Estructura Arduino

Al empezar a trabajar con Arduino, se debe tener muy en claro la estructura de trabajo básica, esta estructura es muy simple y debe aparecer en TODOS los programas, se trata de dos funciones, la función "setup" y la función "loop".

void setup {} // La función "Setup" solo se invoca una vez al inicio del programa, y se usa para configurar los pines, configurar la comunicación serie o establecer un estado inicial o de partida para algún dispositivo.

void loop {} // La función "Loop" será la función donde se encuentre el grueso de nuestro programa, esta función se repetirá una y otra vez ejecutando lo que se encuentre en su interior, esta repetición es la que posibilita que el programa este actualizándose y respondiendo a los eventos que ocurran.

Las funciones "Loop" y "Setup" son necesarias en todos los programas que se hagan, pero no tienen por qué ser únicas, se pueden crear tantas funciones como queramos para organizar el contenido de nuestro programa ó para hacer tareas repetitivas dentro de nuestro programa principal. Las funciones están acotadas por las llaves{}, estas llaves definirán el bloque de la función.

Instrucciones

Delimitadores: son elementos que acotan o detienen el alcance de una función o instrucción.

{} Llaves: definen el principio y el final de un bloque de instrucciones o función, podemos encontrarlas en estructuras de control definiendo el bloque al cual afecta la función. Podemos anidar tantos bloques como queramos.

; Punto y coma: Se utiliza para separar instrucciones, es común olvidar colocarlo al final de línea. Si no lo ponemos no reconocerá el fin de la función y el programa dará un problema al compilar.

Comentarios: Es muy recomendable usar comentario desde el principio para hacer anotaciones sobre el funcionamiento del programa, esto nos va a ahorrar muchos problemas si nuestro programa se hace muy grande y va a permitir que otras personas entiendan como hemos hecho el programa.

/*….*/ Bloque de comentario: Son áreas de texto que pueden abarcar más de una línea, lo que escribamos entre esos símbolos será ignorado por el programa.

// Línea de Comentario: Funciona como el bloque de comentario, con la diferencia que solo será ignorado el texto que este a su derecha, al cambiar de línea perderá el efecto.

Funciones básicas

pinMode: Esta instrucción se usa para configurar los pines de nuestro Arduino como entrada o como salida. Se declarará dentro de la función "SetUp", por lo que la configuración de pines solo se hará una vez antes de empezar a ejecutar el programa principal.

digitalWrite(); Esta función permite escribir valores lógicos digitales en un pin de Salida de una tarjeta Arduino. Entonces, está función requiere que el pin haya sido declarado como salida previamente. Para empezar, esté elemento del lenguaje Arduino, requiere de dos parámetros de entrada.

digitalWrite(#PIN, ESTADO);

PARÁMETROS DE ENTRADA:

#PIN, se refiere al número del PIN de la tarjeta Arduino correspondiente.

ESTADO, aquí se colocara cualquiera de dos palabras: LOW ó HIGH. LOW indica una condición logica 0, es decir 0V, HIGH indicaría entonces lo opuesto, un 1 lógico, es decir 5v o 3.3v.

PARÁMETRO DE SALIDA:

Ninguno.

digitalRead(); Es utilizada normalmente en la función loop(), sirve para leer un valor (o poner en un estado) un pin digital. Los valores o estados posibles son HIGH (alto) o LOW (bajo). El valor leído puede ser almacenado en una variable o comprobarse dinámicamente en una condición.

Ejemplo:

```
// Lee en "pin"

digitalRead(pin);

// Ejemplo: Leer el pin digital 2

digitalRead(2);
```

delay(); Es una función que hace que el procesador espere. Por ejemplo, esta espera permite no hacer nada y esperar hasta la ejecución de la siguiente instrucción durante un retardo de tiempo definido. Entonces esta función tiene un parámetro de entrada del tipo entero, que es la espera en milisegundos.

Ejemplo:

```
void.loop() {

digitalWrite(ledPin, HIGH);      // enciende el LED

delay(1000);                      // espera un segundo

digitalWrite(ledPin, LOW);      //apaga el LED

delay( 1000 );                   // espera por un segundo

 }
```