https://repository.uaeh.edu.mx/revistas/index.php/prepa4/issue/archive

Vida Científica Boletín Científico de la Escuela Preparatoria No. 4

Publicación semestral, Vol. 11, No. 21 (2023) 35-40

Vida Científica

ISSN: 2007-4905

Programación para principiantes en Arduino

Programming for beginners in Arduino

Marisela Vital-Carrillo [a]

Abstract:

Arduino is an open-source hardware platform, based on a simple board that contains a microcontroller, this board connects the physical world with the virtual world, or the analog world with the digital one, in addition to controlling sensors, alarms, communication and physical actuators which work according to the programming that will be created in the Arduino IDE.

The Arduino software is based on a free license so that it is easy to download and users start to acquire programming skills with the use of the main instructions and functions.

Keywords:

Arduino, software, programming, code, instructions

Resumen:

Arduino es una plataforma de hardware de código abierto, basada en una placa sencilla que contiene un microcontrolador, esta placa conecta el mundo físico con el mundo virtual, o bien el mundo analógico con el digital, además de que controla sensores, alarmas, sistemas de comunicación y actuadores físico los cuales funcionan de acuerdo con la programación que se crear en la IDE de Arduino.

El software de Arduino está basado en una licencia libre para que se descargue sin problema y los usuarios comiencen a adquirir habilidades de programación con el uso de las principales instrucciones y funciones.

Palabras Clave*:*

Arduino, software, programación, código, instrucciones

[a] Marisela Vital Carrillo, Universidad Autónoma del Estado de Hidalgo, Escuela Preparatoria Número 4, ORCID: 0000-0002-4203-2583

Email: marisela_vital2403@uaeh.edu.mx

Fecha de publicación: 05/01/2023

Introducción

Con el software de Arduino se programa mediante el uso de un leguaje basados en una programación de alto nivel, el entorno de programación de Arduino es un código abierto , libre en la cual hace fácil escribirlo y cargarlo a

la placa, funciona con los sistemas operativos Windows, Mac OS y Linux. También es posible utilizar otros lenguajes de programación y aplicaciones populares de Arduino, algunos de ellos son Java, Python, Matlab, Adobe Director, VBScript, la programación de Arduino está basado en C y soporta todas las funciones del estándar de C y algunas de C++.
Martínez (2003), dice que el lenguaje de programación C, "Es uno de los más utilizados en la programación de sistemas de software, que es similar a Python en muchos aspectos fundamentales, ya que presenta las mismas estructuras de control, permite trabajar con algunos tipos de datos similares, además de que hace posible definir y usar funciones".

Para los usuarios que comienzan a programar en Arduino se recomienda utilizar el software oficial de Arduino que tiene el siguiente logotipo y su descarga es gratuita.

Figura 1

Para comenzar a utilizar este software es necesario conocer las herramientas principales de la IDE de Arduino.

La figura 2 es una herramienta que verifica que no haya errores en el código del programa

Figura 2

La siguiente herramienta de la figura 3 permite transferir el código del programa a la placa de Arduino para que se almacene en el microcontrolador.

Figura 3

Para abrir un nuevo programa se utiliza el icono de la herramienta de la figura 4, esto genera una nueva ventana donde escribir el código de ese nuevo programa.

Figura 4

Cuando se quiere abrir un programa que ya se ha creado previamente y esta guardado en un dispositivo de almacenamiento se usara la herramienta de la figura 5.

Figura 5

La herramienta de la figura 6 permite guardar el archivo donde se encuentra el código del programa.

Figura 6

El monitor serie, es una de las partes más importantes de la IDE de Arduino, la figura 7 muestra esta herramienta, además sirve para mostrar información de la comunicación entre el ordenador y Arduino en las dos direcciones

Figura 7

La figura 8 muestra el editor que es la pantalla donde se escribe las instrucciones o código para programar la placa de Arduino.

Figura 8

La consola de Arduino es la que se muestra en la figura 9 y sirve para proporcionar información sobre una acción concreta, además de que informa si existe un error en el programa.

Figura 9

Una vez que se identifiquen están partes se puede comenzar a programar.
Es importante en la programación el uso de comentarios para saber que se está creando o que instrucciones se utilizan para ejecutar correctamente el programa.

Para crear un comentario en Arduino se utilizan // y lo que se escribe a la derecha se pone de color gris y no afecta el código de programación.

También se puede poner un comentario de la siguiente manera /* para abrir y */ para cerrar, todo lo que este dentro de esto será un comentario.

En el lenguaje de programación C++ se utiliza el (;) que se considera un punto y aparte, es decir, en el código indica que ya se ha terminado una sentencia y partir de ese momento, sigue otra instrucción que no tiene nada que ver con la anterior. Un ejemplo es:

```
 int led = 3;
int  led = 9;
```

En caso de que la sentencia anterior no tenga al final el punto y coma se producirá un error.

