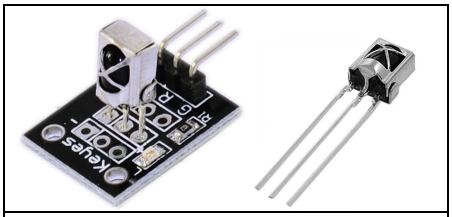
SENSOR RECEPTOR INFRARROJO IR VS1838B
(Para Control Remoto)

Actualmente la mayoría de nuestros equipos electrónicos se manejan con mandos a distancia (dispositivo de control IR) o simplemente control remoto. Estos controles funcionan gracias a un sistema simple como es la luz infrarroja.

Entonces básicamente estos aparatos se activan o realizan alguna función con solo presionar un botón. En los comercios de electrónica podemos encontrar controles específicos para usar con Arduino y placas similares, pero hay que recordar podremos usar casi cualquier control remoto que tengamos en casa.

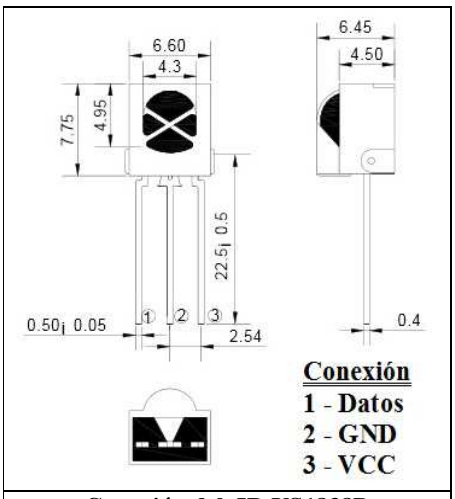


Receptor Infrarrojo IR VS1838 y VS1838B

En esta guía veremos las pautas básicas y algunos ejemplos en que usaremos controles comunes, que usamos en casa para controlar la TV, DVD, o estéreos entre otros.

Es muy práctico recordar que, al emplear luz como medio de transmisión de los mandos a distancia, es necesaria una línea de visión directa con el receptor aunque el haz de luz, puede ser reflejado por superficies reflectantes, como espejos.

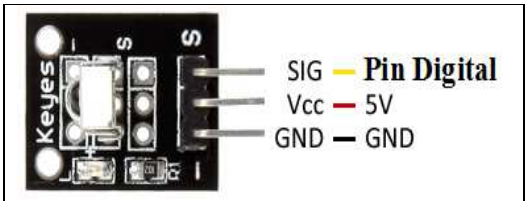
Es posible ver la luz del mando mirando el LED infrarrojo con una cámara digital, por ejemplo de un celular. La luz del led IR se verá como un resplandor morado. Esto puede ser útil para detectar si el mando funciona correctamente.



Conexión del IR VS1838B

Y por ultimo... La luz infrarroja es adecuada para hacer mandos a distancia porque utilizan luz en una frecuencia que no tienen consecuencias en los tejidos vivos. Menos impacto que la luz visible. El alcance es limitado, típicamente inferior a 3m. La distancia depende fuertemente del ángulo de emisión, disminuyendo rápidamente a medida que nos desviamos de la dirección frontal.

Y con respecto a lo que mas nos interesa, debemos saber que podremos emplear un mando a distancia como control remoto para controlar Arduino. Por ejemplo, encender o apagar un sistema de luces o sonido, encender o apagar un dispositivo por relé, controlar un robot o un vehículo.



Conexión del IR VS1838 - Vista Frontal

Por otro lado, también es posible emplear Arduino para clonar un mando a distancia y, por ejemplo, controlar el encendido de la televisión, temporizar el encendido de un equipo de música, o encender el aire acondicionado a través de Internet.

A continuación, veamos algunas características de los dos modelos de receptor que usaremos. Si bien parecen muy distintos, para lo que nosotros necesitamos podemos decir que son prácticamente iguales.

Tensión de alimentación:	2.7 a 5.5 V
Frecuencia de trabajo:	38 kHz
Consumo de corriente:	0.4-1.5 mA
Peso	2 a 5 gramos

MUY IMPORTANTE: Me doy cuenta si estoy viendo nuestro controlador de adelante, si lo estoy viendo 2 letras "S", del lado derecho. Una "S" mas o menos a la mitad, y la otra abajo (también a la derecha), Pero si veo el modulo desde atrás, no veré ninguna letra.

COMO USAR UN RECEPTOR INFRARROJO

(Recibir Comandos Desde Un CONTROL REMOTO)

Para poder usar un control remoto, primero debemos entender los comandos que nos envía (lo que nos dice) y luego usarlos según nos convenga.

En cuanto a la programación, existen múltiples librerías para reconocer y usar mandos a distancia con Arduino. Acá usaremos dos librerías, **ambas las encontraran en nuestra página de librerías**: La primera (IRremote-master) nos facilita mucho la programación, y permite usar cualquiera de los pines digitales que nos brinda Arduino, además de ser muy completa, sin embrago ocupa bastante memoria y no es de las mas rápidas al momento de funcionar. Luego la segunda (IRLremote), es muy rápida, ya que nos permite usar directamente las interrupciones de Arduino, pero, no es tan completa (reconoce pocos modelos de controles remotos).

1) Reconocimiento de los comandos enviados por un Control Remoto.

Programa que reconoce los comandos (teclas) de un control remoto, y nos muestra por medio del monitor del puerto serie, el código (en Hexadecimal) de cada techa presionada en el control.
Es importante destacar, que el código que veremos cuando presionamos una tecla del control remoto, es el que deberemos usar para que nuestro dispositivo reconozca y ejecute una acción en consecuencia.

Librería: IRremote-master (Programa "200_Control_Remoto_001")

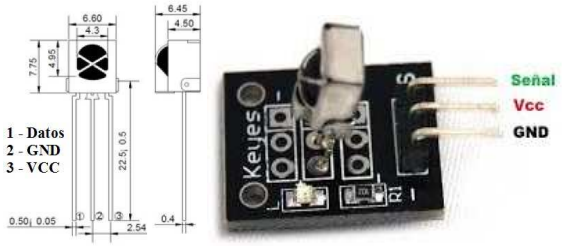
```
#include <IRremote.h>

int PIN_Ctrl_Remoto = 11; // Pin de recepción
IRrecv irrecv(PIN_Ctrl_Remoto);
decode_results Comando;

void setup( ) {
  Serial.begin(9600); //inicia puerto serial
  while (!Serial) {
    ; // Esperamos puerto serie este abierto.
  }
  Serial.println("Iniciando Modulo IR VS1838");
  irrecv.enableIRIn( ); // Activa recepción
}

void loop( ) {
 if (irrecv.decode(&Comando)) {

   // imprime código de tecla en hexadecimal
    Serial.print("Codigo Comado (Tecla): ");
    Serial.println(Comando.value, HEX);

   // imprime código de Marca que usamos
    Serial.print("Marca/Tipo: ");
    Serial.println(Comando.decode_type);
    Serial.println("====================");

   // Se preparar para recibir el siguiente valor
    delay(500);
    irrecv.resume( );
  }
```

Librería: IRLremote (Interrupciones) (Programa "200_Control_Remoto_002")

```
#include "IRLremote.h"

// Arduino interrupcion 0: Pin 2
//           interrupcion 1: Pin 3
const int Nro_Interrupcion = 1;

// Variable para guardar datos del protocolo
uint8_t Cod_Protocolo_IR = 0;

uint16_t IRAddress = 0;

// Variable para guardar codigo de Coamdo
uint32_t Cod_Commando_IR = 0;

void setup( ){
    Serial.begin(9600);
    Serial.println("Iniciando IR VS1838b");
    IRLbegin<IR_ALL>(Nro_Interrupcion);
  }

void IREvent(uint8_t protocol,
             uint16_t address,
             uint32_t command){
  // Recogemos los valores y nos volvemos
  Cod_Protocolo_IR = protocol;
  IRAddress = address;
  Cod_Commando_IR = command;
}

void loop( ){
  // Guardo Dirección interrupción Original
```

```
}
```

Recordar que cada Receptor Infrarrojo tiene su propia forma de conectar:

