CAPÍTULO 13:

CONTROL REMOTO

Control remoto

Llegamos al final del trayecto, el último capítulo del curso, donde vamos a trabajar con librerías de programación. Este manejo te va a ser de gran utilidad para trabajar con todo tipo de sensores y actuadores en proyectos propios.

Estás llegando a la cima nuestra montaña, ahora si empezás el disfrute, con la bajada, donde todo es más rápido.

Esperamos que todo este camino haya sido muy satisfactorio, y que puedes seguir transitando este camino de infinitas posibilidades en el mundo de Arduino.

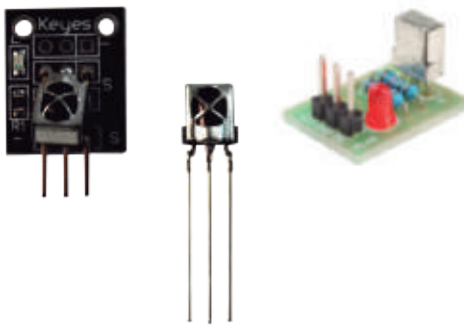
Consejo:

Aprender a usar librerías te va a permitir avanzar rápidamente en nuevos proyectos. Miles de usuarios comparten sus librerías y trabajos. ¡Ahora te toca a vos!

Control infrarrojo

Lo que intentaremos en esta parte es equipar a nuestro robot con un receptor infrarrojo que pueda captar los datos del control, para que ejecute diferentes acciones de acuerdo al botón que presionamos.

En tu kit puedes encontrar un control gris o uno negro, como los que se muestran en la imagen. Cualquiera de los tres cumple con las mismas funciones.



Al igual que con los controles, el receptor infrarrojo viene en tres versiones. Éste es el encargado de captar la señal del control para que el robot la reciba.

¿Cómo funcionan los controles remotos infrarrojos?

Muchos aparatos tecnológicos que usamos en la vida cotidiana están equipados con controles remotos para poder utilizarlos, por ejemplo los televisores, los equipos de audio, los aires acondicionados, etc.

El control es un emisor de luz, por eso en la punta vemos un pequeño diodo LED. La mayoría emite ondas de luz de baja frecuencia que no podemos ver con nuestros ojos. Cuando pulsamos un botón del control, éste envía una serie de ondas lumínicas que son decodificadas por el sensor que se encuentra en el aparato que queremos controlar. El aparato a su vez está programado para realizar alguna acción en particular dependiendo del mensaje que reciba.

Lo veamos en un ejemplo: Cuando presionamos el botón power de un control Sony éste enviará el mensaje por medio de destellos de luz. Nuestro televisor leerá estas ondas e interpretará el mensaje como: "0xA90" (expresión hexadecimal del mensaje). En este caso, nuestro televisor ejecutará las instrucciones de encendido.

Tabla de valores para nuestro control remoto

A continuación encontrarás listados cada uno de los botones de nuestro control infrarrojo y el correspondiente mensaje que envía a nuestro robot. Es importante que prestes mucha atención aquí, ya que según el modelo de tu control remoto (el gris o el negro) estos mensajes varían entre sí.

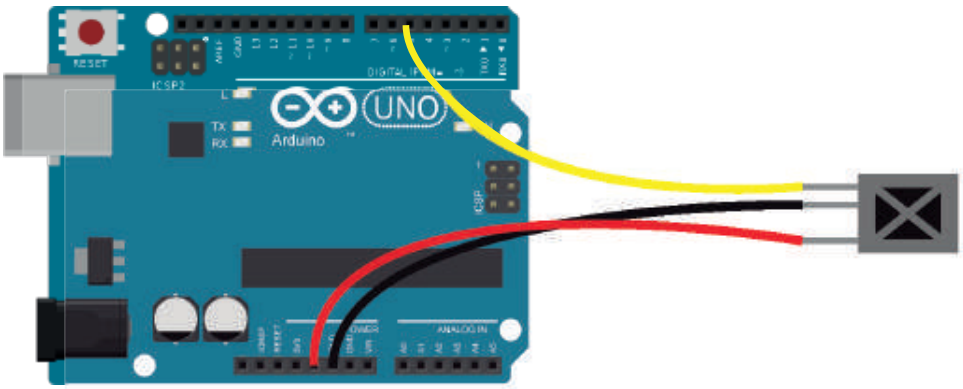
botón	mensaje	botón	mensaje	botón	mensaje
CH-	0xFFA25D	-	0xFFE01F	3	0xFF7A85
CH	0xFF629D	EQ	0xFF906F	4	0xFF10EF
CH+	0xFFE21D	100+	0xFF9867	5	0xFF38C7
Izquierda	0xFF22DD	200+	0xFFB04F	6	0xFF5AA5
Derecha	0xFF02FD	0	0xFF6897	7	0xFF42BD
Play/Pause	0xFFC23D	1	0xFF30CF	8	0xFF4AB5
+	0xFFA857	2	0xFF18E7	9	0xFF52AD

