PRONTUARIO DE ARDUINO

Plantilla de Arduino

```
// Estos símbolos permiten escribir comentarios de una línea
/* Entre estos símbolos puedes escribir comentarios de más
de una línea */
//Zona de inclusión de librerías
// Declaración de variables globales (para todo el programa)
void setup() {
  // Este bloque sólo se ejecuta al cargar el programa
}
void loop() {
  // Este bloque se ejecuta repetidamente
  // Declaración de variables locales (de la función)
  // Llamadas a funciones
}
tipo mifuncion(argumentos) {
  // Este bloque sólo se ejecuta cuando es llamada
  // Tipo es el tipo de dato que se devuelve como las variables (void, int, …)
}
```

Tabla de funciones

Función	Descripción	Ejemplo

pinMode(pin,Modo);	**Configuración de Entradas y Salidas**
	En la función setup(), se definen el modo de trabajo de las patillas de la tarjeta con el comando pinMode(pin, Modo):
	• Pin -> número de pin de la tarjeta
	• Modo:
	  ○ Entrada -> INPUT, 0
	  ○ Salida -> OUTPUT, 1
		void setup() {
		// Pin 13 como salida
		pinMode(13, OUTPUT);
		
		// Pin 10 como entrada
		pinMode(10, INPUT);
		}

digitalWrite(pin,valor);	**Escribe en la salida** del pin el valor **digital**:
	• pin: número de pin de la tarjeta
	• valor:
	  ○ HIGH , 1-> 5V
	  ○ LOW, 0 -> 0V
		// Envía “1” (5V) a la patilla 13
		digitalWrite(13, HIGH);
		
		//Envía “0” (0V) a la patilla 11
		digitalWrite(11, LOW);

digitalRead(pin);	**Lee en la entrada** del pin el valor **digital**:
	● pin: número de patilla de la tarjeta
	● valor: este puede ser:
	○ 0, LOW -> 0V en el pin
	○ 1, HIGH-> 5V en el pin
		//Leo la entrada 10> la guardo en y
		y=digitalRead(10);
		
		// Si hay "0" en pin10 -> espero 1s
		if (digitalRead(10)==0) {
		delay(1000);
		}

Función	**Descripción**	**Ejemplo**

delay(milisegundos);
delayMicroseconds(microseg)
	Espera o paraliza el programa los:
	● milisegundos
	● microsegundos
		// Espero 2 segundos =2000ms
		delay(2000);

analogWrite(pin,valor);	**Manda en el pin de salida un valor PWM**:
	● Definir el pin como SALIDA
	○ Sólo pin 3,5,6,9,10 y 11
	○ Son las que tienen un símbolo de alterna
	● valor: ofrece una señal periódica de pulso variable PWM. 1KHz
	○ Led (255>brillo intenso) y (0>no brilla)
	○ Motor->variador de velocidad (255 vel.max.)
		void setup() {
		pinMode(3,OUTPUT)
		}
		void loop(){
		// Luce poco el led
		analogWrite(3, 125);
		delay(1000); //espera 1s
		//luce mucho el led
		analogWrite(3,255);
		delay(1000); //espera 1s
		}

analogRead(analogPin);	**Lee el valor** de la patilla **analógica**:
	● No hay que definirla como entrada
	● Sólo se pueden utilizar:
	○ A0, A1, A2, A3, A4 y A5
	● El valor en una variable entera y oscila entre (0-1024)
		//zona de variables globales
		int valor=0;
		...
		//leo el valor de la entrada A0
		valor=analogRead(A0);

tone(pin,frecuencia,duracion)
 	
noTone(pin);
	Emite una onda cuadrada, generalmente **para producir un sonido**:
	● Pin: solo salidas 3,5,6,9,10 y 11
	● frecuencia (sonido) en microseg.
	● duración en milisegundos.
		// Emito Do en la patilla 3 corchea
		// Zumbador +R=100Ohm serie
		tone(3,261,500);
		noTone(3); // paro el sonido