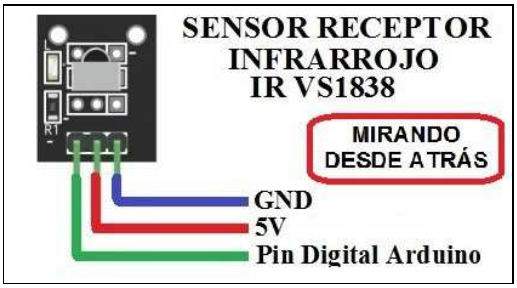


Por si no lo has notado, los pines correspondientes al GND y VCC están invertidos

```
    uint8_t oldSREG = SREG;
    cli( );
    if(Cod_Protocolo_IR){ //Si Hay protocolo
       Serial.print("Protocolo:");
       Serial.println(Cod_Protocolo_IR);
       Serial.print("Direccion :");
       Serial.println(IRAddress, HEX);
       Serial.print("Codigo Comando:");
    Serial.println(Cod_Commando_IR, HEX);
       Cod_Protocolo_IR = 0;
       Serial.println("=================");
    }
// restablesco Dirección interrupción Original
    SREG = oldSREG;
}
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Sensor Receptor Infrarrojo Ir Vs1838 - 3 Cables de conexión Macho/Macho - 1 placa Protoboard - 1 Cable USB - 1 Placa Arduino.

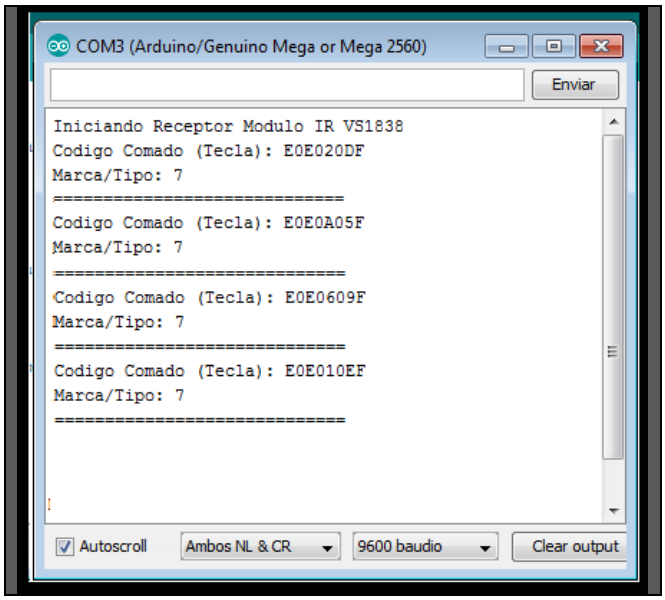


El circuito de conexión corresponde al programa "200_Control_Remoto_001".

Los cables deberán ser conectados: En este caso, la conexión de este primer ejemplo es muy simple. Según se muestran de arriba a abajo. Cable Azul Oscuro: conectar a GND. Cable Rojo: conectar a 5v. Cable Verde: Conectar al Pin Digital que se haya configurado en el programa (en este caso 11).
Si decidieras probar el segundo programa ("**200_Control_Remoto_002**") deberás conectar el Cable Verde al Pin Digital 3 (según lo configurado en el programa).

Ahora a Trabajar

Ya cargado el código del programa "**200_Control_Remoto_001**" en nuestra placa Arduino, abrimos la ventana del monitor serie (que se encuentra en la parte superior derecha de nuestro IDE) y empezamos a presionar botones del control remoto. Así veremos el código de cada tecla.
Cada "**Código Comando**", es el código hexadecimal correspondiente a la tecla presionada en el control remoto. Es la que debemos anotar cuidadosamente, ya que la usaremos luego, cuando queramos reconocer los comandos dados al dispositivo que queramos controlar por medio del control remoto.



La "**Marca/Tipo**" es la Marca del Control Remoto o Tipo de control. Hay que tener en cuanta que cada fabricante pone códigos distintos a las teclas. Es por eso que el control de nuestra TV no sirve para otra TV, a menos que sea otra idéntica.

2) Reconocimiento de los comandos y teclas del Control Remoto.

Este programa nos permitirá reconocer el código (en Hexadecimal) de los comandos (teclas) de un control remoto. Es importante destacar, que ese código que detectamos, deberá reemplazar los que vemos en el programa entre las líneas 6 y 22. Solo reemplazaremos la parte que esta a partir del "**0x**". Es posible que su control no tenga algunas teclas que acá usamos y en su lugar tenga otras teclas. Simplemente reemplácelas en el código

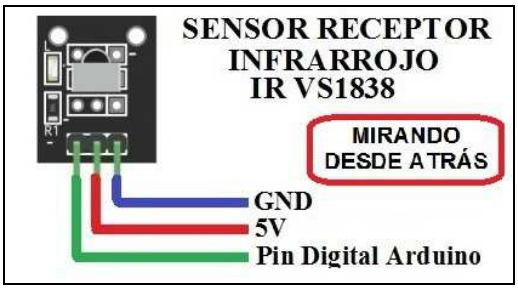
(Programa "200_Control_Remoto_003")

```
1   #include <IRremote.h>
2
3   // Pin que usaremos para leer nuestro SENSOR IR
4   const int PIN_Ctrl_Remoto = 9;
5
6   const long int Tecla_UP = 0xE0E006F9;
7   const long int Tecla_LEFT = 0xE0E046B9;
8   const long int Tecla_OK = 0xE0E016E9;
9   const long int Tecla_RIGHT = 0xE0E0A659;
10  const long int Tecla_DOWN = 0xE0E08679;
11  const long int Tecla_1 = 0xE0E020DF;
12  const long int Tecla_2 = 0xE0E0A05F;
13  const long int Tecla_3 = 0xE0E0609F;
14  const long int Tecla_4 = 0xE0E010EF;
15  const long int Tecla_5 = 0xE0E0906F;
16  const long int Tecla_6 = 0xE0E050AF;
17  const long int Tecla_7 = 0xE0E030CF;
18  const long int Tecla_8 = 0xE0E0B04F;
19  const long int Tecla_9 = 0xE0E0708F;
20  const long int Tecla_0 = 0xE0E08877;
21  const long int Tecla_ASTERISCO = 0xA70;
22  const long int Tecla_POWER_ON = 0xE0E040BF;
23
24  // No Cambiar este Código
25  const long int Tecla_REPEAT = 0xFFFFFFFF;
26
27  IRrecv irrecv(PIN_Ctrl_Remoto);
28  decode_results Comando;
29
30  void setup( ){
31    Serial.begin(9600);
32    while (!Serial) {
33     ; // Esperamos que el puerto serie este abierto.
34    }
35    Serial.println("Iniciando Modulo IR VS1838");
36    irrecv.enableIRIn( );
37  }
38
39  void loop( ){
40    if (irrecv.decode(&Comando)){
41      Serial.print("Codigo Comado (Tecla): ");
42      Serial.print(Comando.value, HEX);
43      Serial.print(" -- ");
44      switch (Comando.value){
45      case Tecla_UP:{
46        Serial.println("Tecla Arriba");
47        }break;
48      case Tecla_LEFT:{
49        Serial.println("Tecla Derecha");
50        }break;
51      case Tecla_OK:{
52        Serial.println("Tecla OK");
53        }break;
54      case Tecla_RIGHT:{
55        Serial.println("Tecla Izquierda");
56        }break;
57      case Tecla_DOWN:{
58        Serial.println("Tecla Abajo");
59        }break;
60      case Tecla_1:{
61        Serial.println("Tecla 1");
62        }break;
63      case Tecla_2:{
64        Serial.println("Tecla 2");
65        }break;
66      case Tecla_3:{
67        Serial.println("Tecla 3");
68        }break;
69      case Tecla_4:{
70        Serial.println("Tecla 4");
71        }break;
72      case Tecla_5:{
73        Serial.println("Tecla 5");
74        }break;
75      case Tecla_6:{
76        Serial.println("Tecla 6");
77        }break;
78      case Tecla_7:{
79        Serial.println("Tecla 7");
80        }break;
81      case Tecla_8:{
82        Serial.println("Tecla 8");
83        }break;
84      case Tecla_9:{
85        Serial.println("Tecla 9");
86        }break;
87      case Tecla_0:{
88        Serial.println("Tecla 0");
89        }break;
90      case Tecla_ASTERISCO:{
91        Serial.println("Tecla ASTERISCO");
92        }break;
93      case Tecla_POWER_ON:{
94        Serial.println("Tecla Prender/Apagar");
95        }break;
96      case Tecla_REPEAT:{
97        Serial.println("Error de Lectura - Repita");
98        }break;
99      default: {
100   Serial.println("CODIGO TECLA NO RECONOCIDO");
101       }break;
102     } // Fin switch( )
103     delay(500);
104     irrecv.resume( );
105   }// Fin del if(
106 }// Fin del loop
```

CIRCUITO PARA NUESTRO PROYECTO

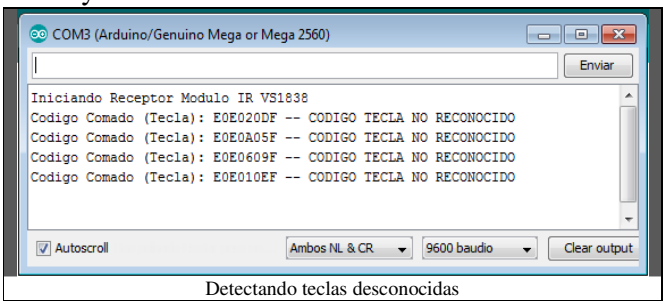
Lista de Materiales: 1 Sensor Receptor Infrarrojo Ir Vs1838 - 3 Cables de conexión Macho/Macho - 1 placa Protoboard- 1 Cable USB - 1 Placa Arduino.

Los cables deberán ser conectados: En este caso, la conexión de este primer ejemplo es muy simple. Según se muestran de arriba a abajo. Cable Azul Oscuro: conectar a GND. Cable Rojo: conectar a 5v. Cable Verde: Conectar al Pin Digital que se haya configurado en el programa (en este caso 11).



Ahora a Trabajar

Una vez hayamos cargado el código del programa "**200_Control_Remoto_003**" en nuestra placa Arduino, abrimos la ventana del monitor serie (que se encuentra en la parte superior derecha de nuestro IDE) y empezamos a presionar los botones del control remoto. Así veremos el código detectado de cada tecla y si es reconocida o no.

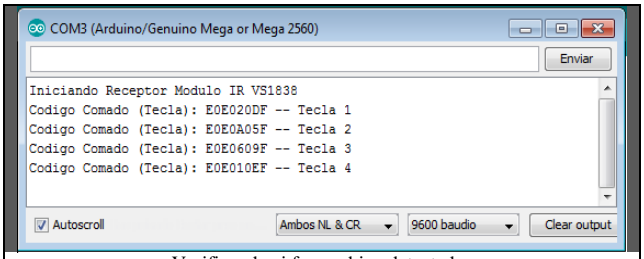


Detectando teclas desconocidas

Entonces cuando una tecla no sea reconocida, solo habrá que tomar el código de la tecla que vemos en el monitor del puerto serie y reemplearlo en el programa. Esto debe hacerse tecla por tecla.

En la imagen de la izquierda, presioné en orden, las teclas correspondientes a los numero 1, 2, 3 y 4. Me muestra el código de las teclas y me avisa que el programa no las reconoce. Entonces copio esos códigos y los escribo en el programa, compilo y envío a nuestro Arduino para verificar si todo esta bien.

El resultado podemos verlo en la imagen de la derecha, teclas reconocidas. ==========>



Verificando si fueron bien detectadas

El proceso de detección debe realizarse tecla por tecla y con todas las que necesitemos para controlar el dispositivo desde nuestro control remoto.

3) Prender y apagar 3 Leds independientemente desde un control remoto.

Desde un control remoto, prender y apagar 3 Leds.
Las Acciones que se deben poder realizar desde el control remoto serán:

Tecla 1- Prender / Apagar el Leds 02.	Tecla 2- Prender / Apagar el Leds 02.
Tecla 3- Prender / Apagar el Leds 02.	Tecla Power- Prender / Apagar Todos los Leds simultáneamente

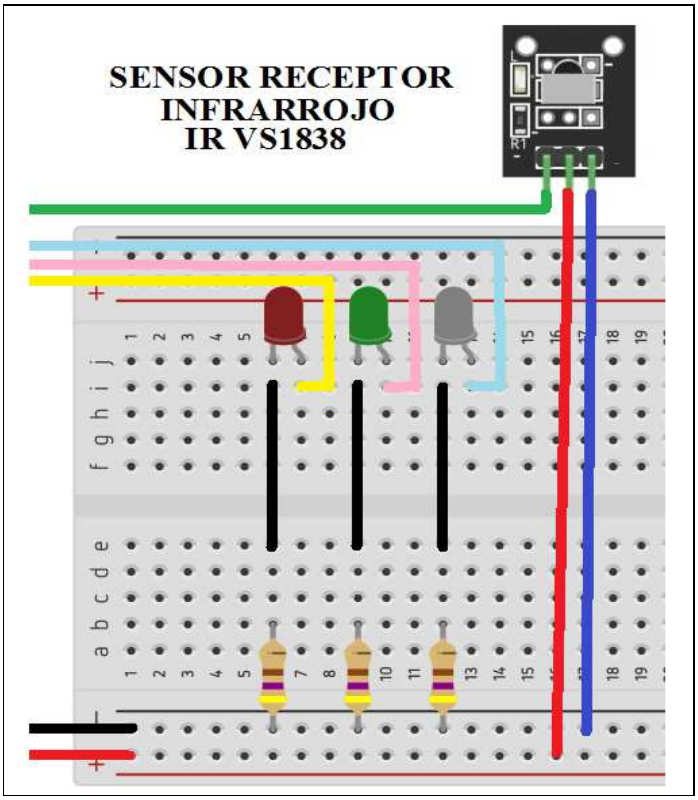
(Programa "200_Control_Remoto_004_3_Leds")