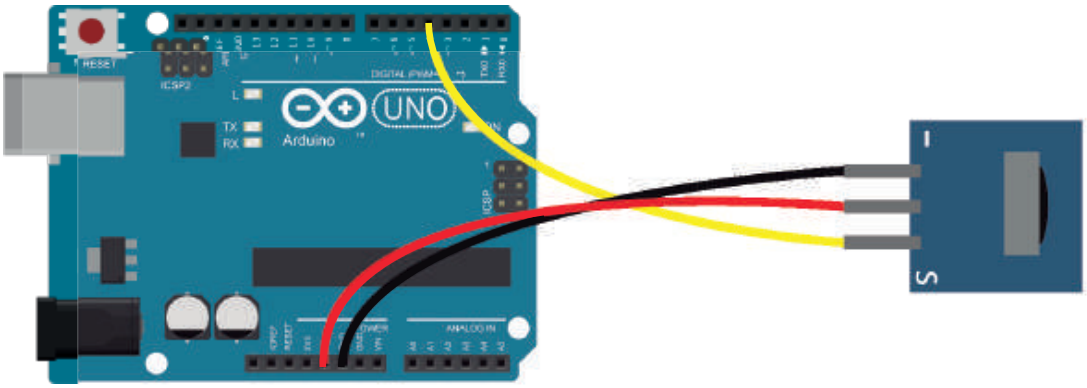
botón	mensaje	botón	mensaje	botón	mensaje
Arriba	0xFF629D	**3**	0xFFB04F	*	0x32C6FDF7
Abajo	0xA3C8EDDB	**4**	0x9716BE3F	**0**	0xFF
Derecha	0xFFC23D	**5**	0x3D9AE3F7	#	0xFF52AD
Izquierda	0xFF22DD	**6**	0x6182021B		
OK	0xD7E84B1B	**7**	0x8C22657B		
1	0xC101E57B	**8**	0xFF38C7		
2	0x97483BFB	**9**	0xFF5AA5		

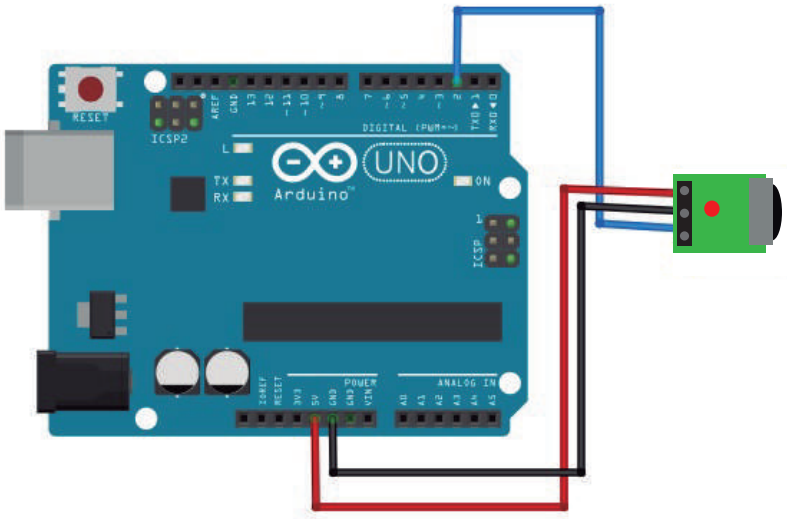
botón	mensaje	botón	mensaje	botón	mensaje
Encendido	0xFD00FF	Vol (-)	0xFD906F	3	0xFD48B7
Vol (+)	0xFD807F	Arriba	0xFD50AF	4	0xFD28D7
Func/Stop	0xFD40BF	0	0xFD30CF	5	0xFDA857
Rew	0xFD20DF	EQ	0xFDB04F	6	0xFD6897
Play/pause	0xFDA05F	ST/Rept	0xFD708F	7	0xFD18E7
Forw	0xFD609F	1	0xFD08F7	8	0xFD9867
Abajo	0xFD10EF	2	0xFD8877	9	0xFD58A7

Esquema de conexiones del receptor IR

Ambos receptores poseen tres terminales, sin embargo difieren en su conexión. A continuación podremos ver cada una de ellas:







Clases y objetos

Los programas que hicimos hasta ahora, fueron escritos usando una **Programación estructurada**. Según esta forma de programar, podremos hacer cualquier programa utilizando un conjunto de instrucciones escritas en orden, desde arriba hacia abajo, y usando dos estructuras básicas para modificar esa secuencia: los condicionales (IF / IF...ELSE) y las iteraciones o bucles (FOR, WHILE, etc).

Al utilizar la librería **IRremote.h** hicimos referencia al uso de clases y objetos, que son conceptos de otra forma de programación: la **Programación Orientada a Objetos (POO)**.
Esta es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que los otros tipos de programación.

La POO utiliza objetos interactuando entre sí para escribir el programa y por ende tenemos que aprender a pensar de una manera distinta. En este curso no es necesario profundizar en la programación orientada a objetos, pero en los siguientes párrafos vamos a explicar qué son las clases y los objetos para que quede un poco más claro por si quisieras ahondar en esta lógica de programación en un futuro.

¿Cómo se piensa en objetos?

Pensar en términos de objetos es muy parecido a cómo lo haríamos en la vida real. Por ejemplo, vamos a pensar en un celular para tratar de modelarlo en un esquema de POO.

Una clase es la estructura de un objeto, es decir, la definición de todos los elementos de los que está hecho un objeto. Esta estructura se compone de dos partes:

· **Atributos: son las características propias del objeto.** En el caso de un teléfono celular podrían ser la marca, el modelo, su sistema operativo, etc.

Los atributos se definen por medio de variables que adquieren valores al crearse un objeto en particular.

· **Métodos: son las cosas que puede hacer el objeto.** Regresando al ejemplo del celular, éste puede sacar fotos, llamar a otros teléfonos, enviar mensajes, etc.

Estos métodos se programan mediante funciones como las que ya hemos creado en nuestros programas.

Entonces, cuando queremos usar una clase, debemos crear un objeto y darle valores específicos. Por ende, un objeto es una instancia de una clase, es decir, un celular en particular. Esto significa que puede haber varios objetos de una misma clase.

¿Cómo crear un objeto?

