Grupo de trabajo

PRÁCTICAS Y EVALUACÓN DE ACTIVIDADES CON ARDUINO.

IES JUAN SEBASTIÁN ELCANO

SANLUCAR DE BARRAMEDA Curso 2016/2017

Participantes

María del Carmen Sotelino Polonio

Cesar Fuentes Barrientos

Mónica López Sánchez

José Manuel Romero Hermosilla

Juan Francisco Merino Erencia

El objetivo de este grupo de trabajo ha sido tener un primer contacto con la programación basada en Arduino, ya que hoy día es una herramienta educativa muy potente, que reúne parte de programación software con un montaje hardware.

En este primer contacto nos hemos enriquecido todos los profesores que hemos colaborado en este grupo de trabajo, ya que hemos puesto en común nuestros conocimientos y hemos profundizado en la programación y montaje de circuitos gracias a las experiencias y conocimientos previos de cada uno de nosotros. Esta puesta en común de conocimientos ha sido muy satisfactoria.

Además de poner nuestros conocimientos en común, hemos elaborado una guía de prácticas con Arduino la cual será de gran ayuda en los cursos siguientes en la asignatura de Tecnología tanto para la ESO como para Bachillerato.

Las primeras sesiones las hemos dedicado a familiarizarnos con la placa Arduino, la cual puede trabajar tanto en Windows como en Linux o Ubuntu, pero la instalación en estos sistemas operativos varía según sea uno u otro. En la guía explicamos el proceso a seguir para la instalación del programa Arduino con el que se programa finalmente la placa Arduino.

Además de ver la instalación en los distintos sistemas operativos, hemos realizado prácticas y montajes con elementos de entra de datos a la placa, como son pulsadores y sensores. Estas prácticas consistieron en conectar pulsadores y sensores a la placa arduino, tanto de forma analógica como

digital, siendo capaces de leer esos valores de entrada y realizar operaciones internas con ellos.

Posteriormente, hemos realizado prácticas con actuadores, los cuales son dispositivos que se conectan a la placa Arduino y son gobernados por esta. Entre los actuadores utilizados podemos señalar leds, motores CC, ventilador CC y zumbadores.

Las siguientes sesiones consistieron en ir practicando con estos sensores y actuadores, recogiendo información del entorno, y realizando diversos programas que en función de esa información, manejaran de una forma u otra nuestros actuadores de forma automática.

A continuación se muestra la guía de prácticas con Arduino que hemos elaborado, donde se pone en orden de menor a mayor dificultad lo aprendido entre todos en estas sesiones. Esta guía está enfocada para realizar prácticas guiadas con los alumnos, explicando en qué consiste las práctica, los elementos que se requieren para llevarlo a cabo (tipos de sensores, placa protoboard, número de cables de conexionado, actuadores), el montaje de estos elementos y el programa que habría que cargar en la placa.

En nombre de los participantes de este grupo de trabajo, quiero recalcar la utilidad práctica de esta guía de prácticas que hemos elaborado, ya que además de estar enfocada hacia alumno, nos ha hecho profundizar en los conocimientos de programación con Arduino y nos ha dado la oportunidad de disfrutar aprendido y compartiendo los conocimientos entre los compañeros integrantes de este grupo.

Guía ARDUINO

Instalación:

Para instalar el entorno de programación arduino en:

- Windows:

Entramos en https://www.arduino.cc/ y picamos en Software, allí nos aparece una ventana a la izquierda con distintas opciones de Windows, elegir la versión y hacer click en instalar

Linux: Entramos en https://www.arduino.cc/ y picamos en Software, allí nos aparece una ventana a la izquierda con distintas opciones de Linux, entre ellas Linux 32 Bits, Linux 64 Bits y Linux ARM, elegir la versión según nuestro equipo y hacer click en instalar.

- Ubuntu:

Tenemos que instalar los siguientes paquetes:

- librxtx-java
- avr-libc & gcc-avr
- sun-java6-jre

Para ello debemos de entrar en Sistema > Administración > Gestor de Paquetes Synaptic
En la ventana del Synaptic proceda a seleccionar cada uno de los paquetes.

Una vez instalados entramos en http://arduino.cc/ y desde su aquí nos permite instalarlo.