```
1  #include <IRremote.h>
2
3  const int PIN_Ctrl_Remoto = 9;
4
5  const long int Tecla_1 = 0xE0E020DF;
6  const long int Tecla_2 = 0xE0E0A05F;
7  const long int Tecla_3 = 0xE0E0609F;
8  const long int Tecla_POWER_ON = 0xE0E040BF;
9
10 // No Cambiar este Código
11 const long int Tecla_REPEAT = 0xFFFFFFFF;
12
```

```
38 void loop( ){
39   if (irrecv.decode(&Comando)){
40     Serial.print("Codigo Comado (Tecla): ");
41     Serial.print(Comando.value, HEX);
42     Serial.print(" -- ");
43     switch (Comando.value){
44     case Tecla_1:{
45       Estado_Led_01 = 1-Estado_Led_01;
46       digitalWrite(PinLed_01,Estado_Led_01);
47       }break;
48     case Tecla_2:{
49       Estado_Led_02 = 1-Estado_Led_02;
```

```
13  IRrecv irrecv(PIN_Ctrl_Remoto);
14  decode_results Comando;
15
16  const int PinLed_01 =  3;
17  const int PinLed_02 =  5;
18  const int PinLed_03 =  7;
19
20  int Estado_Led_01=0,
21      Estado_Led_02=0,
22      Estado_Led_03=0,
23      Led_Estado_Grupal=0;
24
25  void setup( ){
26    pinMode(PinLed_01,OUTPUT);
27    pinMode(PinLed_02,OUTPUT);
28    pinMode(PinLed_03,OUTPUT);
29
30    Serial.begin(9600);
31    while (!Serial) {
32     ; // Esperamos que el puerto serie este abierto.
33     }
34
35    Serial.println("Iniciando Modulo IR VS1838");
36    irrecv.enableIRIn( );
37  }
---
50          digitalWrite(PinLed_02, Estado_Led_02);
51          }break;
52        case Tecla_3:{
53          Estado_Led_03 = 1-Estado_Led_03;
54          digitalWrite(PinLed_03, Estado_Led_03);
55          }break;
56        case Tecla_POWER_ON:{
57          Led_Estado_Grupal= 1-Led_Estado_Grupal;
58          Estado_Led_01 = Led_Estado_Grupal;
59          Estado_Led_02 = Led_Estado_Grupal;
60          Estado_Led_03 = Led_Estado_Grupal;
61          digitalWrite(PinLed_01,Led_Estado_Grupal);
62          digitalWrite(PinLed_02,Led_Estado_Grupal);
63          digitalWrite(PinLed_03,Led_Estado_Grupal);
64          }break;
65        case Tecla_REPEAT:{
66          Serial.println("Error de Lectura - Repita");
67          }break;
68        default: {
69    Serial.println("CODIGO NO RECONOCIDO");
70          }break;
71       } //Fin switch( )
72       delay(500);
73       irrecv.resume( );
74     }//Fin del if( )
75  }// Fin del loop
```

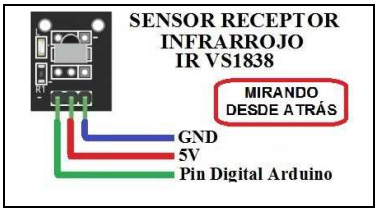
CIRCUITO PARA NUESTRO PROYECTO



Lista de Materiales: 1 Sensor Receptor Infrarrojo Ir Vs1838 - 3 Leds de colores Variados, 3 resistencias de 470Ω. - 8 Cables de conexión Macho/Macho - 3 cables de conexión Macho/Hembra - 1 placa Protoboard- 1 Cable USB - 1 Placa Arduino.

Los cables deberán ser conectados, ver a la izquierda, desde arriba hacia abajo: Cable Verde: Al pin que se haya configurado el Sensor IR, que para nuestro ejemplo es el pin 7. Cable Celeste: Pin de Control para el Led numero 3. Cable Rosado: Pin de Control para el Led numero 2. Cable Amarillo: Pin de Control para el Led numero 1. Cable Negro: conectar a GND. Cable Rojo: Conectar a 5V.

Recordatorio de la forma para conectar el modulo receptor que estamos usando.



4) Controlar 2 motores desde un control remoto.

Desde un control remoto, controlar los dos motores pertenecientes a un vehicolo.
Las Acciones que se deben poder realizar el vehiculo al obedecer los comandos enviados desde el control remoto serán:

- Prende y apagar el vehiculo.	- Adelante / Atrás
- Izquierda y Derecha	

(Programa "200_Control_Remoto_005_2_MotoresCC")

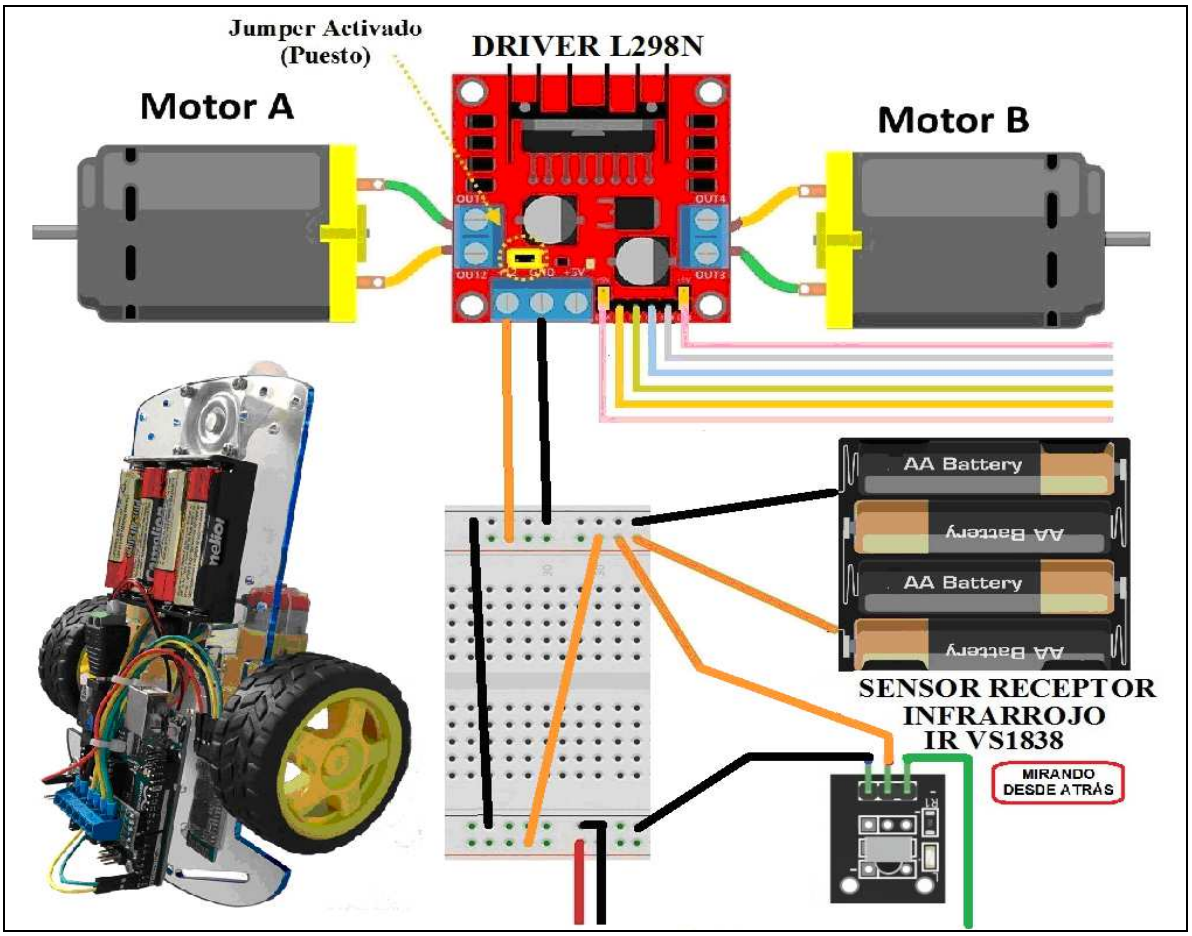
```
#include <IRremote.h> // Libreria usada para reconocer Ctrl Remoto.

const int PIN_Ctrl_Remoto = 11;