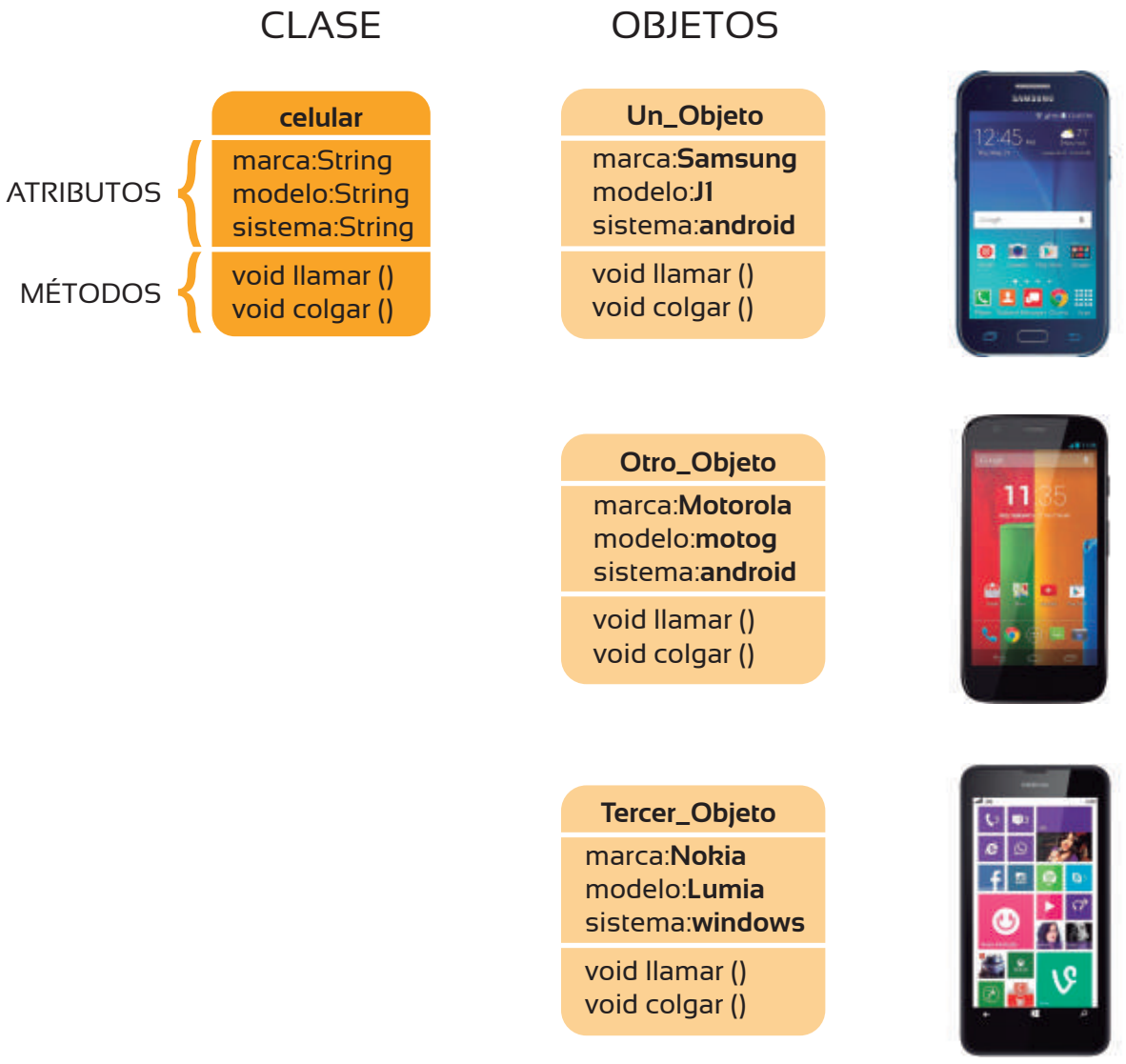
Para crear un objeto se hace de manera muy similar a la definición de variables, pero en este caso, en vez de usar los tipos de datos int, float, etc., usaremos el nombre de la clase, y en lugar del nombre de la variable, escribiríamos el nombre de nuestro objeto. La diferencia es que debemos agregar como parámetro los valores de los atributos de nuestro objeto en particular.

Veamos rápidamente cómo quedaría la programación de un objeto retomando nuestro ejemplo del teléfono celular.

```
int ledTest = 13; // acá creamos una variable
celular  miCelular (samsung, J1, android); // acá creamos un objeto de la clase celular
```

Ahora supongamos que queremos escribir una línea de código para llamar a alguien usando nuestro objeto. Para esto, vamos a invocar al objeto por su nombre, y su método (función) **"Llamar".** Deberíamos escribir una línea como la siguiente:

```
miCelular.llamar (351455877)   /* a modo ilustrativo, le pasamos el numero de
                               teléfono con el que deseamos comunicarnos como
                               parámetro a la función. */
```



Manejo de librerías

Para nuestro robot utilizaremos un control remoto y un receptor de infrarrojo. Las conexiones ya están solucionadas, sólo hay que asegurarnos que el sensor esté conectado a la placa de Arduino. Si nos fijamos bien, al lado de los pines donde está conectado el receptor dice un pequeño número, ese es el pin del arduino al que está conectado y lo vamos a utilizar en la programación.

Lo primero que debemos hacer en nuestra programación es importar la librería que vamos a utilizar. Para esto hacemos el siguiente include:

```
#include <IRremote.h>
```

Para tener la librería en tu computadora utilizá este **link de DESCARGA** y seguí los pasos de instalación del video tutorial.

Recordemos que esto incluye la librería dentro de nuestro programa para que podamos utilizarla. Ahora vamos a crear nuestro objeto receptor por medio de la clase "IRrecv" de la librería que acabamos de incluir.

```
IRrecv receptor (3);
```

¡IMPORTANTE! Entre paréntesis debes escribir el número del puerto al que está conectado el receptor.