A partir de la versión Ubuntu 10.10, desde el «centro de software de Ubuntu» se instala directamente Arduino.

Práctica 1 Parpadear un Led

Vamos a conectar un Led a la placa Arduino, en este ejemplo se ha utilizado el pin13, haciendo que se encienda durante 1 segundo y que se apague durante 1 segundo, repitiéndose en bucle.

Componentes:

- Placa Arduino
- Placa Protoboard
- 1 Led
- 2 Cables de conexión

Código:

```
void setup() {                  //comienza la configuración
pinMode(13, OUTPUT);      //configura el pin 13 como de salida
}                                   //termina la configuración
void loop() {                  //comienza el bucle principal del programa
digitalWrite(13, HIGH);      //envía 5V al pin (salida) 13
delay (1000);                   // espera 1 segundo pin 13 con 5V
digitalWrite(13, LOW);     //envía 0V al pin (salida) 13
delay (100);                 //espera 100 ms pin 13 con 0V
}
```

Práctica 2 Encendido secuencial de grupo de Leds

Vamos a encender y apagar 4 leds secuencialmente. Los leds de este ejercicio están conectados a los pines 8,9,10 y 11.
Se deben encender y posteriormente apagar los leds desde el pin 5 al 8, con un tiempo de duración de encendido y apagado de medio segundo.

Componentes:

- Placa Arduino
- Placa Protoboard
- 4 Leds
- 5 Cables de conexión

Código:

```
int tiempo=500;      //declara una variable como entero y de valor
                          //0.5 segundos
void setup() {          //comienza la configuración
pinMode(8,OUTPUT);
pinMode(9,OUTPUT);
pinMode(10,OUTPUT);
```

```
pinMode(11,OUTPUT);
}
void loop() {                    //comienza el bucle principal del programa
digitalWrite(5,HIGH);
delay(tiempo);
digitalWrite(5,LOW);
delay(tiempo);
digitalWrite(6,HIGH);
delay(tiempo);
digitalWrite(6,LOW);
delay(tiempo);
digitalWrite(7,HIGH);
delay(tiempo)
digitalWrite(7,LOW);
delay(tiempo);
digitalWrite(8,HIGH);
delay(tiempo);
digitalWrite(8,LOW);
delay(tiempo);
}
```

Práctica 3 Semáforo

Vamos a simular dos semáforo que controlan un cruce, para lo cual vamos a utilizar 6 leds, de forma que semáforo primero tendrá un led rojo, otro verde y otro amarillo, y el semáforo segundo también dispondrá de sus propios leds rojo, verde y amarillo. Estos leds han sido conectados a los pines 3,4,8,6,7,8.

Componentes:

- Placa Arduino
- Placa Protoboard
- 6 Leds
- 7 Cables de conexión

Código:

```
int leds[]={3,4,5,6,7,8);
int tiempo1=3000;
int tiempo2=500;
int n;

void setup() {
for (n=0;n<6;n++) {
pinMode (leds[n],OUTPUT);
}
}

void loop () {
```

```
digitalWrite (leds[0],HIGH);
digitalWrite (leds[5],HIGH);
delay (tiempo1);

digitalWrite (leds[5],LOW);
digitalWrite (leds[4],HIGH);
delay (tiempo2);

difitalWrite[leds[0],LOW);
digitalWrite (leds[2],HIGH);
digitalWrite (leds[4],LOW);
digitalWrite (leds[3],HIGH);
delay (tiempo1);

digitalWrite (leds[2],LOW);
digitalWrite(leds[1],HIGH);
delay (tiempo2);

}
```

Práctica 4 Manejo secuencial de Leds con un pulsador

Vamos a encender luces leds de forma secuencial, en este caso cuatro Leds, mediante un pulsador. Los Leds estarán conectados a los pines 5,6,7,8 y el pulsador estará conectado al pin 4. Y el funcionamiento de la práctica consistirá en que por cada pulsación del pulsador, se irá encendiendo el led conectado al pin 5, luego el 6 y así sucesivamente hasta encender el led 8. Si continuamos pulsando el pulsador, empezarán a apagarse los leds en orden inverso.