Las **{ }** sirven para definir el principio y final de un bloque de instrucciones, por lo regular se utilizan en los bloques de la estructura general de programación de Arduino como es setup (), loop, if, entre otros.
Si se abra una llave siempre se debe de cerrar sino esto ocasiona errores en la programación.

En la programación de Arduino se utilizan **variables** que son herramientas de programación que ayuda almacenar y recuperar información de los programas.

Además, para empezar a programar en Arduino es necesario conocer los tipos de datos que utiliza este software, entre ellos esta:
Entero (int), que al utilizar este tipo dato es el más común y se utiliza para declarar valores enteros, se debe de escribir de la siguiente manera en la programación de un circuito.

Entero largo (long), este tipo de datos se utiliza para almacenar número más grandes ya sean positivos o negativos.

Decimal (float), este tipo de datos almacenan números decimales.

Byte, es un tipo de dato pequeño, solo ocupa 1 byte a la memoria y el rango de valores que puede almacenar es de 0 a 255 sin signo.

Booleano (boolean), este tipo de dato solo ocupa dos valores 0 y 1 o bien falso y verdadero.

Carácter (char), este tipo de dato permite almacenar una letra de una forma especial. Al programar se tiene que utilizar la letra entre comillas simples ‘ ‘ o bien utilizar el valor del código ASCII.

Una ve conociendo los tipos de datos tenemos que aprender a estructurar un programa de Arduino y para ello se utilizaran las funciones setup() y loop() y e la estructura del programa se antepone en estas funciones la palabra reservada **void** que indica que la función no devuelve ningún valor.

La función **setup**(), se invoca solo una vez cuando el programa comienza, se utiliza para inicializar los modos de trabajos de los pines, o el puerto en serie, mostrar un mensaje de bienvenida en una pantalla LCD o bien hacer girar un servomotor de una posición inicial por defecto.

La función **loop**(), contiene el código que se ejecutara continuamente, esta función es el núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo.

La instrucción **pinMode**, es utilizada en la parte de la función de setup() y sirve para configurar el modo de trabajo de un pin.
Existen dos modos diferentes de funcionamiento de los pines:
INPUT, cuando se coloca este modo el pin es utilizado como entrada de información a la placa de Arduino.
OUTPUT, cuando se coloca este modo el pin es utilizado como salida de información a la placa de Arduino.
Ejemplo

```
pinMode(3,OUTPUT);
```

En cuanto a la instrucción de **digitalRead**, leer el valor de un pin dando como resultado HIGH o LOW, el pin se puede especificar con una variable o con una constante.
HIGH, envía el valor de 1 y LOW, envía el valor de 0.
Ejemplo

```
Valor = digitalRead(5);
```

digitalWrite, es una instrucción que envía al pin definido como OUTPUT, del valor de HIGH o LOW, esta instrucción se utiliza por lo regular para apagar y prender un led, ejemplo

```
digitalWrite(5,HIGH); // prende el led que esta en el pin 5
```

analogWrite, esta instrucción permite escribir valores analógicos comprendidos en 0 y 255 a los pines PWM como 3,5,6,9,10 y 11 de la placa de Arduino que son los que permiten valore analógicos por ser pines modulación por ancho de pulso (PWM) los podemos identificar por la tilde(~) Ejemplo

```
analogWrite(3,200);
```

Tal como lo puedes ver en el ejemplo la función requiere el pin PWM que es donde se va a enviar el valor de 200

El valor que se puede enviar a estos pines de salida analógica puede darse en forma variable o constante, pero siempre con un margen de 0 a 255.

analogRead, con esta instrucción se lee el valor de un determinado pin definido como entrada analógica con una resolución de 10 bits y funciona solo con los pines de valor A0 a A5 y su rango de valor que podemos leer es de 0 a 1023.

Cuando se programan estos pines analógicos no se necesita declararlos como INPUT u OUTPUT ya que serán siempre definidos como INPUT, ejemplo

```
int valor = analogRead(5); // asigna a valor lo que se lee en la entada del pin
```

Delay, esta instrucción detiene la ejecución del programa la cantidad de tiempo en ms que se indica en la propia instrucción, el valor de 1000 equivale a 1 segundo.