// Valores o codigos de las Teclas Usadas
const long int Tecla_UP = 0xE0E006F9;
const long int Tecla_LEFT = 0xE0E046B9;
const long int Tecla_RIGHT = 0xE0E0A659;
const long int Tecla_DOWN = 0xE0E08679;
const long int Tecla_POWER_ON = 0xE0E040BF;

// No Cambiar este Codigo
const long int Tecla_REPEAT = 0xFFFFFFFF;

IRrecv irrecv(PIN_Ctrl_Remoto);
decode_results Comando;

//================================================================================
int PinIN1 = 3,      //El pin 3 de Arduino se conecta con el pin In1 del L298N
    PinIN2 = 5,      //El pin 5 de Arduino se conecta con el pin In2 del L298N
    PWM_Izquierdo_ENA = A0; //Pin Analógico Arduino (Control de Velocidad)
int PinIN3 = 7,    //Pin Digital que conecta Arduino con el pin In3 del L298N
    PinIN4 = 9,    //Pin Digital que conecta Arduino con el pin In4 del L298N
    PWM_Derecho_ENB = A1; //Pin Analógico Arduino (Control de Velocidad)

int V = 250; //Declaramos una variable para guardar la velocidad
//================================================================================

int Estado; // Variable usada para reconocer el estado (Prendido / Apagado) del vehiculo

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

/***  Estas lineas deben estar desactivadas cuando el vehiculo no este conectado a la PC ***/
//   while (!Serial) {
//    ; // Esperamos puerto serie este abierto.
//   }
/*******************************************************************************/

  Serial.println("Iniciando Modulo IR VS1838");
  irrecv.enableIRIn( );

  pinMode(PinIN1, OUTPUT);
  pinMode(PinIN2, OUTPUT);
  pinMode(PWM_Izquierdo_ENA, OUTPUT);
  pinMode(PinIN3, OUTPUT);
  pinMode(PinIN4, OUTPUT);
  pinMode(PWM_Derecho_ENB, OUTPUT);
  Estado = 0; // El Vehiculo inicia Apagado
}

void Adelante_MA(int Veloc){ // Motor Izquierdo
 digitalWrite(PinIN1,LOW);
 digitalWrite(PinIN2,HIGH);
 analogWrite(PWM_Izquierdo_ENA, Veloc);
 }
```

```
57  void Adelante_MB(int Veloc){ // Motor Derecho
58   digitalWrite(PinIN3,LOW);
59   digitalWrite(PinIN4,HIGH);
60   analogWrite(PWM_Derecho_ENB, Veloc);
61   }
62  void Atras_MA(int Veloc){ // Motor Izquierdo
63   digitalWrite(PinIN1,HIGH);
64   digitalWrite(PinIN2,LOW);
65   analogWrite(PWM_Izquierdo_ENA, Veloc);
66   }
67  void Atras_MB(int Veloc){ // Motor Derecho
68   digitalWrite(PinIN3,HIGH);
69   digitalWrite(PinIN4,LOW);
70   analogWrite(PWM_Derecho_ENB, Veloc);
71   }
72  void Parar_MA( ){ // Motor Izquierdo
73   digitalWrite(PinIN1,LOW);
74   digitalWrite(PinIN2,LOW);
75   analogWrite(PWM_Izquierdo_ENA, 0);
76   }
77  void Parar_MB( ){ // Motor Derecho
78   digitalWrite(PinIN3,LOW);
79   digitalWrite(PinIN4,LOW);
80   analogWrite(PWM_Derecho_ENB, 0);
81   }
82  void Avanzar(int Velocidad){
83     Adelante_MA(Velocidad);
84     Adelante_MB(Velocidad);
85  }
86  void Retroceder(int Velocidad){
87   Atras_MA(Velocidad);
88   Atras_MB(Velocidad);
89  }
90  void Alto( ){
91   Parar_MA( );
92   Parar_MB( );
93  }
94  void Derecha(int Velocidad){
95   Adelante_MA(Velocidad);
96   Parar_MB( );
97  //  Atras_MB(Velocidad);
98  }
99  void Izquierda(int Velocidad){
101  Parar_MA( );
102 //  Atras_MA(Velocidad);
103  Adelante_MB(Velocidad);
104 }
105
106 void loop( ){
107   if (irrecv.decode(&Comando)){
108     switch (Comando.value){
109     case Tecla_UP:{
110       Serial.println("Tecla Arriba");
111       Avanzar(V);
112       }break;
113     case Tecla_DOWN:{
114       Serial.println("Tecla Abajo");
115       Retroceder(V);
116       }break;
117     case Tecla_LEFT:{
118       Serial.println("Tecla Derecha");
119       Derecha(V);
120       }break;
121     case Tecla_RIGHT:{
122       Serial.println("Tecla Izquierda");
123       Izquierda(V);
```

```
124         }break;
125       case Tecla_POWER_ON:{
126         Serial.println("Tecla Prendido / Apagado");
127         Estado = 1 - Estado;
128         if(Estado == 0) Alto( );
129         else Avanzar(V); // Evito que se lea dos comando seguidos
130         delay(1000);
131         }break;
132       case Tecla_REPEAT:{
133         Serial.println("Error de Lectura - Repita");
134         }break;
135       default: {
136         Serial.println("CÓDIGO TECLA NO RECONOCIDO");
137         }break;
138       } //Fin switch( )
139       irrecv.resume( );
140     }//Fin del if( ) recibe comando
141   }// Fin del loop
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Sensor Receptor Infrarrojo Ir Vs1838 - 1 conector Plug Macho con Bornera para Alimentación de 5.5 x 2.1 - 12 cables conectores Macho/Hembra - 5 Cables conectores Macho/Macho - 1 Porta pilas de 4 Pilas AA - 2 Motores DC 6v con Caja reductora y Rueda goma - 1 Driver L298N - 1 placa Protoboard- 1 Cable USB - 1 Placa Arduino.



Los cables deberán ser conectados: *(Primer grupo, 6 Cables a la derecha de la imagen),* De arriba hacia abajo. Cable Rosado: Es el acelerador del motor B (ENB) y se debe conectar al Pin Analógico que se configure en el Programa (en este ejemplo A1). Cable Gris Claro: Conectar al Pin Digital que se haya configurado en el programa a IN4 (en este caso 9). Cable Celeste: Conectar al Pin Digital que se haya

configurado en el programa a IN3 (en este caso 7). Cable Verde Claro: Conectar al Pin Digital que se haya configurado en el programa a IN2 (en este caso 5). Cable Amarillo: Conectar al Pin Digital que se haya configurado en el programa a IN1 (en este caso 3). Cable Rosado: (último del grupo) Es el Acelerador del motor A (ENA) y se debe conectar al Pin Analógico que se configure en el Programa (en este ejemplo A0).

(Segundo grupo, 3 Cables al pie de la imagen), De izquierda a derecha. Los Cables Rojo y Negro: Serán utilizados para alimentar nuestra placa Arduino, através del conector Plug Macho con Bornera (Donde conectaremos los cable - Ver imagen de la derecha). Cable Verde: Correspondiente al Receptor Infrarrojo, debe conectarse al Pin configurado para recibir los datos del Censor (en este ejemplo 11).

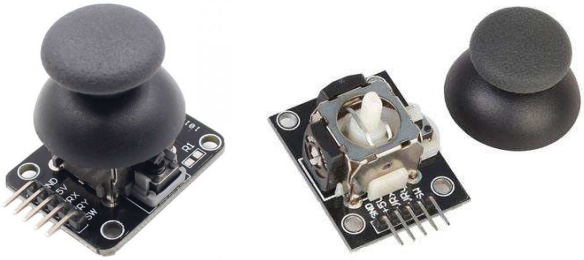
Puedes NO conectar los cables correspondientes al control de velocidad (cables Rosados correspondientes a los Pines A0 y A1), y dejar los Jumpers que trae el puente H por defecto. Los motores quedaran fijos con la máxima velocidad.

IMPORTANTE: Mientras nuestra placa Arduino este conectada a la PC por el puerto USB, NO DEBE estar conectada con el conector Plug Macho. Este solo debe conectarse cuando se desconecte el cable USB y deba mantener prendida nuestro Arduino.

Por otro lado, SI puede estar conectado el porta Pilas mientras Arduino Envía o Recibe datos desde la PC. Lo que resulta muy practico ya que puede realizar todas las prueba requeridas.

JOYSTICK ANALÓGICO

Un joystick analógico es un periférico de entrada, un sencillo controlador que podemos añadir a nuestros proyectos, se puede describir como una palanca de mando que gira sobre una base e informa su ángulo o dirección al dispositivo con el que se encuentra conectado.

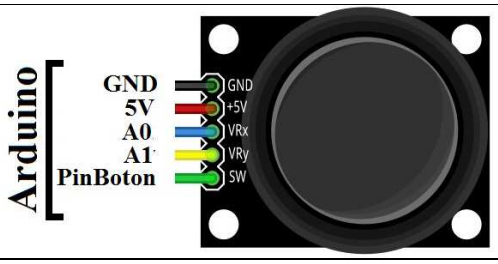


Vistas de un Joystick Analógico

La trasmisión de la información es muy rápida y nos permite mover en cualquier dirección el cursor en una pantalla de computadora o videojuego, controlar un brazo robótico o un vehiculo.

Un Joystick esta compuestos por una serie de dispositivos, unidos entre si, de una forma magistral. Generalmente al desarmar un joystick veremos dos potenciómetros a 90º que transforman el movimiento de los dos ejes, X e Y del mando en una señal eléctrica proporcional a su posición y que además suele incluir un botón. Y mirándolos detenidamente, veremos que generalmente suelen tener 5 pines: X, Y, botón, 5V y GND. En realidad ya usamos todos estos componentes previamente y la única curiosidad del joystick es que resulta un elemento muy cómodo para posicionar algo, aunque no sea demasiado preciso.