Luego debemos declarar un objeto que va a interpretar el mensaje que recibamos, es decir, que va a decodificar la señal para que podamos usarla en la programación de nuestro robot.

```
decode_results resultado;
```

Ahora vamos a habilitar la recepción de mensajes. Dentro de la función setup() debemos escribir la siguiente línea:

```
receptor.enableIRIn();
```

¿Cómo utilizar estos dos objetos?

El receptor va a recibir la señal del control remoto y dejarla guardada en el objeto resultado por medio de la siguiente función:

```
receptor.decode(&resultado)
```

Esta función devuelve un true si recibe alguna señal o false si no detecta nada, por esto normalmente se usa como condición de una bifurcación. Es decir:

```
if (receptor.decode(&resultado)) {
}
```

Luego, si recibió un mensaje, podemos usar el resultado de la decodificación por medio de la línea:

```
resultado.value;
```

Por último, debemos preparar el receptor para recibir un nuevo mensaje por medio de la función resume().

```
receptor.resume();
```

/terminales-files/common/LibrerialRremote.zip

Veamos un ejemplo

En la siguiente programación vamos a detectar lo que capta el sensor infrarrojo y vamos a mostrarlo por medio del puerto serie.

```
#include <IRremote.h>

IRrecv receptor(3);
decode_results resultado;

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

void loop() {
        if (receptor.decode(&resultado)) {
        Serial.println(resultado.value,HEX);
        receptor.resume();
        }
delay(50);
}
```

switch... case

Al igual que las instrucciones IF, **switch... case** controla el flujo del programa permitiéndonos especificar diferentes instrucciones que serán ejecutadas en base a una condición. **Switch** compara el valor de una variable determinada con los valores especificados en cada una de las instrucciones **case**. Cuando se encuentra una sentencia **case** cuyo valor coincide con el de la variable, el código dentro de ese **case** es ejecutado.

Su sintaxis correcta es:

```
switch (variable) {
        case 1:         // instrucciones para el estado "1"
                break;
        case 2:         // instrucciones para el estado "2"
                break;
        default:        // instrucciones para el estado "default"
                break;
}
```

La palabra reservada **break** interrumpe la sentencia switch, y se usa normalmente al final de cada declaración case. Sin esta declaración de interrupción break, switch continuará la ejecución de las siguientes expresiones hasta que se alcanza una interrupción o hasta el final de la sentencia switch.

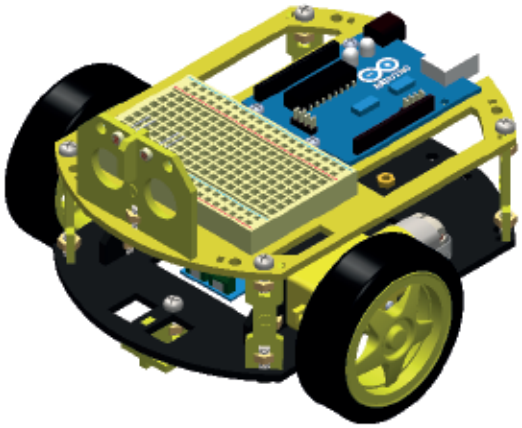
Por último, tenemos el caso **default**. Si switch no encuentra coincidencias entre la variable y los distintos cases, ejecutará las instrucciones escritas en la opción default.

A continuación veremos un ejemplo para que comprendas mejor cómo funciona esta estructura en particular:

```
switch (var) {          // la variable "var" contiene el valor a comparar con los case

    case 1:             // se ejecuta cuando "var"  es igual a 1
        break;

    case 2:             // se ejecuta cuando "var"  es igual a 2
        break;

    case 3:             // se ejecuta cuando "var"  es igual a 3
        break;

    case 4:             // se ejecuta cuando "var"  es igual a 4
        break;

    case 5:             // se ejecuta cuando "var" es igual a 5
        break;

    case 6:             // se ejecuta cuando "var"  es igual a 6
        break;

    ....

    case 16:            //podemos poner tantos case como sea necesario
        break;

    default:            /* si nada coincide, switch toma el caso default o predeterminado. Sin
                           embargo, declarar default es opcional. Si nuestra programación está
                           bien hecha, nuestro robot nunca debería seguir el camino default. */
        break;
}
```

Ensamble 1

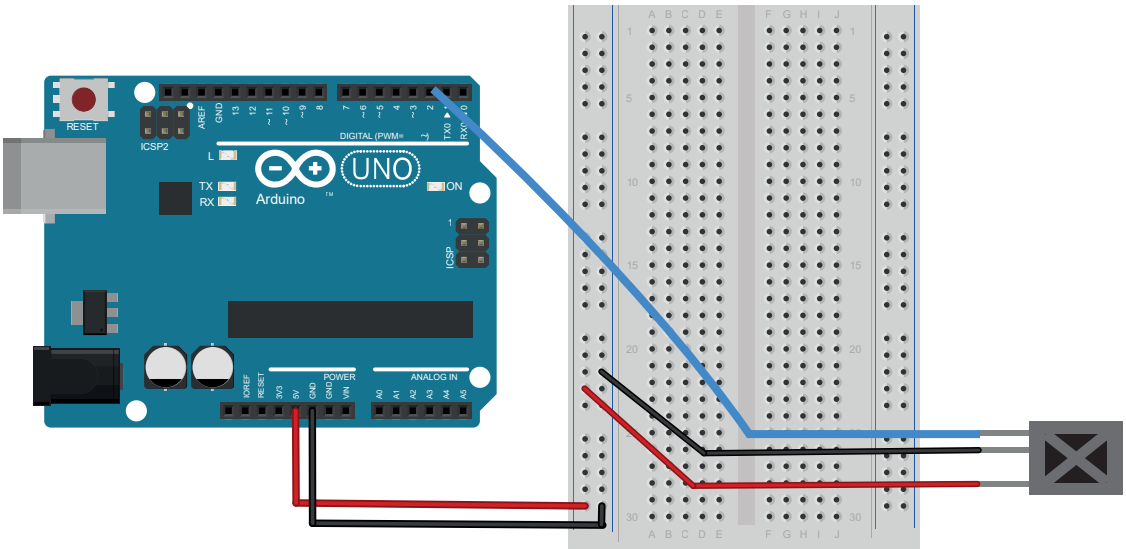
A continuación veremos la estructura y componentes para realizar un experimento puerto serie y leer los valores que recibe el sensor IR. Podemos usar la misma estructura del proyecto pasado, ya que trabajaremos principalmente sobre la protoboard y el Arduino. Si, es recomendable que retires todas las conexiones del capítulo anterior. También podés usar el ensamble de Hola mundo, para tener más ordenadas las conexiones.