Map, esta función permite relacionar el valor que se encuentra en un rango de valores a otro rango de valores. La función se utiliza de la siguiente forma

Map((Valor a transformar, InicialMenor, InicialMayor, FinalMenor, FinalMayor);

Un ejemplo es cunado queremos que un led suba su intensidad con un potenciómetro el cual comprende valor de 0 a 1023 y se conecta a las entradas analógicas, por tanto, la función map tendrá que transformar ese valor en un rango de 0 a 255 para que led funcione como se requiere.

```
int valor;
map (valor, 0, 1023, 255,0);
```

También se utilizan instrucciones de control como es la función **If- else, for, while**, que son sentencia que llevan una condición especifica para que se ejecuten las instrucciones que van dentro de ellas, ejemplo

```
For (int = 0; i< 255; +=1)
{
analogWrite(Red,i);
delay(2000);
}
```

La sentencia de **FOR** es utilizada para repetir un bloque de instrucciones un número determinado de veces, es decir, que mientras la condición que se utilice no sea verdadera se repetirán las instrucciones y cuando sea verdadera deja de ejecutarse el bloque.

```
if (digitalRead(5)==HIGH)
{
digitakWrite(verde) = HIGH;
}
else
{
digitalWrite(rojo) = HIGH
}
```

El ejemplo comprueba que si el valor del pin 5 es HIGH la salida será verde, en caso de que no la salida será rojo.

La sentencia if se utiliza para comprobar si una condición se cumple o no y ejecutar un bloque de instrucciones del código u otro dependiendo de ello, mediante esta instrucción se pueden ejecutar las diferentes sentencias dependiendo de los resultados de comparación de 2 elementos.

#Define, esta función permite la definición de valores constantes en todo el programa. En el desarrollo de la programación de Arduino no va a permitir entre otras cosas, que si cambiamos l número del pin en a que tenemos conectado el circuito se cambia el valor de la constante en forma automática en todas las sentencias en las que estamos usando ese valor se actualicen y todo siga funcionado en forma correcta. Ejemplo
#define verde 3;
Este ejemplo define una constante que se llama verde y que tiene el valor 3, es decir, Led verde está conectado al pin 3.

Cuando se utiliza la instrucción **INPUT_PULLUP**, indica que el circuito utilizará las resistencias internas de la placa de Arduino y cuando se utiliza INPUT se deberá utilizar las resistencias para los elementos de entrada.

Arduino ya incluye librerías y se utilizaran con la sentencia de #include esto es porque pueden incluir las librerías ya desarrolladas en la IDE. Las librerías que se utilizan con mayor frecuencia son:

LiquidCrystal, que contiene todos los elementos necesarios para interactuar de forma sencilla con una pantalla LCD.

SimpleDHT: Contiene todos los elementos necesarios para interactuar con el sensor de humedad y temperatura que se ha utilizado.

Las funciones principales a utiliza con el puerto Serie son:

Serial.begin(velocidad), donde la velocidad serán los baudios por segundo a los que se va a operar el protocolo.

Serial.begin(9600);
Con este ejemplo se configura la velocidad de comunicación entre la placa de Arduino y la computadora, se ha configurado a 9600 bps que es el valor típico de comunicación del puerto serie.

Serial.print (mensaje), es utilizado para enviar datos como texto ASCII,
Serial.print(“bienvenidos”);

Se utilizarán comillas simples para caracteres y comillas dobles para cadenas de texto.

Estas instrucciones y sentencias son las más se utilizan para comenzar a programar en Arduino.

Conclusión

Aprender Arduino tiene ventajas como el desarrollo de la habilidad creativa, el pensamiento lógico para poder manipular las herramientas e instrucciones de esta placa y programar con facilidad el software.

Arduino es una herramienta flexible que ayuda a crear o innovar proyectos que pueden ser utilizados en la vida real y que además tengan un beneficio para la sociedad.

Arduino no es exclusivo para los programadores, esta placa la pueden utilizar cualquier individuo desde nivel básico donde aprenderá a construir circuitos y manipular todos los elementos que los conforma, así mismo identificara la programación como se puede programar cada elemento de la placa de Arduino mediante instrucciones sencillas y graficas con el uso del simulador de tinkercad, que los ayudara a crear programas sencillo en el software oficial de Arduino.

Referencias

[1] Castaño Giraldo, S. A. (septiembre de 2018). introducción de Arduino. Recuperado el 01 de septiembre de 2022, de https://controlautomaticoeducacion.com/arduino/introducción/

[2] Evans, B. (2017). Manual-Manual de Programación. Recuperado el 27 de agosto de 2022, de https://arduinobot.pbworks.com/f/Manual+Programacion+Arduino.pdf

[3] Moreno Muñoz, A., & Córcoles Córcoles, S. (s.f.). Aprendiendo Arduino en un fin de semana. Time Of Software. Recuperado el 25 de agosto de 2022, de https://www.bolanosdj.com.ar/MOVIL/ARDUINO2/Arduinounfinsemana.pdf

[4] Pomares Baeza, J. (2009). Grupo e innovación Educativa en Automática. Recuperado el 28 de agosto de 2022, de Manual de Arduino: https://rua.ua.es/dspace/bitstream/10045/11833/1/arduino.pdf

[5] Universidad Industrial de Santander. (s.f.). Introducción de Arduino. Recuperado el 2022 de agosto de 28, de https://halley.uis.edu.co/tierra/wp-content/uploads/2014/12/arduino.pdf.