En esta guía usaremos uno de los mas difundidos en el mercado actualmente, y muy barato y de fácil conexión (Ver imagen). y entre sus principales características podemos enumerar:

- Voltaje: 5V - Valor Potenciómetro interno: 10k - Dimensiones: 4.0 cm x 2.6 cm x 3.2 cm (1.57 in x 1.02 in x 1.26 in). - Rango de temperaturas: 0 a 70 °C	

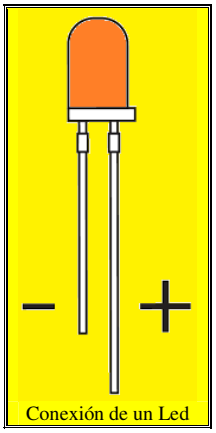
Sin más preámbulos, veamos los primeros ejemplos de código, con los que podremos comprender como incorporarlos a nuestros proyectos

5) Reconocimiento de los valores entregados por un Joystick.

Programa que reconoce los valores entregados por un Joystick, y a continuación se muestran através del monitor del puerto serie.

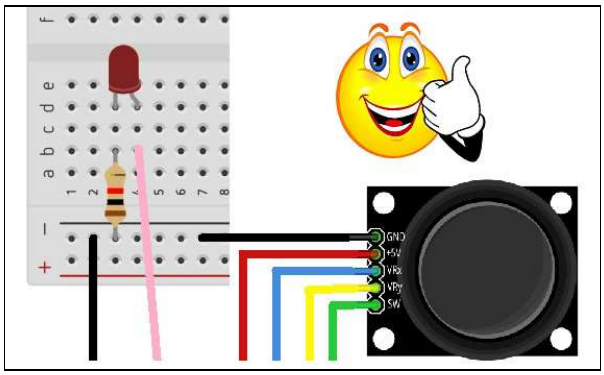
(Programa: "450_1_Joistick_01")

```
1  const int pinJoyX = A0; //Pin donde leeremos la Coordenada X del Joystick
2  const int pinJoyY = A1; //Pin donde leeremos la Coordenada Y del Joystick
3  const int pinJoyBoton = 9; //Pin donde leeremos Si es Presionado Botón del Joystick
4  const int pinLED = 5;   // Pin del Led que prendemos cuando es presionado el Botón
5
6  int Valor_X,
7      Valor_Y,
8      Valor_Boton;
9
10 void setup( ) {
11   Serial.begin(9600);
12    while (!Serial) {
13    ; // Esperamos puerto serie este abierto.
14    }
15   Serial.println("Programa Joystick 01 está Iniciando");
16
17   pinMode(pinJoyBoton , INPUT_PULLUP); //Activar resistencia Interna Pull Up
18   pinMode(pinLED, OUTPUT);
19 }
20
21 void loop( ) {
22   // Leemos valores Entregados por el Joystick
23   Valor_X = analogRead(pinJoyX);
24
25   delay(50);   //Esta Pausa es necesaria entre lecturas analógicas
26
27   Valor_Y = analogRead(pinJoyY);
28
29   Valor_Boton = digitalRead(pinJoyBoton);
30
31 /*** Prendo/Apago Led Asociado al Botón del Joystick ***/
32   if(Valor_Boton == LOW) digitalWrite(pinLED, HIGH);
33   else digitalWrite(pinLED, LOW);
34
35 /****** Mostrar valores por el Monitor Puerto Serie *****/
36   Serial.print("X:" );              // Estos Mensajes deberían ser eliminados
37   Serial.print(Valor_X);           // Una vez se comprenda bien el funcionamiento
38   Serial.print(" | Y: ");          // ya que hacen demasiado lento el proceso
39   Serial.print(Valor_Y);
40   Serial.print(" | Pulsador: ");
41   Serial.println(Valor_Boton);
42 /*********************************************/
43
44   delay(30;  // Solo para hacer un poquito más lento el programa
45 }
```



Conexión de un Led

CIRCUITO PARA NUESTRO PROYECTO



Lista de Materiales: 1 Joystick analógico para Arduino (de 5 pines), 1 Led, 1 Resistencia de 470Ω, 4 Cables conectores Hembra/ hembra, 1 cable conector Macho/Hembra, 6 Cables conectores Macho/Macho - 1 placa Protoboard- 1 Cable USB - 1 Placa Arduino.

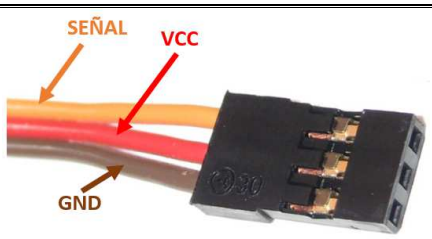
Los cables deberán ser conectados: abajo, de Izquierda a derecha, Cable Negro: Conectar a GND. Cable Rosado: Al pin usado para prender el Led (para nuestro ejemplo Pin 5). Cable Rojo: Conectar a +5V. Cable Azul: Conectar A0. Cable Amarillo: Conectar a A1. Cable Verde: Conectar al Pin Digital que se haya configurado el Botón del Joystick en el programa (en este caso Pin 9).

6) Controlar 2 Servos Motores y un Led con un Joystick.

Tomando los valores Entregados por el Joystick (Eje X, Eje Y, valor del Botón), Mover Ambos Servos, en forma independiente, entre 0 y 180 Grados. Por otro lado, prender el Led al presionar el botón.

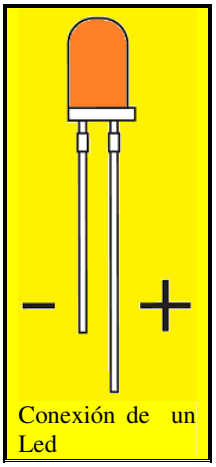
(Programa: "450_1_Joistick_02")

```
1   #include <Servo.h>                  // Incluye librería Servo
2
3   Servo A_Servo_X,  // Declaro dos Objetos Tipo Servo llamados Servo_X
4         A_Servo_Y;  // ..................................................... y Servo_Y
5
6   const int A_PinServo_X = 2,
7             A_PinServo_Y = 3;
8
9   int A_Angulo_X = 90, // Posición Inicial A_Servo_X
10    A_Angulo_Y = 90,  // Posición Inicial A_Servo_Y
11    A_Salto    = 4;  // Controla salto de movimiento Servos A
12
13  // Leeremos dos Valores del Joystick correspondientes al Eje X y Eje Y, los que usaremos para
14  // controlar 2 Servo Motores de un Brazo Robótico - Nos queda libre un Botón.
15  // IDEA: con 2 Joystick controlaremos 4 Servo Motores y 2 Botones.
16  int A_Pin_Eje_X = A0,
17      Valor_X,
18      A_Pin_Eje_Y = A1,
19      Valor_Y;
20
21  int A_pinJoyBoton = 6,
22    A_PinLED = 8,
23    Valor_Boton;
24
25  void setup( ){
26    Serial.begin(9600);
27    while (!Serial) {
28     ; // Esperamos que el puerto serie este abierto.
29     }
30    Serial.println("Programa Joistick Iniciando...");
31    A_Servo_X.attach(A_PinServo_X) ;  // Conectar A_Servo_X al pin
32    A_Servo_Y.attach(A_PinServo_Y) ;  // Conectar A_Servo_Y al pin
33    pinMode(A_pinJoyBoton , INPUT_PULLUP); //Activar resistencia Interna Pull Up
34    pinMode(A_PinLED, OUTPUT);
35  }
36
37  /*************************************************************/
38  /*** Esta función es apta para manejar Grúas y Brazos Robóticos ****/
39  int Calcular_Angulo( int Valor, int Angulo, int Salto ){   // Función SobreCargada
40      if( Valor < 400 ) Angulo = Angulo - Salto ;
```



Recuerda como Conectar un Servo Motor

```
41      else if( Valor > 600) Angulo = Angulo + Salto ;
42      if(Angulo < 1) Angulo = 1;
43      else if( Angulo > 179) Angulo = 179;
44      return(Angulo);
45  }
46  /****** Esta Función es apta para Manejar Autos y Drones *********/
47  int Calcular_Angulo( int Valor ){  // Función SobreCargada
48      int Angulo;
49      Angulo = (int) map( Valor, 0, 1024, 1, 179);
51      return(Angulo);
52  }
53  /************************************************************/
54
55  void loop( ){
56  /********** Leo Eje X del Joystick A **********/
57    Valor_X = analogRead(A_Pin_Eje_X);
58
59  /* Activar una de las dos funciones y ver como cambias el funcionamiento */
60    A_Angulo_X = Calcular_Angulo(Valor_X, A_Angulo_X, A_Salto);
61  //  A_Angulo_X = Calcular_Angulo( Valor_X );
62
63    delay (50);
64
65  /********** Leo Eje Y del Joystick A **********/
66    Valor_Y = analogRead(A_Pin_Eje_Y);
67
68  /* Activar una de las dos funciones y ver como cambias el funcionamiento */
69  //  A_Angulo_Y = Calcular_Angulo( Valor_Y, A_Angulo_Y, A_Salto);
70    A_Angulo_Y = Calcular_Angulo( Valor_Y );
71
72  /********** Leo Boton Joystick A **********/
73     Valor_Boton = digitalRead(A_pinJoyBoton);
74
75  /*** Prendo/Apago Led Asociado al Botón del Joystick A ***/
76     if(Valor_Boton == LOW) digitalWrite(A_PinLED, HIGH);
77     else digitalWrite(A_PinLED, LOW);
78
79  /*** Muevo Servos Asociados a los Ejes del Joystick A ***/
80    A_Servo_X.write(A_Angulo_X);
81    A_Servo_Y.write(A_Angulo_Y);
82
83    //Serial.print("A_Angulo X: ");   // Solo Activar para ver Valores de Rotación de cada servo
84    //Serial.print( A_Angulo_X );     //  luego desactivar ya que hace todo muy lento
85    //Serial.print(" A_Angulo Y: ");
86    //Serial.println( A_Angulo_Y );
87
88    delay (20);
89  }
```



