Febrero 2017

Curso Arduino avanzado

Ricardo Castrillejo Abad

INDICE

1. Curso

1.1 Objetivo del primer día

Conoceremos nuevos componentes y haremos un pequeño repaso de los anteriores, utilizaremos la ficha técnica de los componentes con el fin de crear nuestro método para que el aparato funcione correctamente.

Utilizaremos funciones nuevas como “Map”.

Veremos en detalle cómo trabajar con pines PWM y entradas Analógicas.

Practica 1, consistirá en conseguir la variación automática de un led en función de la luminosidad ambiente y muestreo simultáneo de los datos a través del puerto serial.

Practica 2, añadiremos una pantalla LCD al ejercicio anterior y mostraremos lo datos a través de ella.

1.2 Objetivo del segundo día

Veremos cómo usar varios puertos serie con nuestro Arduino.

Conoceremos el modulo Bluetooth HC06

Practica Final, Crearemos una App usando MIT AppInventor2, que mediante una conexión vía Bluetooth nos permitirá controlar un led a distancia colocado en Arduino UNO.

Leer el apartado 8.1 Recomendaciones.

2. Avanzado: Clase 1 (2h)

2.1 Objetivos (30min)

Al completar esta clase se obtienen los siguientes conocimientos:

- Manejo de Datasheet propios de componentes
- Control de pines analógicos y PWM
- Función Map();

2.2 Entradas Analógicas en Arduino.

Una señal analógica es una magnitud que puede tomar cualquier valor dentro de un intervalo –Vcc y + Vcc**.** Por ejemplo, una señal analógica de tensión entre 0V y 5V podría valer 2,72V, o cualquier otro valor con cualquier número de decimales. Por contra, recordemos que una señal digital de tensión teórica únicamente podía registrar dos valores 0V o 5V.

2.2.1 Precisión en la medición

Para entender la precisión de una entrada analógica es necesario entender cómo funciona un conversor analógico digital (ADC), que es su componente fundamental.

2.2.2 Precisión relativa y referencia de tensión analógica (AREF)

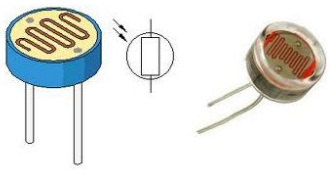
2.3 Señales analógicas de salida en Arduino (PWM)(15 min)

En este apartado vamos a ver los fundamentos en los que se basa la generación de salidas analógicas en Arduino. El procedimiento para generar una señal analógica es el llamado PWM.

2.4 Ejemplo práctico (generar una señal analógica)(15 min)

Pondremos en práctica lo aprendido con un simple ejemplo, variando la intensidad lumínica de un led con un programa, consolidando los conocimientos de señal analógica y pines PWM.

2.5 Fotoresistencia LDR(30 min)



El LDR (Light Dependent Resistor) o resistencia dependiente de la luz o también fotocélula, es una resistencia que varía su resistencia en función de la luz que incide sobre su superficie. Cuanto mayor sea la intensidad de la luz que incide en la superficie del LDR menor será su resistencia y cuanto menos luz incida mayor será su resistencia.

2.5.1 Datasheet

Obtendremos la información necesaria del componente.

2.5.2 Funcionamiento LDR

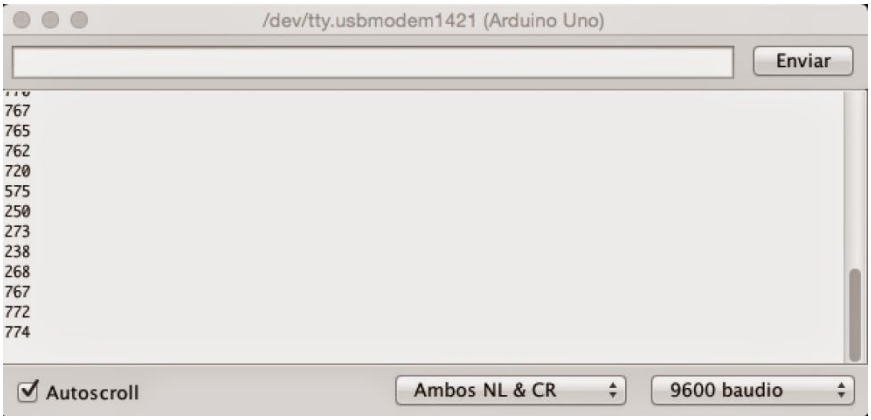
Aprovecharemos la capacidad de arduino para realizar unas operaciones matemáticas simples, que nos serán de utilidad para comprender el funcionamiento de la fotoresistencia LDR.