Componentes:

- Placa Arduino
- Placa Protoboard
- 4 Leds
- 7 Cables de conexión
- 1 Pulsador

Código:

```
int cadenaleds[]={5,6,7,8};
int pulsador=4;
int tiempo=200;
int n=0;
void setup() {
for(n=0;n<4;n++) {
pinMode (cadenaleds[n],OUTPUT);
}
pinMode (pulsador,INPUT);
}
void flash() {
```

```
for (n=0;n<4;n++) {
digitalWrite (cadenaleds[n],HIGH);
delay (tiempo);
digitalWrite (cadenaleds[n],LOW);
delay (tiempo);
}
}

void loop() {
if (digitalRead(pulsador)==HIGH) {
flash ();
}
}
```

Practica 5 Ventilador

Vamos a encender y apagar un ventilador de CC junto con un Led dependiendo de la temperatura que prefijemos, para lo cual nos vamos a ayudar de un sensor llamado resistencia NTC, la cual varía su valor resistivo según la temperatura. El sensor NTC lo conectaremos a una entrada analógica, la cual lee valores comprendidos entre (0 y 1024), fijando nosotros que valor se corresponde con la temperatura a la que queremos que se encienda el ventilador y el diodo LED.
El ventilador CC irá conectado a una Pila de 5 Voltios, y la alimentación del ventilador con la pila irá gobernada por un relé electrónico que abrirá y cerrara el circuito. Es el relé electrónico el dispositivo el cual maneja y va conectado nuestra placa Arduino, separando así la etapa de control de la etapa de pontecia.

Componentes:

- Placa Arduino
- Placa Protoboard
- 1 Led
- 1 Ventilador CC
- 1 Pila 5V
- 1 Relé electrónico
- 10 Cables de conexión

Código:

```
int led=5;
int ntc=0;
int motor=10;
int medida=0;
int nivel=700;  //variable que guarda el límite de temperatura al que se
                //activa el ventilador
void setup()
{
```

```
pinMode(led,OUTPUT);
pinMode(motor,OUTPUT);
Serial.begin(9600);
}
void loop()
{
medida=analogRead(ntc);
if(medida>nivel)       //si la señal del sensor supera el nivel marcado:

{
digitalWrite(led,HIGH);            //se enciende el Led
digitalWrite(motor,HIGH);          //arranca el ventilador arranca
}

else   {                  // si la señal está por debajo del nivel marcado

digitalWrite(led,LOW);
digitalWrite(motor,LOW);         // el ventilador se para
}
}
```

Practica 6 Variación luminosidad

Vamos a variar la luminosidad de un led, aprovechando la capacidad PWM de algunos pines digitales de arduino, que simulan una salida analógica mediante variación de la anchura de pulso de la señal de salida. Vamos a conectar un Led a uno de los pines digitales que dispone de señal PWM, en este caso el pin 6, y vamos a conseguir que su luminosidad varíe desde 0 voltios, (apagado) hasta un nivel máximo de tensión de 5 Voltios (luminosidad máxima). Para conseguirlo los valores de la señal PWM irán variando entre los intervalos 0 y 255.

Componentes:

Placa Arduino
Placa Protoboard
1 Led
2 Cables de conexión

Código:

```
int luminosidad = 0;       // variable para asignar la luminosidad al led
int led = 6;               // pin del led
void setup()
{
pinMode(led,OUTPUT);
}
void loop()
{
for (luminosidad = 0 ; luminosidad <= 255; luminosidad=luminosidad+3)
{
analogWrite(led, luminosidad);     // ilumina el led con el valor asignado
                                   //a luminosidad (entre 0 y 255)
delay(30);                         // espera 30 ms para que se vea el efecto
```

Practica 7 Sensor LDR

Un sensor LDR es un componente electrónico pasivo cuyo valor de la resistencia varía en función de la luz que recibe. Cuanta más luz reciba, el valor de su resistencia será menor.

Se conectarán 5 LED que irán encendiéndose dependiendo del dicho valor de resistencia, ligado inversamente con la cantidad de luz, de forma que conforme vaya disminuyendo la cantidad de luz, se irán encendiendo los LED de forma progresiva.

