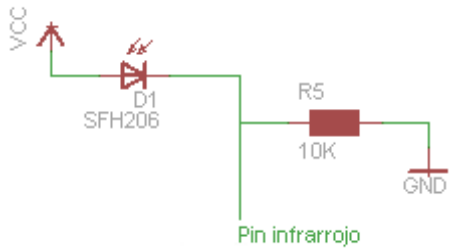


Figura 10: *Circuito del receptor infrarrojo*

El esquema cómo se puede observar es bastante similar al de la barrera láser ya que el funcionamiento es bastante similar. La única diferencia está en el led infrarrojo, el cual si que necesita una resistencia, aunque con el voltaje que se usa no es necesario que sea demasiado resistiva. Por esa razón, se optó por una resistencia de 82ohms.

El circuito del receptor también es bastante similar al de la fotorresistencia. La única diferencia es que en este caso se usó una resistencia de 10Kohms para poder medir con mayor precisión.

-Disparador cámara/flash: El funcionamiento para disparar la cámara y el flash es prácticamente el mismo. En ambos casos consiste en generar un cortocircuito que permita el paso de la electricidad de tal manera que se dispare la cámara o el flash.

En el caso de la cámara, por lo general, los conectores tienen tres partes. Dos de ellos son los encargados de, por un lado, activar el enfoque automático y, el otro, es el encargado de disparar la cámara. El tercer pin que queda es el que cuando se conecta con uno de los otros dos genera la acción bien sea de disparo de la cámara o el enfoque automático.

El flash funciona de una manera idéntica, sólo que en lugar de tener tres conectores como es el caso de la cámara tiene únicamente dos que cuando se crea el cortocircuito entre ambos se genera el disparo del flash.

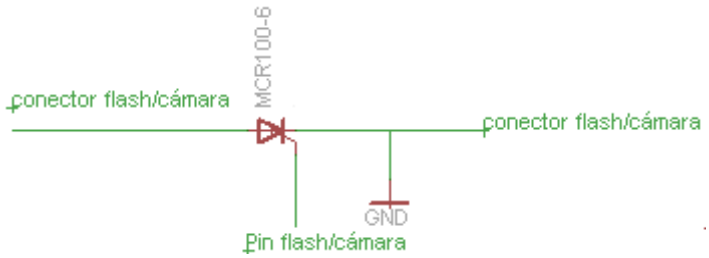


Figura 11: *Circuito de disparo del flash o de la cámara*

Como se puede ver en el esquema, únicamente se usa un tiristor para cada disparador (uno para el flash y otro para la cámara). En el ánodo del tiristor se conecta uno de los conectores (bien sea de la cámara o del flash), mientras que en el cátodo se conectaría el otro conector. A su vez, colocamos una toma de tierra en el cátodo y a la puerta conectaríamos la entrada digital que controla el disparo. Cuando en la puerta el valor digital valga 1, permite pasar corriente, lo cual activa el cortocircuito que dispara el flash o la cámara. Cuando el valor digital de la puerta vuelve a cero, los puntos que antes estaban cortocircuitados dejarán de estarlo, volviendo a estar desconectados, de tal forma que, en el caso de la cámara, dejará de hacer la foto y en el caso del flash evitará que se siga disparando.

Correspondencia nombres usados en los esquemas/puerto arduino:

Esquema	Arduino
Pin láser	Pin analógico 1
Pin impacto	Pin analógico 2
Pin sonido	Pin analógico 3
Pin infrarrojo	Pin analógico 4
Pin flash/cámara	Pin digital 0 (flash)/Pin digital 1 (cámara)

Tabla 5: *Analogía sensor/pin Arduino*

8.- Resultados

Una vez visto el funcionamiento del proyecto, es hora de ver los resultados que se pueden llegar a obtener de una manera completamente casera.