2.5.3 Practica 1.1, Obtención de los datos analógicos de la Fotoresistencia

Crearemos un primer programa que será la base para las siguientes prácticas, utilizando el puerto serial y el uso de la función analogRead();

```
int lightPin = 0;  //define el pin de la foto-resistencia
int valor;    //define una variable en la que haremos los cálculos
```

```
void setup()
{
  Serial.begin(9600);  //Inicializa la comunicación serie
}

void loop()
{
  valor = analogRead(lightPin);
  Serial.println(valor); delay(1000);}
```



2.5.4 Practica 1.2 Variación de la Intensidad lumínica de un Led (15 min)

Añadiremos un led al ejercicio anterior y veremos cómo modificar su intensidad lumínica en función de los datos que nos proporciona la fotorresistencia.

```
int lightPin = 0;  //define el pin de la foto-resistencia
int ledPin=9;    //define el pin para el  LED
int valor;    //define una variable en la que haremos los cálculos

void setup()
{
  Serial.begin(9600);  //Inicializa la comunicación serie
  pinMode( ledPin, OUTPUT );
}

void loop()
{
  valor = analogRead(lightPin);
  analogWrite(ledPin, valor);
  Serial.println(valor);
  delay(1000); //pequeño retardo para darle
        // tiempo al LED a responder.
}
```

2.6 Dudas generales(15 min)

3. Clase 2 avanzado (2h)

3.1 Función Map(15min)

El comando map() re-mapea un número desde un rango hacia otro. Esto significa que, un valor contenido en el al rango desdeBajo-desdeAlto será mapeado al rango hastaBajo-hastaAlto.

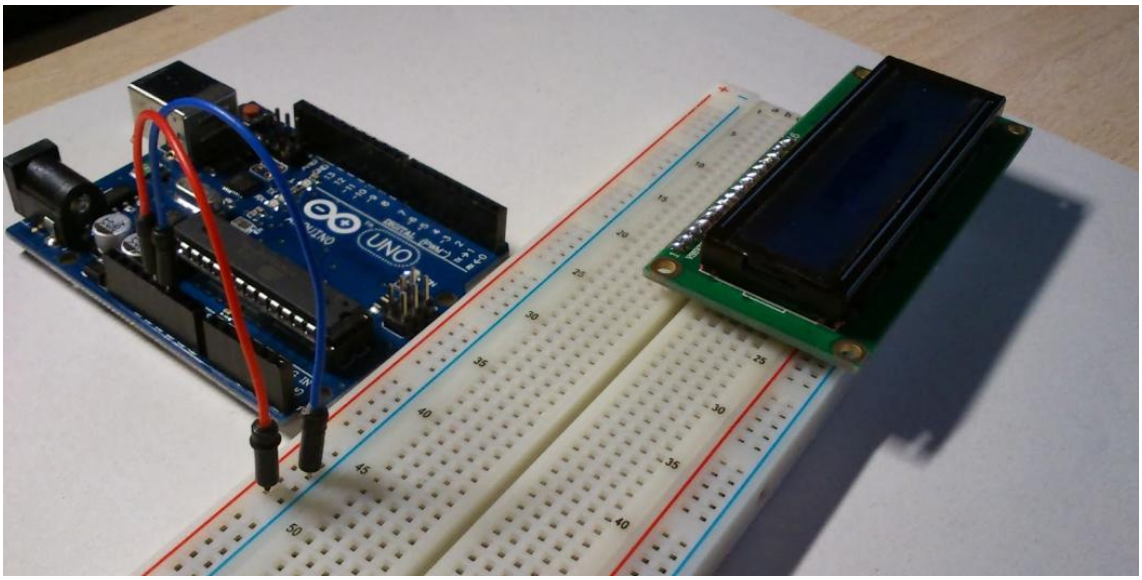
3.2 Añadiendo Map a nuestra practica(15 min)

```
int lightPin = 0;  //define el pin de la foto-resistencia
int ledPin=11;    //define el pin para el  LED
int valor;    //define una variable en la que haremos los cálculos
int min = 0;      //valor mínimo que da la foto-resistencia
int max = 0;      //valor máximo que da la foto-resistencia

void setup()
{
 Serial.begin(9600);  //Inicializa la comunicación serie
  pinMode( ledPin, OUTPUT );
}

void loop()
{
   valor = analogRead(lightPin);
   valor = map(valor, min, max, 0, 255);
   analogWrite(ledPin, valor);
   Serial.println(valor);
   delay(10); //pequeño retardo para darle
          // tiempo   al LED a responder.
}
```