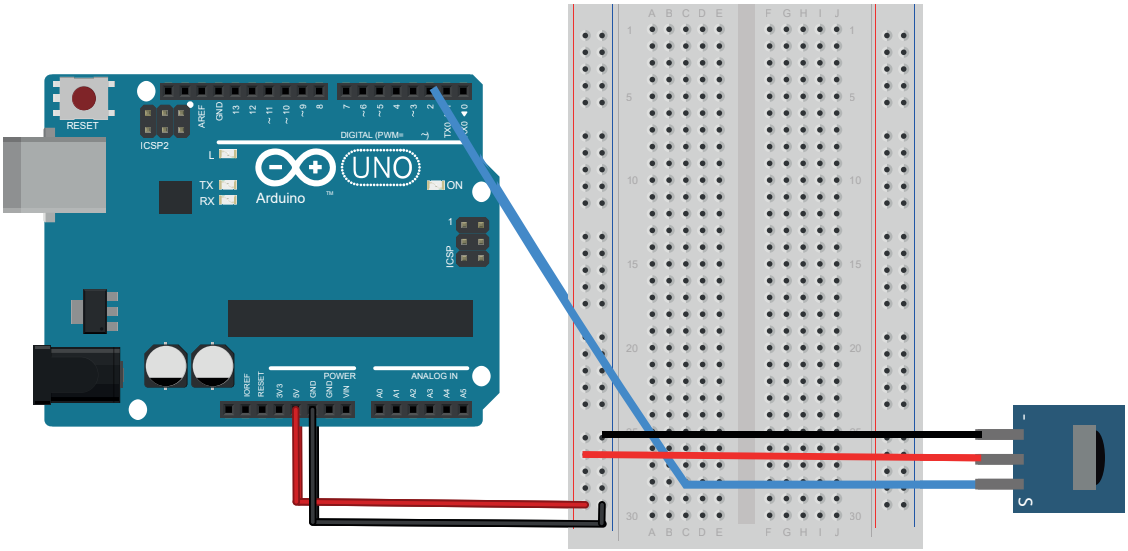


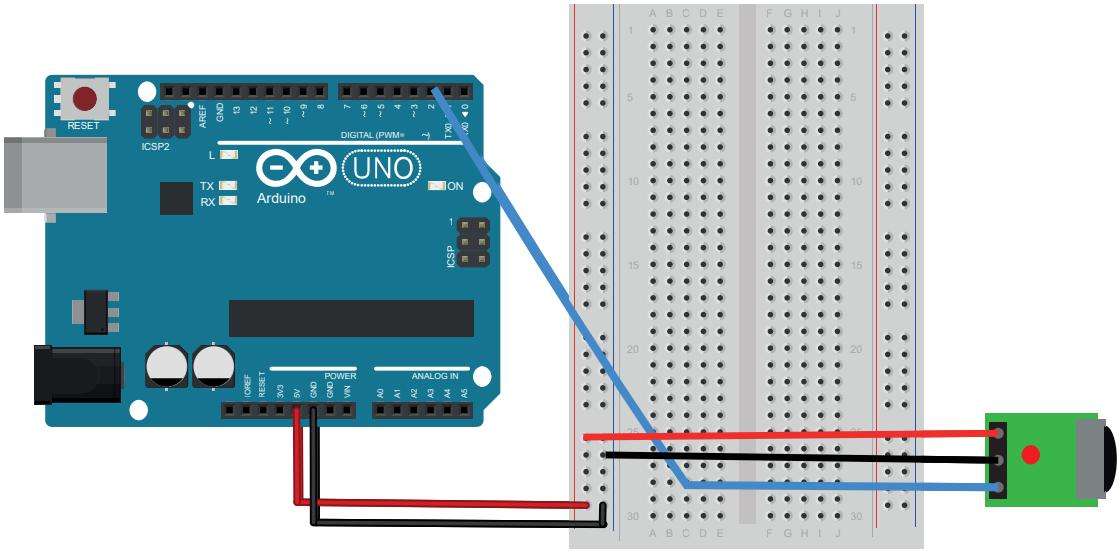
Materiales:
-Estructura de seguidor de luz
-Receptor IR
-Control remoto