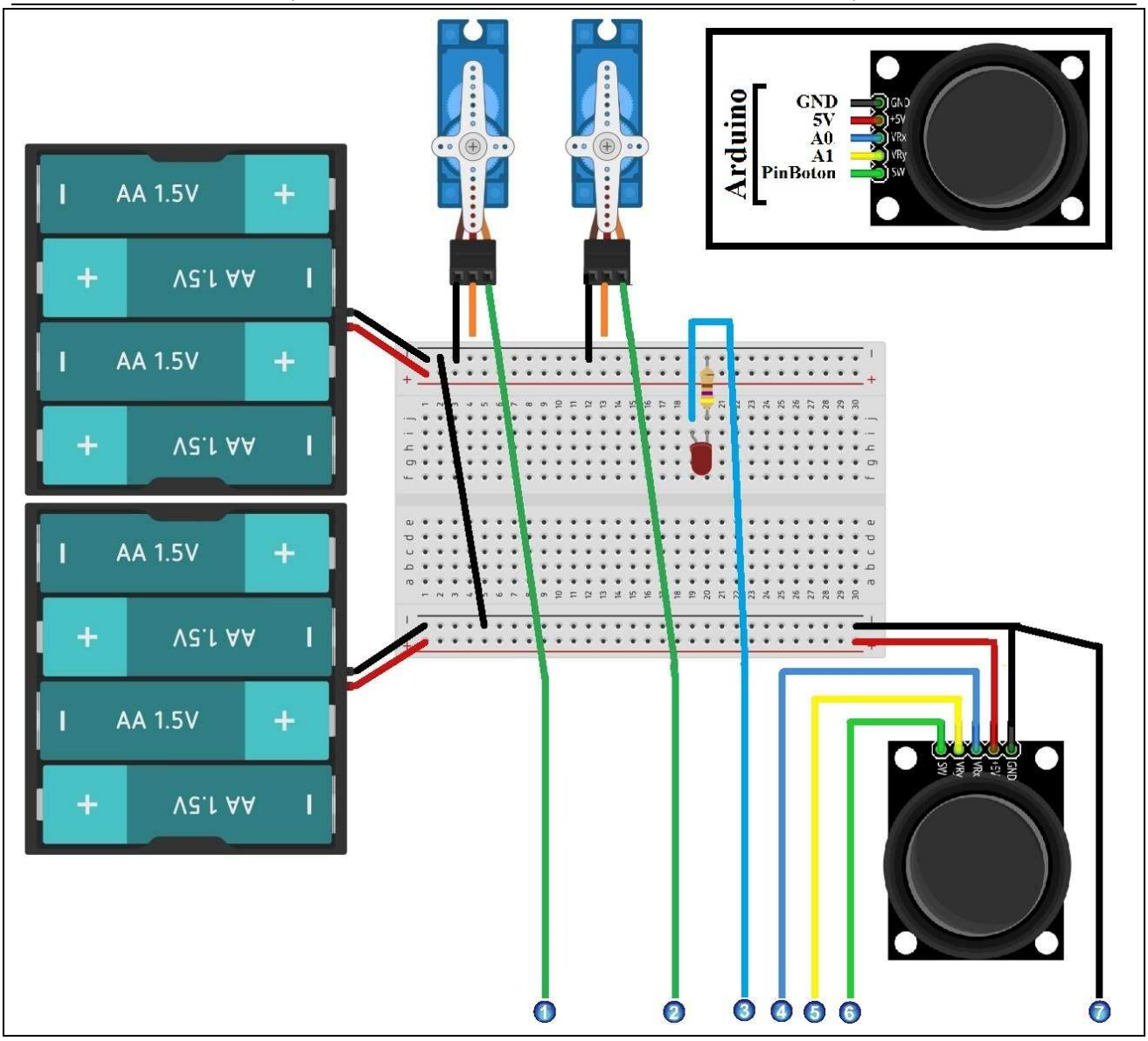
Conexión de un Led

Una Funciones está sobrecargada, cuando en el mismo programa hay dos o mas funciones que se llaman igual, pero hacen diferentes cosas y sus parámetros son diferentes en tipo o cantidad. En este Ejercicio, encontraremos que las funciones "**Calcular_Angulo**( )" están sobrecargadas, ya que una tiene tres parámetros y la otra solo dos, y en su interior hacen cosas muy diferentes.

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Joystick analógico para Arduino (de 5 pines), 2 Servos Motor MG90 o MG90S, 1 Led, 1 Resistencia de 470Ω, 4 Cables conectores Hembra/Hembra, 1 cable conector Macho/Hembra, 14 Cables conectores Macho/Macho, 2 Porta Pilas de 4 Pilas AA (1,5V), 8 Pilas AA (1,5V), 1 placa Protoboard, 1 Cable USB - 1 Placa Arduino

-o-



Los cables deberán ser conectados: abajo, de Izquierda a derecha, Cable 1 (verde): Conectar al Pin 2. Cable 2 (verde): Conectar al pin 3. Cable 3 (celeste): Conectar al pin 8. Cable 4 (azul): Conectar al pin Analógico A0. Cable 5 (amarillo): Conectar al pin Analógico A1. Cable 6 (verde): Conectar al Pin Digital que se haya configurado el Botón del Joystick en el programa (en este caso Pin 6). Cable 7 (negro) conectar a GND de Arduino (es el GND común de todo el circuito).

7) Controlar 4 Servos Motores y 2 Led con dos Joysticks.

Tomando los valores Entregados por dos Joysticks (Eje X, Eje Y y valor del Botón de cada Joystick), Mover 4 Servos, en forma independiente, entre 0 y 180 Grados. Por otro lado, controlar un Led con cada Joystick (al presionar el respectivo botón).

(Programa: "450_2_Joistick_01")

```
1  #include <Servo.h>                    // Incluye librería Servo
2
3  Servo A_Servo_X, // Declaro Cuatro Objetos Tipo Servo
4        A_Servo_Y, // .....
5        B_Servo_X, // .....
6        B_Servo_Y; // .....
7
8  const int A_PinServo_X = 2,
9            A_PinServo_Y = 3,
```

```
10            B_PinServo_X = 4,
11            B_PinServo_Y = 5;
12
13  /************* Joystick A *************/
14  int A_Angulo_X = 90, // Posición Inicial A_Servo_X
15      A_Angulo_Y = 90, // Posición Inicial A_Servo_Y
16      A_Salto    = 4;  // Controla el salto (cantidad de Unidades)por movimiento
17
18  /************* Joystick B *************/
19  int B_Angulo_X = 90, // Posición Inicial B_Servo_X
20     B_Angulo_Y = 90, // Posición Inicial B_Servo_Y
21     B_Salto    = 4;  // Controla el salto (cantidad de Unidades)por movimiento
22
23  // Leeremos dos Valores cada Joystick (Eje X, Eje Y y Botón) que usaremos para
24  //  controlar4 Servo Motores de un Brazo Robótico o Dron -
25  //  Nos quedan libres lo que pudieran controla los 2 Botones.
26  int A_Pin_Eje_X = A0,
27     A_Pin_Eje_Y = A1,
28     B_Pin_Eje_X = A2,
29     B_Pin_Eje_Y = A3,
30     Valor_X,  // Variable temporal usada para guardar el valor leído del Eje X
31     Valor_Y;  // Variable temporal usada para guardar el valor leído del Eje Y
32
33  int A_pinJoyBoton = 6,
34      B_pinJoyBoton = 7,
35      A_PinLED = 8,
36      B_PinLED = 9,
37      Valor_Boton; // Variable temporal usada para guardar el valor leído del Botón del Joystick
38
39  void setup(){
40     Serial.begin(9600);
41     while (!Serial) {
42      ; // Esperamos que el puerto serie este abierto.
43     }
44     Serial.println( "Programa 2 Joistick Iniciado..." );
45     A_Servo_X.attach(A_PinServo_X) ;  // Conectar A_Servo_X al pin
46     A_Servo_Y.attach(A_PinServo_Y) ;  // Conectar A_Servo_Y al pin
47     B_Servo_X.attach(B_PinServo_X) ;  // Conectar B_Servo_X al pin
48     B_Servo_Y.attach(B_PinServo_Y) ;  // Conectar B_Servo_Y al pin
49     pinMode(A_pinJoyBoton , INPUT_PULLUP); //Activar resistencia Interna Pull Up
51     pinMode(A_PinLED, OUTPUT);
52     pinMode(B_pinJoyBoton , INPUT_PULLUP); //Activar resistencia Interna Pull Up
53     pinMode(B_PinLED, OUTPUT);
54  }
55
56  /*********************************************************************/
57  /*** Esta función es apta para manejar Grúas y Brazos Robóticos ****/
58  int Calcular_Angulo( int Valor, int Angulo, int Salto){  // Función SobreCargada
59      if( Valor < 400 ) Angulo = Angulo - Salto ;
60      else if( Valor > 600) Angulo = Angulo + Salto ;
61      if(Angulo < 1) Angulo = 1;
62      else if( Angulo > 179) Angulo = 179;
63      return(Angulo);
64  }
65  /****** Esta Función es apta para Manejar Autos y Drones *********/
66  int Calcular_Angulo( int Valor){  // Función SobreCargada
67      int Angulo;
68      Angulo = (int) map( Valor, 0, 1024, 0, 180);
69      return(Angulo);
70  }
71  /*********************************************************************/
72
73  void loop(){
74  /********** Leo Eje X del Joystick A **********/
75    Valor_X = analogRead(A_Pin_Eje_X);
76
```