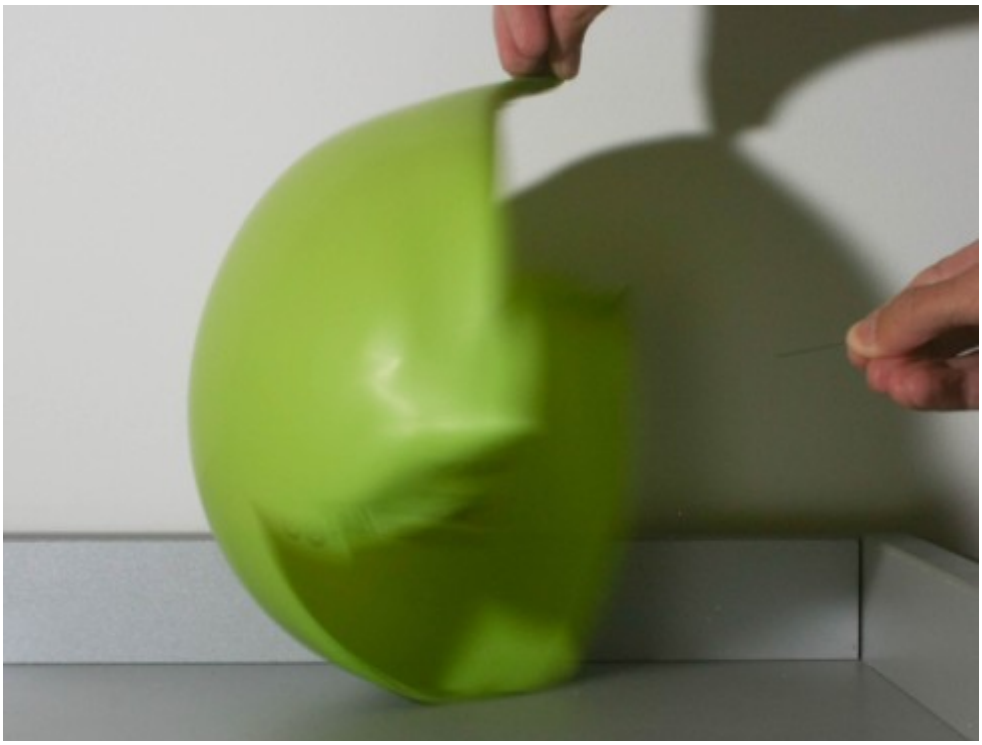


Figura 12: *Globo explotando. Sensor sonido. Retraso 0ms*

La razón por la que se ve un poco de movimiento en una parte del globo es debido a que la potencia del flash fue puesta en nivel 6 de 7. Esto provoca que, al ser mayor la luminosidad, tarda más en apagarse, por esa razón, se aprecia un poco de movimiento. Si se pone a menor potencia quedará completamente congelada la imagen.

Figura 13: *Huevo cayendo. Sensor impacto. Retraso 10ms*

Aunque con el plástico que puse para proteger la mesa queda un poco difuminada la clara del huevo, se puede apreciar la salpicadura que produce al romperse apenas 10ms después del impacto.

Figura 14: *Palillos cayendo. Sensor barrera láser. Retraso 75ms*

En esta imagen, a pesar de no ser excesivamente espectacular, se puede ver lo que ocurre una vez pasados 75ms desde que el láser detectó su caída.

Figura 15: *Palillos cayendo. Sensor barrera láser. Retraso 0ms*

A pesar de que esta imagen no contiene nada espectacular ni es probablemente un resultado que se busque, se incluye para mostrar la velocidad de la barrera láser que, cómo se puede comprobar, el flash es disparado justo en el momento en el que el palillo está en la barrera láser.

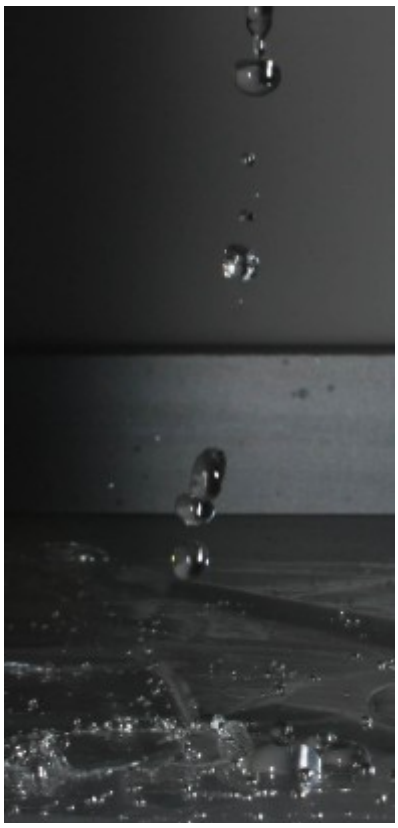


Figura 16: *Gotas de agua. Sensor barrera infrarroja. Retraso 75ms*.

Dado que la barrera infrarroja se crea para detectar eventos con mayor precisión que el láser, un ejemplo típico de esta barrera suelen ser gotas de agua cayendo. Además, especialmente con el agua es recomendable usar la barrera infrarroja para evitar los posibles reflejos que puede producir el láser sobre las gotas de agua.

9.- Conclusiones

Con la realización de este proyecto ahora es posible realizar fotografías de alta velocidad de una manera sencilla, con la ayuda de cuatro sensores diferentes los cuales cubren prácticamente todas las necesidades para este tipo de fotografía.

Gracias al trabajo de optimización en el código Arduino, se ha conseguido una gran velocidad de reacción por parte de la placa, el cual es prácticamente instantáneo a la hora de disparar el flash desde el momento en el que detecta el evento.

Además, gracias a la interfaz gráfica que se incluye, es posible realizar muchos tipos de fotografía, con diferentes valores, todos ellos controlados a través del ordenador, permitiendo

también en un futuro añadir más opciones de una manera bastante simple.

En lo personal, me ha permitido conocer el funcionamiento de la placa Arduino y de esa manera aumentar mis conocimientos en electrónica. Además, aprendí a desarrollar interfaces gráficas en java y a realizar conexiones con otros elementos a través del puerto USB del ordenador.

También ha sido muy gratificante el hecho de poder ir resolviendo gran parte de los problemas que se van presentando en muchas ocasiones de manera individual, aunque siempre con el apoyo de Alexandre, mi tutor del proyecto, especialmente para la parte electrónica.

Este proyecto aún es posible mejorarlo. Si bien los cuatro sensores disponibles permiten prácticamente cualquier tipo de fotografía, aún es posible añadir otros elementos como por ejemplo una electroválvula, la cual permitiría fotografiar con buena precisión gotas de agua.

También sería interesante permitir un segundo flash, lo cual nos permitiría poder iluminar mejor la escena, o añadir la opción de disparar el flash varias veces en un lapso de tiempo determinado, de tal manera que se viera la trayectoria que recorre el objeto fotografiado.

Y por supuesto, aumentar la compatibilidad con más tipos de ordenadores a través del puerto serie, haciéndolo compatible tanto con sistemas UNIX como con versiones de java de Windows para 64 bits.

10.- Bibliografía

-Apuntes de java “Aide mémoire UML & Java” del profesor de la Universidad de Cergy Pontoise Marc Lemaire

http://www.monografias.com/trabajos45/amplificadores-operacionales/amplificadores-operacionales2.shtml – Información sobre los amplificadores operacionales

http://es.wikipedia.org - Wikipedia

http://alumni.media.mit.edu/~benres/simpleserial/ - Web del proyecto SimpleSerial

http://arduino.cc/ - Web de Arduino (con manual de referencia)