Serial.setTimeout(miliseg)	**Tiempo que se espera a la escucha del USB**
	● Necesario para readBytesUntil()
		void setup() {
		Serial.begin(9600);
		// Espera 10s
		Serial.setTimeout(10000);
		}

Serial.begin(vel_trans);	**Configura la transmisión USB** de la tarjeta con el Ordenador.
	● vel_trans -> velocidad de la transmisión en baudios (bits/seg).
		void setup() {
		// transmisión a 9600 baudios
		Serial.begin(9600);
		}

Serial.print(valor);
Serial.println(valor);
Serial.write(variable);
Serial.write(buffer,length);
	Manda al ordenador por USB el contenido de la variable "valor".
	● Tiene que haberse configurado la comunicación (ver anterior punto)
	● println() es como print() pero añade a "valor" un final de línea (EOL = End Of Line)
	Donde variable puede ser:
		// Mando el contenido de jj y EOL
		Serial.println(jj);
		
		// Mando el contenido de jj
		Serial.print(jj);

	• Número • string: cadena de caracteres • buffer: vector de valores • length: longitud del vector	

Serial.read();
Serial.parseInt();
Serial.readBytes(buffer,length);
 	
Serial.readBytesUntil(caracter, buffer,length);
	Escucha al ordenador por USB:
	• read(): lee un byte
	• buffer: vector de bytes
	• length: longitud del vector
	• caracter: letra que sirve para parar la escucha.
	• parseInt(): devuelve el nº entero.
		// Lee un byte del PC y guarda en y
		y=Serial.read();

Función	**Descripción**	**Ejemplo**
DECISIONES Y CONDICIONES	En determinadas condiciones hay que hacer o dejar de hacer el ciclo normal de repeticiones	

if(condicion) {
// si se cumple
} else {
// si no se cumple
}
	No es imprescindible “else”
	• **Si se cumple** la condición, entonces se hace lo que hay entre las llaves de después de if
	• **si no se cumple**, lo que se ponga después de else
	Los operadores para la if:
	• == igual, != distinto
	• > mayor, < menor
	• >= mayor o igual, <= menor o igual
	• && y, || o
		// Si i=3, enciende LED13
		if(i==3) {
		digitalWrite(13,HIGH);
		}
		// Si j>4, enciende LED13
		// si no, apaga LED13
		if (j>4) {
		digitalWrite(13,HIGH);
		} else {
		digitalWrite(13,LOW);
		}

for(cond_ini; cond_d; modif) {
// repeticiones si cumple
}
	Desde la condición inicial “cond_ini”, **hasta** condición “cond_d”, se realiza lo que hay entre las llaves modificando la variable según “modif”.
	Operadores válidos: =,>,<,>=,<=
		// Parpadea 10seg el LED13
		for (i=1; i<=10;i=i+1) {
		digitalWrite(13,HIGH);
		delay(500); //espero 1s
		digitalWrite(13,LOW);
		delay(500);
		}

while(condición) {
// si se cumple
}
	Mientras se cumpla la condición, se repite lo que haya entre las llaves	// Lee 10 valores cada 1ms
		int i[];
		while (i<11) {
		delayMicroseconds(998);
		lee[i]=digitalRead(1);
		i=i+1;
		}

switch(variable) {
case valor1:
// Hacer si cierto
break;
case valor2:
// Hacer si cierto
break;
...
case valorn:
// Hacer si cierto
break;
default:
// Hacer si no cierto ninguna
break;
	Para la “variable” se hacen las sentencias según el valor, si no coincide ninguna, se realiza lo que hay en default	switch(barrido) {
		case ‘1’:
		lee=digitalRead(1);
		break;
		case ‘2’:
		lee=digitalRead(2);
		break;
		case ‘3’:
		lee=digitalRead(3);
		break;
		default:
		lee=digitalRead(10);
		break;
		}