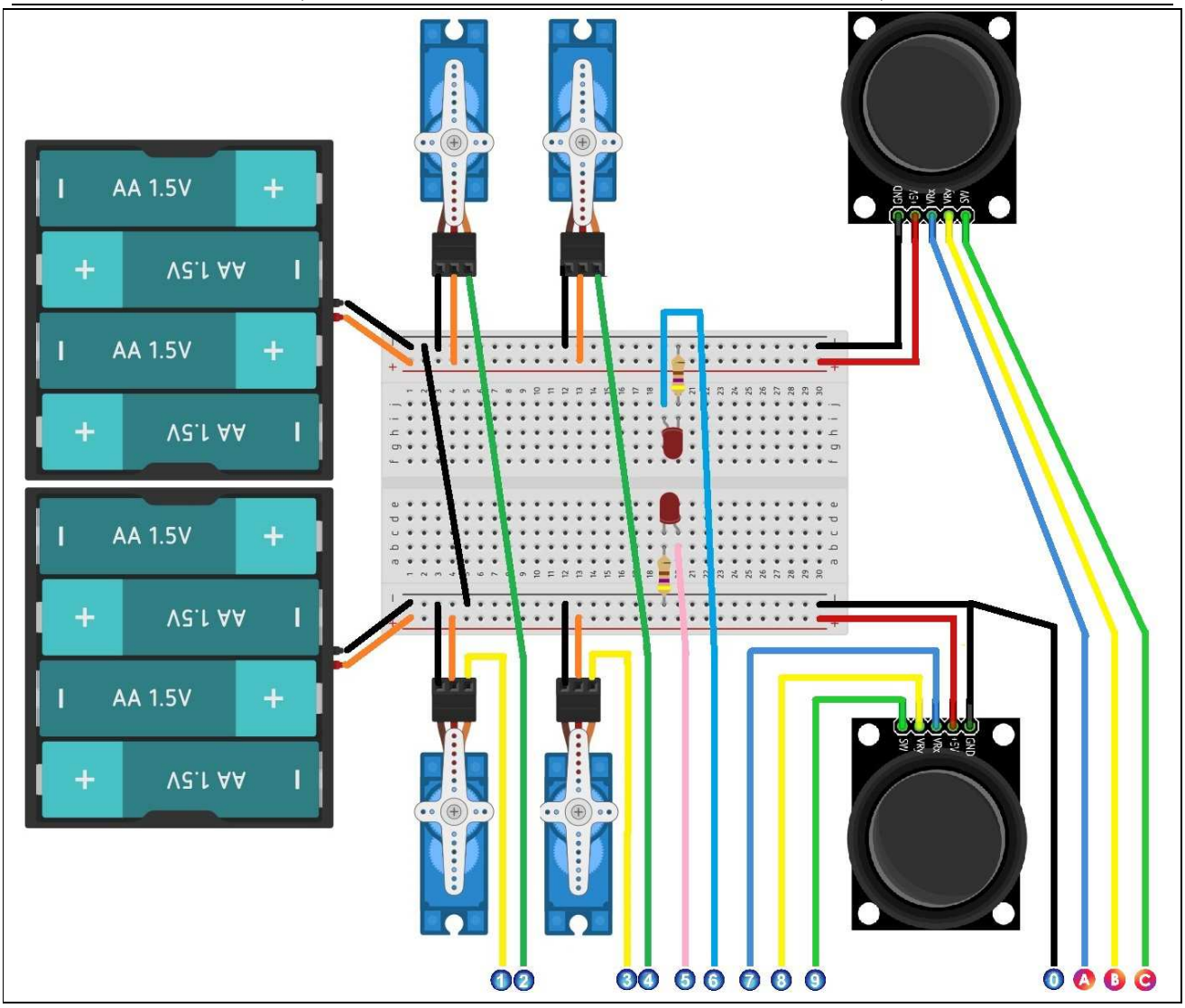
```
77    A_Angulo_X = Calcular_Angulo(Valor_X, A_Angulo_X, A_Salto);
78   //  A_Angulo_X = Calcular_Angulo( Valor_X );
79
80    delay (50);
81
82   /********** Leo Eje Y del Joystick A **********/
83    Valor_Y = analogRead(A_Pin_Eje_Y);
84
85   //  A_Angulo_Y = Calcular_Angulo( Valor_Y, A_Angulo_Y, A_Salto);
86    A_Angulo_Y = Calcular_Angulo( Valor_Y );
87
88   /********** Leo Botón Joystick A ***********/
89     Valor_Boton = digitalRead(A_pinJoyBoton);
90
91   /*** Prendo/Apago Led Asociado al Botón del Joystick A ***/
92     if(Valor_Boton == LOW) digitalWrite(A_PinLED, HIGH);
93     else digitalWrite(A_PinLED, LOW);
94
95   /********** Leo Eje X del Joystick B **********/
96    Valor_X = analogRead(B_Pin_Eje_X);
97
98    B_Angulo_X = Calcular_Angulo(Valor_X, B_Angulo_X, B_Salto);
99   //  B_Angulo_X = Calcular_Angulo( Valor_X );
100
101   delay (50);
102
103  /********** Leo Eje Y del Joystick B **********/
104   Valor_Y = analogRead( B_Pin_Eje_Y );
105
106  //  B_Angulo_Y = Calcular_Angulo( Valor_Y, B_Angulo_Y, B_Salto);
107   B_Angulo_Y = Calcular_Angulo( Valor_Y );
108
109  /********** Leo Botón Joystick B ***********/
110    Valor_Boton = digitalRead( B_pinJoyBoton );
111
112  /*** Prendo/Apago Led Asociado al Botón del Joystick B ***/
113    if(Valor_Boton == LOW) digitalWrite(B_PinLED, HIGH);
114    else digitalWrite(B_PinLED, LOW);
115
116  /*** Muevo Servos Asociados a los Ejes del Joystick A ***/
117   A_Servo_X.write( A_Angulo_X );
118   A_Servo_Y.write( A_Angulo_Y );
119
120  /*** Muevo Servos Asociados a los Ejes del Joystick B ***/
121   B_Servo_X.write( B_Angulo_X );
122   B_Servo_Y.write( B_Angulo_Y );
123
124   delay (30);
125  }
```

Una Funciones está sobrecargada, cuando en el mismo programa hay dos o mas funciones que se llaman igual, pero hacen diferentes cosas y sus parámetros son diferentes en tipo o cantidad. En este Ejercicio, encontraremos que las funciones "**Calcular_Angulo( )**" están sobrecargadas, ya que una tiene tres parámetros y la otra solo dos, y en su interior hacen cosas muy diferentes.

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 2 Joystick analógico para Arduino (de 5 pines), 4 Servos Motor MG90 o MG90S, 2 Led, 2 Resistencia de 470Ω, 8 Cables conectores Hembra/Hembra, 2 cable conector Macho/Hembra, 20 Cables conectores Macho/Macho, 2 Porta Pilas de 4 Pilas AA (1,5V), 8 Pilas AA (1,5V), 1 placa Protoboard, 1 Cable USB - 1 Placa Arduino

-o-

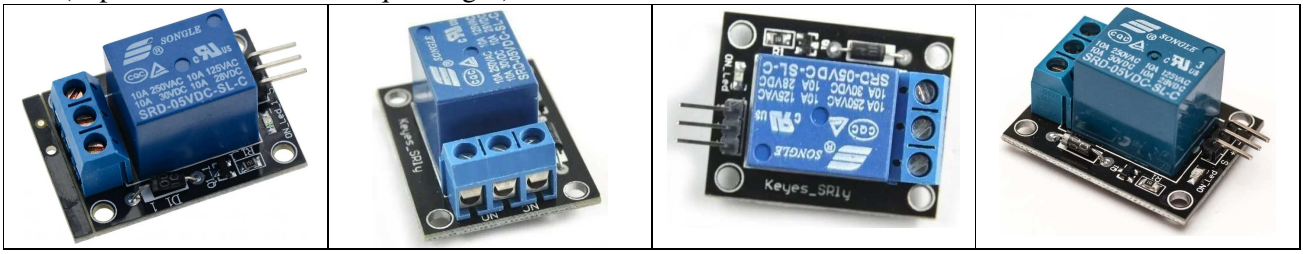


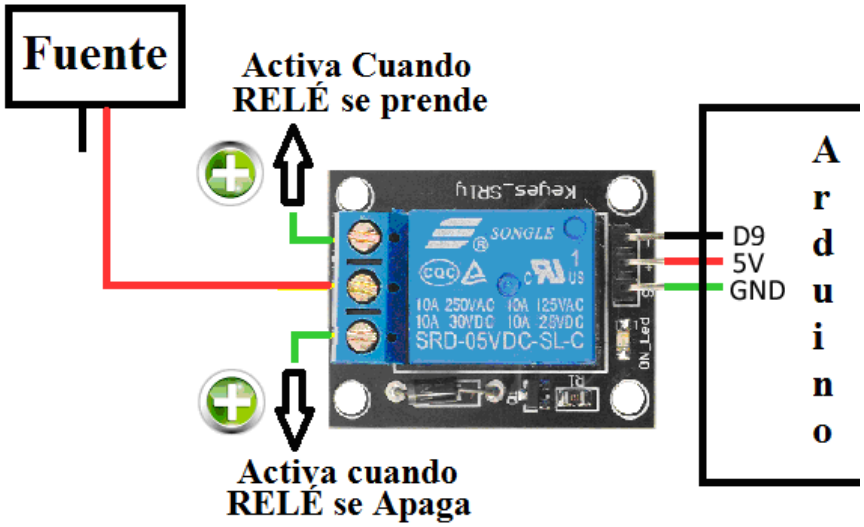
Los cables deberán ser conectados: abajo, de Izquierda a derecha, Cable 1 (amarillo): Conectar al Pin 2. Cable 2 (verde): Conectar al pin 3. Cable 3 (amarillo): Conectar al Pin 4. Cable 4 (verde): Conectar al pin 5. Cable 5 (rosado - Prende Led): Conectar al pin 8. Cable 6 (celeste - Prende Led): Conectar al pin 9. Cable 7 (azul): Conectar al pin Analógico A0. Cable 8 (amarillo): Conectar al pin Analógico A1. Cable 9 (verde): Conectar al Pin Digital que se haya configurado el Botón del Joystick (en este caso Pin 6). Cable 0 (negro) conectar a GND de Arduino (es el GND común de todo el circuito). Cable A (azul): Conectar al pin Analógico A2. Cable B (amarillo): Conectar al pin Analógico A3. Cable C (verde): Conectar al Pin Digital del Botón del Joystick (en este caso Pin 7).

Uso de un Relé

(En Construcción)

Si quieres usar un relé con nuestro Arduino necesitaremos un relé cuya bobina se pueda excitar con 5V o 3V (dependiendo del modelo que tengas).