4. Practica 2 Muestreo de datos a través de Pantalla LCD

4.1 Pantalla LCD(15 min)

Recordatorio rápido de la pantalla LCD, pines, librería a utilizar etc..



4.2 Práctica libre (30 min)

Con todo lo aprendido tenemos material necesario para intentar realizar la práctica de forma individual.

4.3 Resolución de la práctica (30 min)

4.4 Dudas generales (15 min)

5. Avanzado clase 3(2h)

5.1 Varias comunicaciones Serial

Para poder comunicarnos con el modulo Bluetooth y realizar un seguimiento con el monitor serial necesitamos conocer la librería de SoftwareSerial();

Además evitaremos errores al dejar libres los pines 0 y 1 de nuestro Arduino Uno.

5.2 Configuración de SoftwareSerial();

La biblioteca SoftwareSerial ha sido desarrollada para permitir la comunicación serie en otros pines digitales del Arduino, usando el software para replicar la funcionalidad (de ahí el nombre de "SoftwareSerial"). Es posible tener múltiples puertos serie de programas con velocidades de hasta 115200 bps. Un parámetro permite la señalización invertida para dispositivos que requieren ese protocolo.

```
#include <SoftwareSerial.h>

 const byte rxPin = 2;
 const byte txPin = 3;
```

```
// configura un nuevo objeto serie
SoftwareSerial mySerial (rxPin, txPin);
```

6. El modulo Bluetooth HC06

6.1 Descripción

Características

Pines

Velocidades permitidas y recomendadas

Tipos de Clases

Bluetooth Máster y Esclavo

6.2 Configurando nuestro Bluetooth HC06

6.2.1 Comandos AT

Conoceremos los comandos AT para poder proceder con la primera práctica.

Estos comandos se usan para la comunicación con muchos módulos.

6.2.2 Primera práctica

Comando AT	Descripción	Respuesta
AT	Test de comunicación.	Responde con un **OK**
AT+VERSION	Retorna la versión del Modulo	**OKlinvorV1.8**
AT+BAUDx	Configura la velocidad de trasmisión del modulo según el valor de "x"1 = 1200 bps 2 = 2400 bps 3 = 4800 bps 4 = 9600 bps (por defecto) 5 = 19200 bps 6 = 38400 bps 7 = 57600 bps 8 = 115200 bps 9 = 230400 bps A = 460800 bps B = 921600 bps C = 1382400 bps	AT+BAUD4 Configura la velocidad a 9600 baud rate Responde con **OK9600**
AT+NAMEx	Configura el nombre con el que se visualizara el modulo, soporta hasta 20 caracteres	AT+NAMEDIYMakers Configura el nombre del modulo a DIYMakers Responde con**OKsetname**
AT+PINxxxx	Configura el Pin de acceso al modulo (password).1234 por defecto.	AT+PIN1122 Configura el pin a 1122 Responde con **OKsetPIN**

- Se proporcionara el programa que nos permite configurar el modulo (se explicara cómo funciona).