Utilizaremos los pines 12, 11, 10, 9 y 8 como salidas digitales a los que conectaremos los diodos. El sensor LDR irá en serie con una resistencia de 1kΩ, y lo alimentaremos en un extremo con 5 voltios y el otro extremo a tierra. Justo en la conexión entre el sensor y la resistencia conectaremos un clave que irá uno de los pines de entrada analógicos de la placa, en nuestro ejemplo el A0.

Componentes:

Placa Arduino
Placa Protoboard
5 Led
1 sensor LDR
1 Resitencia 1kΩ
9 Cables de conexión

Código:

```
int valorLDR = 0;

//Decimos que pines vamos a utilizar para LED

int pinLed1 = 12;
int pinLed2 = 11;
int pinLed3 = 10;
int pinLed4 = 9;
int pinLed5 = 8;

//Y que pin para la LDR

int pinLDR = 0;

void setup()
{
 //Establecemos como salida los pines para LED

 pinMode(pinLed1, OUTPUT);
 pinMode(pinLed2, OUTPUT);
 pinMode(pinLed3, OUTPUT);
 pinMode(pinLed4, OUTPUT);
 pinMode(pinLed5, OUTPUT);

 //Le decimos que vamos a usar una referencia externa
//analogReference(EXTERNAL);

}

void loop()
{
 valorLDR = analogRead(pinLDR);   //Guardamos el valor leído en una variable
                                  //y comenzamos las comparaciones:
 if(valorLDR >= 1023)
 {
   digitalWrite(pinLed1, LOW);
   digitalWrite(pinLed2, LOW);
   digitalWrite(pinLed3, LOW);
   digitalWrite(pinLed4, LOW);
   digitalWrite(pinLed5, LOW);
 }
 else if((valorLDR >= 823) & (valorLDR < 1023))
 {
   digitalWrite(pinLed1, HIGH);
   digitalWrite(pinLed2, LOW);
   digitalWrite(pinLed3, LOW);
   digitalWrite(pinLed4, LOW);
   digitalWrite(pinLed5, LOW);
 }
```

```
  else if((valorLDR >= 623) & (valorLDR < 823))
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, HIGH);
    digitalWrite(pinLed3, LOW);
    digitalWrite(pinLed4, LOW);
    digitalWrite(pinLed5, LOW);
  }
  else if((valorLDR >= 423) & (valorLDR < 623))
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, HIGH);
    digitalWrite(pinLed3, HIGH);
    digitalWrite(pinLed4, LOW);
    digitalWrite(pinLed5, LOW);
  }
  else  if((valorLDR >= 223) & (valorLDR < 423))
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, HIGH);
    digitalWrite(pinLed3, HIGH);
    digitalWrite(pinLed4, HIGH);
    digitalWrite(pinLed5, LOW);
  }
  else
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, HIGH);
    digitalWrite(pinLed3, HIGH);
    digitalWrite(pinLed4, HIGH);
    digitalWrite(pinLed5, HIGH);
  }

}
```

Practica 8 Notas musicales con un zumbador

Vamos a reproducir mediante un zumbador una escala de notas musicales. Un zumbador es un transductor electroacústico que produce un sonido o zumbido continuo o intermitente de un mismo tono. Puede servir como mecanismo de señalización o aviso, como ocurre en multitud de electrodomésticos.

Para la conexión del circuito conectaremos la patilla de la izquierda del zumbador con el PIN GND de nuestra placa, y la patilla de la derecha, con el PIN 12 del arduino.

Componentes:

Placa Arduino
Placa Protoboard
1 zumbador
2 Cables de conexión

Código:

```
int speakerPin = 12; // Introducimos la variable por donde saldrá nuestra señal
//digital hasta el zumbador

int numTones = 10; // Definimos una variable con el número de tonos que va a
reproducir

int tones[] = {261, 277, 294, 311, 330, 349, 370, 392, 415, 440};
//              mid C  C#   D     D#    E     F     F#    G     G#   A
// Arriba se muestran las equivalencias entre frecuencias y Notas de la escala natural.
void setup()
{
// Generamos un bucle que recorra nuestro vector. Este será el encargado de introducir
//una determinada frecuencia al zumbador cada vez, conforme hayamos declarado el
//vector de tonos.
  for (int i = 0; i < numTones; i++)
  {
    tone(speakerPin, tones[i]);
    delay(500);
  }
  noTone(speakerPin);
}
void loop()
{
}
```