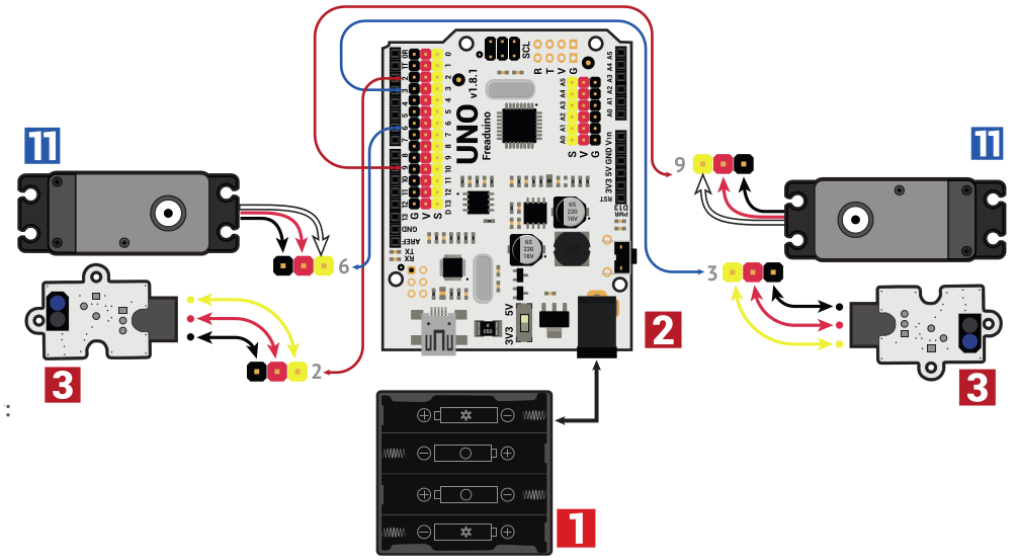
}		
INTERRUPCIONES	Permiten interrumpir el normal desarrollo para ejecutar la función correspondiente.	

attachInterrupt(i,f,m);
 	
detachInterrupt(i);
	Asigna una función a una señal externa de interrupción:
	• i: interrupción 0->pin1 y 1->pin2
	• f: nombre de la función a ejecutar
	• m: modo de detectar la interrupción
	  ○ LOW -> pin a 0
	  ○ CHANGE -> pin cambia
	  ○ RISING -> pin pasa 0->1
	  ○ FALLING-> pin pasa 1->0
	• detachInterrupt: elimina asignación
		volatile int state=LOW; //cambiara valor
		void setup()
		{
		pinMode(13, OUTPUT); //LED
		attachInterrupt(0, cambia(), CHANGE);
		//Interrupcion en pin1
		}
		void loop() { digitalWrite(13, state); }
		void cambia()
		{
		state = !state; // si es 1 -> 0 o 0 -> 1
		}

Tabla de frecuencias y notas (tone)

Octava	Do (C)	Do#	Re (D)	Re#	Mi (E)	Fa (F)	Fa#	Sol (G)	Sol#	La (A)	La#	Si (B)
1	33	35	37	39	41	44	46	49	52	55	58	62
2	65	69	73	78	82	87	93	98	104	110	117	123
3	131	139	147	156	165	175	185	196	208	220	233	247
4	262	277	294	311	330	349	370	392	415	440	466	494
5	523	554	587	622	659	698	740	784	831	880	932	988
6	1047	1109	1175	1245	1319	1397	1480	1568	1661	1760	1865	1976

Tarjeta Arduino UNO compatible (Funduino)



Los servomotores requieren la librería “servo.h” y asignarle un pin. Si se asigna: 90 → parar, 0→ sentido horario y 180→ sentido antihorario.

```
#include <Servo.h>  //Incluye la librería Servo
Servo servo_9;      //Creo un objeto servo
void setup()
{
  servo_9.attach(9); //Servo conectado en el pin9
}
void loop()
{
  servo_9.write(90); //Indicamos que el servo se pare
  delay(1000);       //Espero 1s
}
```

Los sensores IR ofrece HIGH (5V) cuando hay blanco y LOW (0V) cuando tiene cerca algo negro.