- Cambiaremos nombre, y veremos cómo cambiar velocidades y Pin.

6.2.3 Funciones para usar HC06

Available(); begin(); isListening(); read(); overflow(); peek(); print(); println(); listen(); write();

7. Practica final primera parte (Arduino)

Comenzaremos con la practica final de este curso, con lo aprendido en esta clase podemos empezar a hacer el código que irá en Arduino, con el objetivo de que el modulo Bluetooth este a la escucha y detecte un carácter que posteriormente le enviaremos desde el Smartphone.

7.1 Dudas generales

8. Clase 2(2h)

8.1 Recomendaciones

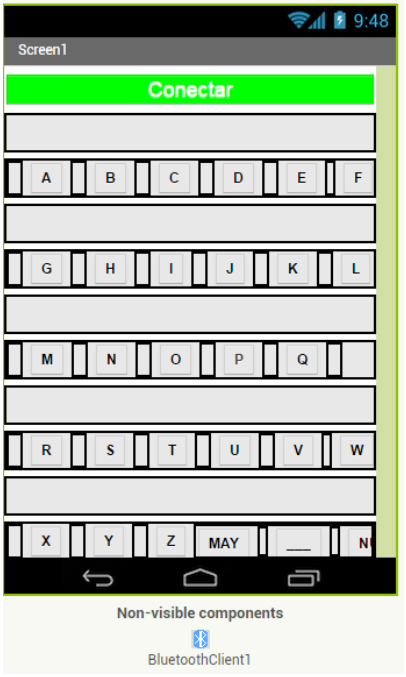
Para esta parte recomiendo traer un Smartphone que funcione con Android, porque usaremos AppInventor2 y solo nos sirve para móviles con este sistema.

Tener instalado en el Smartphone una App para leer códigos BIDI.

8.2 MIT AppInventor2

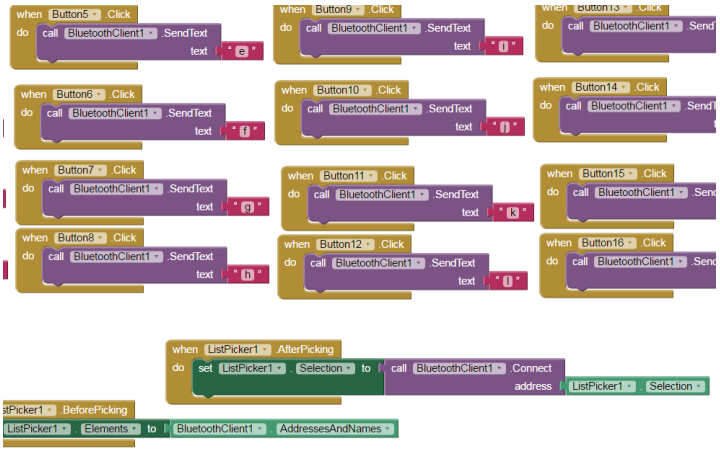
- Entorno
- Espacios de trabajo
- Programación con bloques
- Apariencia
- Descarga

8.2.1 Apariencia



- Aprenderemos a usar botones, conectar con nuestro Bluetooth HC06, Layout y haremos un recorrido rápido por las diferentes posibilidades que tiene AppInventor2.

8.2.2 Programación con Bloques



9. Práctica final ultima parte

Como en esta última parte hemos visto muchas cosas nuevas, el programa que irá en el Smartphone se habrá ido haciendo en paralelo con la explicación, puntos 8.2.1 Apariencia y 8.2.2 Programación con bloques, con el fin de que todos puedan tener la App funcionando al fin de esta práctica.

Enlazaremos esta práctica con Android con la programación que teníamos hecha anteriormente en Arduino.

9.1 Dudas generales