clase-04

martes 05 abril 2022, presencial

repaso clase anterior y programa hoy (10 min)

la clase pasada aprendimos:

- electricidad y magnetismo
- componentes eléctricos
- circuitos eléctricos
- instalación de software para el curso

hoy aprenderemos:

- fundamentos de programación en Arduino
- comunicación serial entre Arduino y computador
- programar semáforo en Arduino
- construir circuito para semáforo

fundamentos de programación en Arduino (1 hora)

Arduino está basado en Processing, y hereda las 2 funciones principales:

```
// setup() ocurre al principio de los tiempos, una vez
void setup() {

  // codigo para configurar condiciones iniciales

}

// loop() ocurre despues de setup(), en bucle
void loop() {

  // codigo para refrescar, leer y escribir informacion

}
```

nuestro Arduino puede almacenar distintos tipos de variables, las que son útiles para distintos propósitos.

en este curso usaremos las siguientes:

```


// bool almacena valores verdadero o falso (true, false)
bool verdad = true;

// byte almacena 8 bits, valores enteros entre 0 y 255
```

```
byte ochoBits = 255;

// int almacena numeros enteros, tamano 2 bytes = 16 bits
int numeroEntero = -4;

// float almacena numeros con parte decimal
float numeroDecimales = 123.456;

// char almacena un caracter, entre comillas simples ''
char miInicial = 'a';
// tambien puedes escribir directamente el valor decimal
char miInicialDecimal = 141;

// String almacena un arreglo de caracterers, entre comillas dobles ""
String verso = "habia una vez";
```

ejemplo con LED interno

código completo en ej_00_led_interno

nuestro Arduino Uno tiene un LED interno conectado internamente al pin 13.

creamos una variable de tipo número entero para almacenar el valor 13.

en la configuración (setup) hacemos que el pin digital 13 sea una salida (OUTPUT).

en el refresco (loop) hacemos que la nuestra salida digital del pin 13 alterne entre valores digitales 1 y 0, con una pausa de 1 segundo entre cada estado.

ejemplo imprimir String

código completo en ej_01_led_imprimir_string

nuestro Arduino puede imprimir valores a través del puerto serial.

estos mensajes podemos leerlos en el monitor serial del software Arduino IDE.

en el monitor serial tenemos opciones de configuración de velocidad de baudios, de avanzar automáticamente, o de registrar el tiempo de llegada de cada mensaje.

ejemplo imprimir variables internas

código completo en ej_02_led_imprimir_variable

también podemos imprimir valores de variables internas a través del puerto serial.

ejemplo arrojar dado

código completo en ej_03_arrojar_dado

puerto serial y salida

para comunicar nuestro Arduino a través del puerto serial, debemos abrir el puerto y configurar una velocidad.

la velocidad estándar es de 9600 baud, y si quisiéramos transmitir por protocolo MIDI, la velociad es 115200 baud.

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println("hola :)");
}
```

imprimamos una variable

```
int x = 1;

void setup() {
  Serial.begin(9600);
}

void loop() {
  x = x + 1;
  Serial.println(x);
}
```

hagamos un dado digital, y practiquemos imprimir un mensaje en partes

```
int minDado = 1;
int maxDado = 6;

void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.print("dado: ");
  Serial.println(random(minDado, maxDado + 1));
  delay(1000);
}
```

puerto serial y entrada

```

```

```
int byteEntrada;

void setup() {
  Serial.begin(9600);
}

void loop() {

  if (Serial.available() > 0) {
    //  leer el byte
    byteEntrada = Serial.read();

    // imprimir el resultado recibido en sistema decimal
    Serial.print("recibido: ");
    // DEC para asegurarnos que se imprime en decimal
    Serial.println(byteEntrada, DEC);
  }
}
```

poema condicional

```
int byteEntrada;

int numeroMin = 48;
int numeroMax = 57;

char *poema[] = {"verso0", "verso1", "verso2",
                 "verso3", "verso4", "verso5",
                 "verso6", "verso7", "verso8",
                 "verso9"
                };

void setup() {
  Serial.begin(9600);
}

void loop() {

  if (Serial.available() > 0) {
    //  leer el byte
    byteEntrada = Serial.read();

    if (byteEntrada <= numeroMax && byteEntrada >= numeroMin) {
      Serial.println(poema[byteEntrada-numeroMin]);
    }
  }
}
```

el número impreso está en ASCII.

13 es CR, por carriage return, en español retorno de carro.

conectar Arduino con componentes

qué es código

Diferencias entre espacios y tabulaciones.

Diferencias entre UTF-8 y emojis y sistemas de Strings.