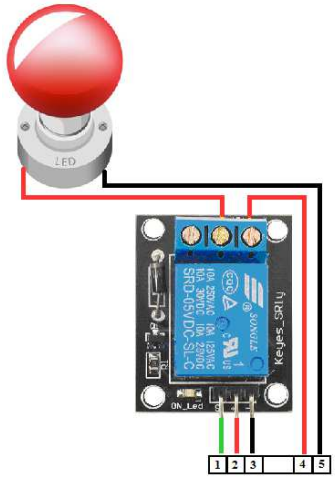


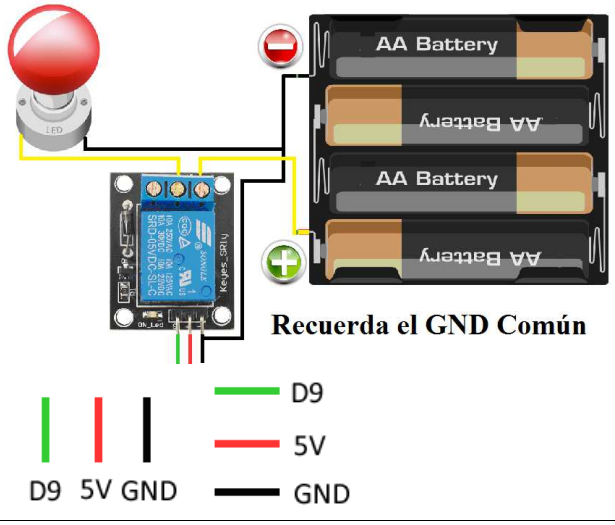


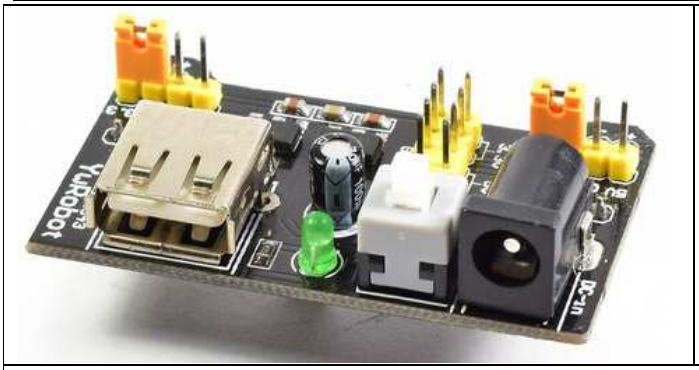
(Programa: "950_Rele_01_Solo_Con_Arduino")

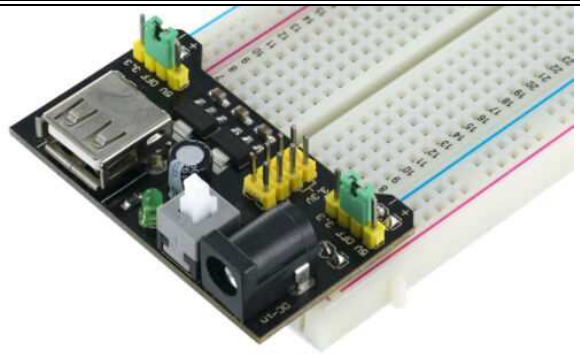
Para probar TODOS los proyectos acá presentados puedes usar este programa.
Es muy Simple.

```
const int pin = 9;

void setup() {
  Serial.begin(9600);    //iniciar puerto serie
  while (!Serial) {
    ; // Esperando se conecte el Puerto serie
  }
  pinMode(pin, OUTPUT);  //definir pin como salida
}

void loop(){
  digitalWrite(pin, HIGH);   // poner el Pin en HIGH
  Serial.println("Prende");
  delay(3000);               // esperar un segundo

  digitalWrite(pin, LOW);    // poner el Pin en LOW
  Serial.println("Apaga");
  delay(3000);               // esperar un segundo
}
```



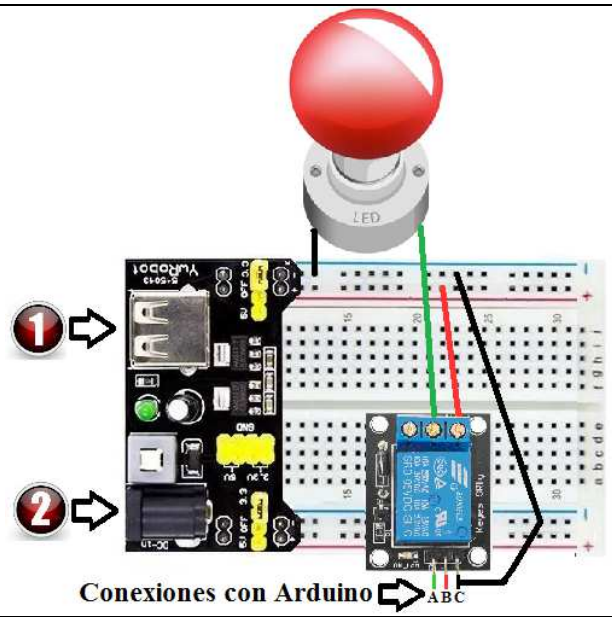


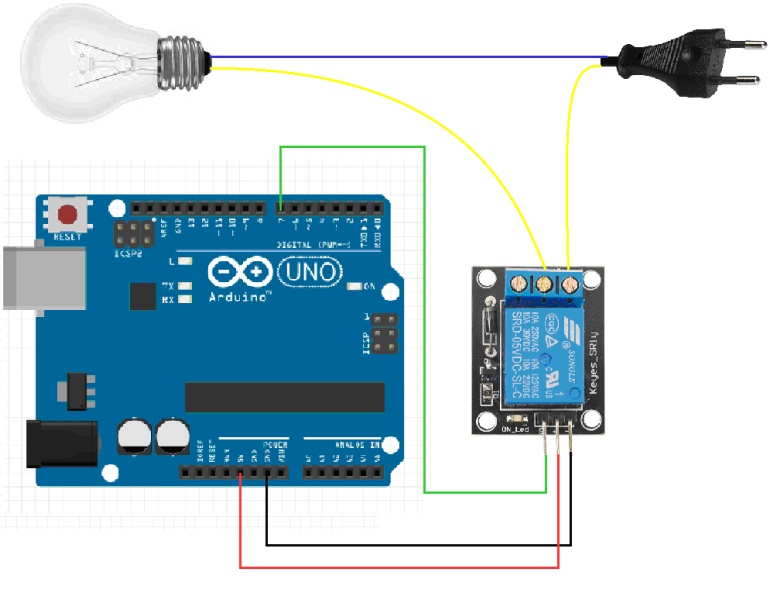


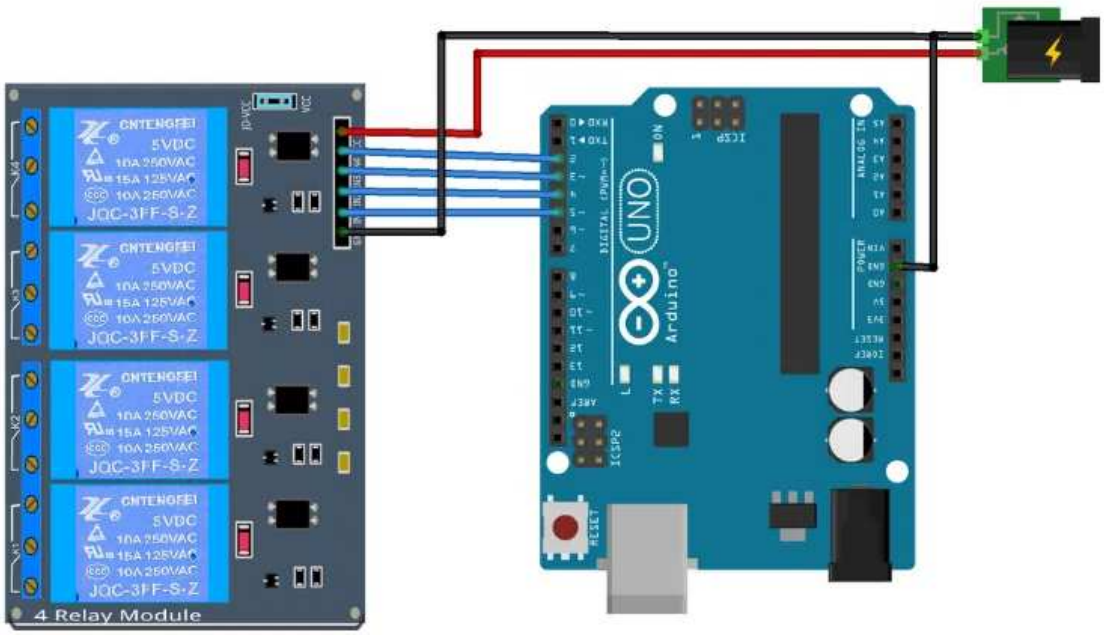


Fuente Para Protoboard 3.3v - 5.0v

(Recomendada, muy práctica y más barata que usar 4 pilas)






4 Relay Module
JQC-3FF-S-Z
5VDC
10A 250VAC
UNO
Arduino

01000101 01101100 00100000 01110110 01100101 01110010 01100100 01100001 01100100
01100101 01110010 01101111 00100000 01110000 01100101 01101100 01101001 01100111
01110010 01101111 00100000 01101110 01101111 00100000 01100101 01110011 00100000
01110001 01110101 01100101 00100000 01101100 01101111 01110011 00100000 01101111
01110010 01100100 01100101 01101110 01100001 01100100 01101111 01110010 01100101
01110011 00100000 01100101 01101101 01110000 01101001 01100101 01100011 01100101
01101110 00100000 01100001 00100000 01110000 01100101 01101110 01110011 01100001
01110010 00100000 01100011 01101111 01101101 01101111 00100000 01101100 01101111
01110011 00100000 01101000 01101111 01101101 01100010 01110010 01100101 01110011
00101100 00100000 01110011 01101001 01101110 01101111 00100000 01110001 01110101
01100101 00100000 01101100 01101111 01110011 00100000 01101000 01101111 01101101
01100010 01110010 01100101 01110011 00100000 01100101 01101101 01110000 01101001
01100101 01100011 01100101 01101110 00100000 01100001 00100000 01110000 01100101
01101110 01110011 01100001 01110010 00100000 01100011 01101111 01101101 01101111
00100000 01101100 01101111 01110011 00100000 01101111 01110010 01100100 01100101
01101110 01100001 01100100 01101111 01110010 01100101 01110011 00101110

Si tienes algunas Correcciones y/o Sugerencias, por favor contáctame.