Circuito electrónico

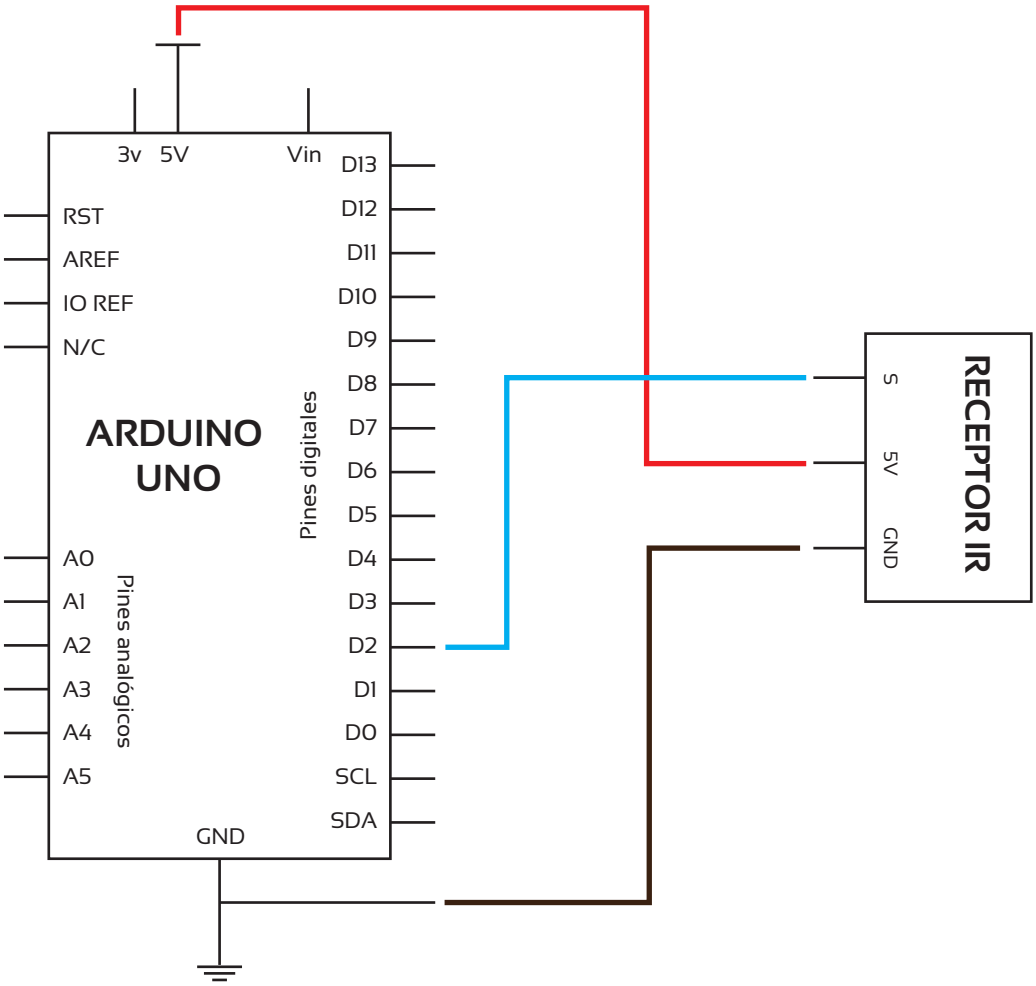
Observamos las conexiones de nuestro circuito electrónico, vemos 3 conexiones similar, una para cada receptor (realizar la correspondiente)




RESET
ICSP2
DIGITAL (PWM~)
UNO
Arduino
ON
TX
RX
L
ICSP
POWER
ANALOG IN


RESET
ICSP2
DIGITAL (PWM~)
UNO
Arduino
ON
TX
RX
L
ICSP
POWER
ANALOG IN

Circuito electrónico 1 en símbolos



Código de programación 1

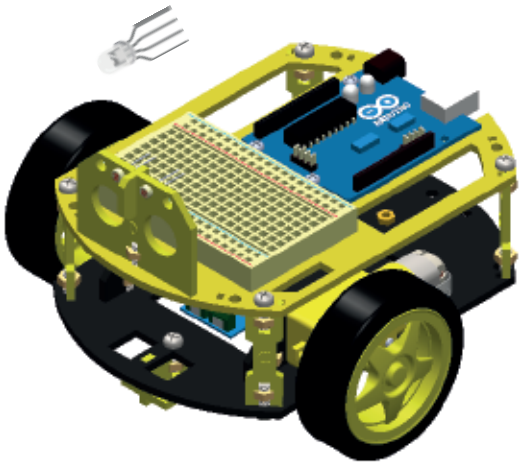
Desarrollaremos el código de programación leer los datos recibidos a través de un control remoto y los vamos a colocar en una tabla para su posterior uso (podés usar un blog de notas o word, para ir anotando los códigos que vayas registrando.

```
#include <IRremote.h>                          //Manejo de librería
#include <IRremoteInt.h>                       //descarga:
                                               https://github.com/z3t0/Arduino-IRremote/zipball/master

int ledTest = 13;


int ReceptorIR = 2;                            //Seteo del receptor infrarojo
IRrecv receptorIr(ReceptorIR);                 // Creamos un objeto de la clase IRrecv llamado
decode_results codigoLeido;                    "receptorIr".

void setup() {

  Serial.begin(9600);
  receptorIr.enableIRIn();

  pinMode(ledTest, OUTPUT);                    //Seteo Serial para testeo y el receptor infrarojo


                                               //Configuramos los pines que necesitamos
                                               como salidas utilizando las variables declaradas
  digitalWrite(ledTest, LOW);                  anteriormente.


                                               //Una vez realizado esto, nos aseguramos que
digitalWrite(ledTest, HIGH);                   están apagadas las salidas
  delay(500);
  digitalWrite(ledTest, LOW);                  //Saludo inicial
  delay(500);
  digitalWrite(ledTest, HIGH);
  delay(500);
  digitalWrite(ledTest, LOW);
  delay(500);
  digitalWrite(ledTest, HIGH);
  delay(500);
  digitalWrite(ledTest, LOW);
  delay(500);
}

void loop() {
  if (receptorIr.decode(&codigoLeido))
  {
    Serial.print("0x");                       //Texto para lectura en monitor serie.
    Serial.println(codigoLeido.value, HEX);
    delay(50);
    receptorIr.resume();
  }
}
```

Nota: recordá ir registranto o armando un listado del código de cada uno de los botones de tu control: EJ: Botón 9 = 0xFF52AD"

Ensamble 2

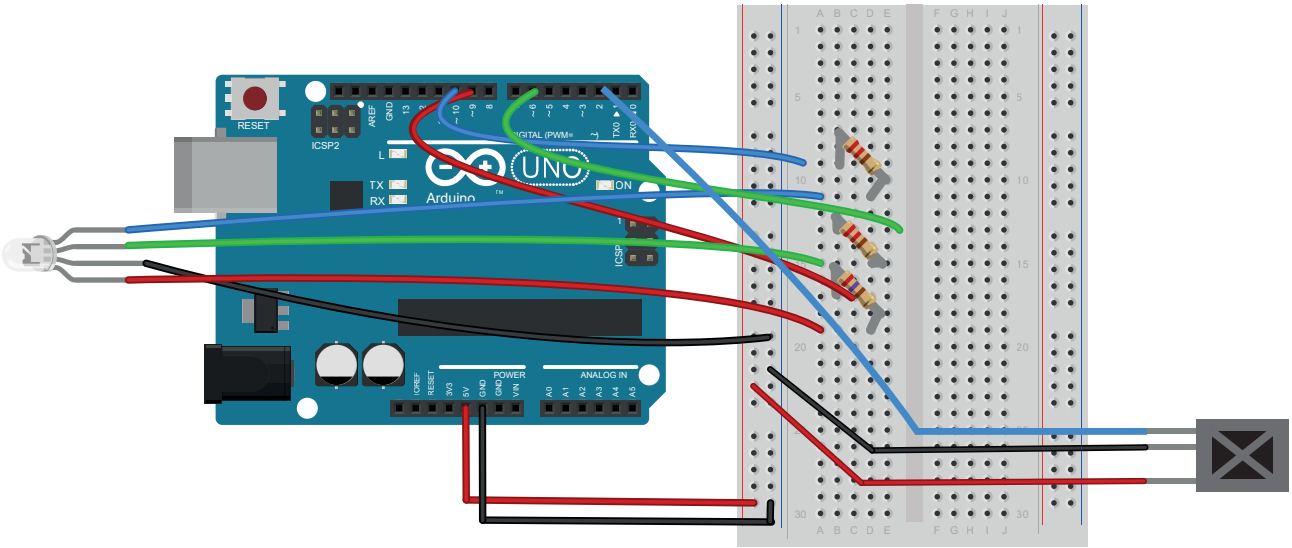
Finalmente veremos la estructura y componentes para realizar el ultimo proyecto. Haremos que el led RGB cambie de color con nuestro control remoto y el sensor IR.

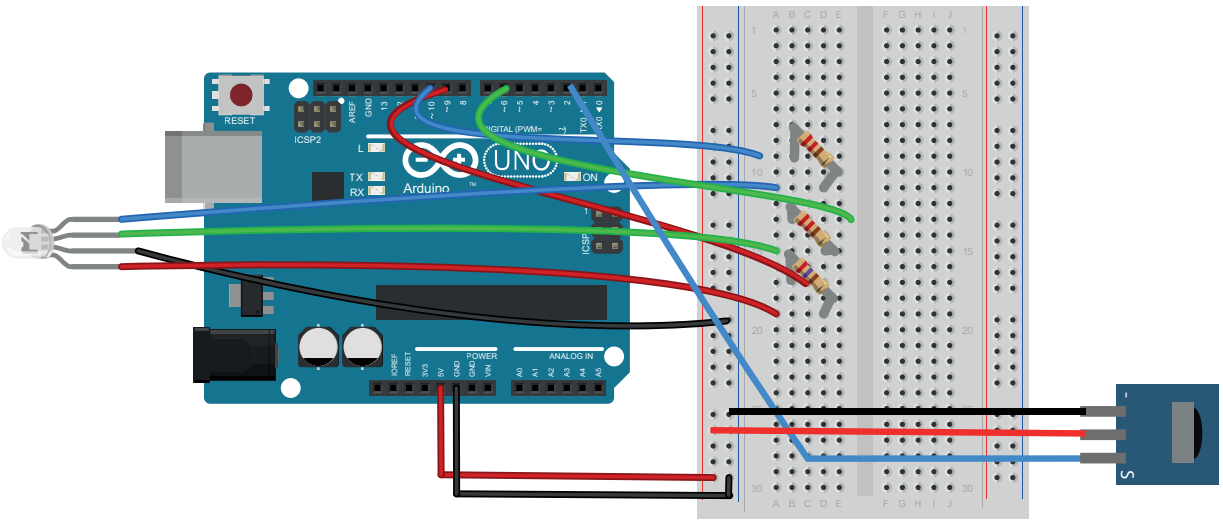


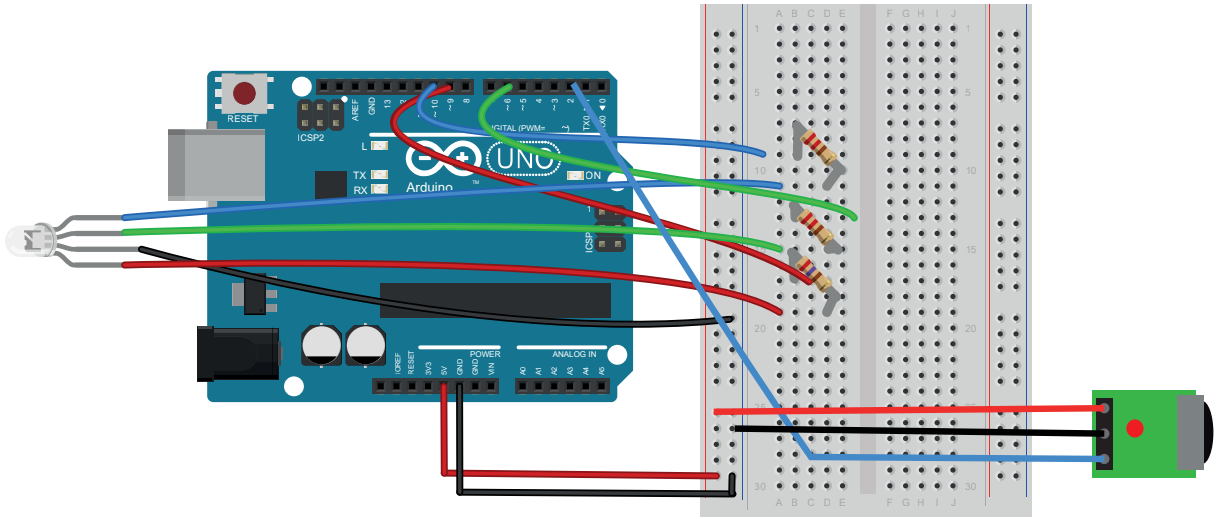
Materiales:
-Estructura de seguidor de luz
-Receptor IR
-Control remoto
-Led RGB

Circuito electrónico 2

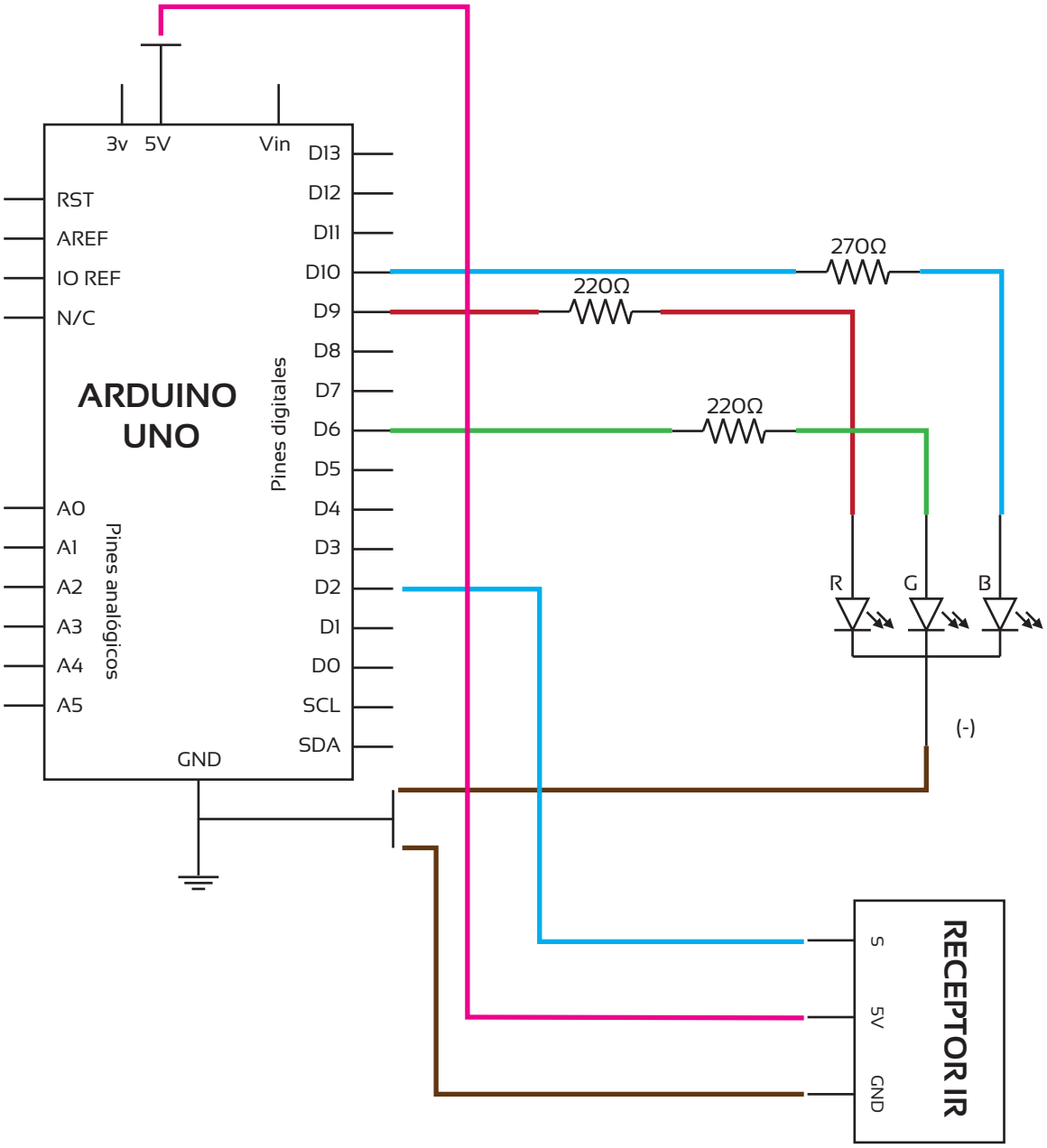
Observamos las conexiones de nuestro circuito electrónico, nuevamente mostraremos las 3 conexiones posibles. Recordar tambíen que puede variar según el tipo de led RGB que tengas.




RESET
ICSP2
UNO
Arduino
TX
RX
ON
POWER
ANALOG IN


RESET
ICSP2
UNO
Arduino
TX
RX
ON
POWER
ANALOG IN

Circuito electrónico 2 en símbolos



Código de programación 2

En este segundo código vamos a encender el led RGB con los distintos botones del control remoto, usando los valores tomados anteriormente.

```
#include <IRremote.h>                       //Manejo de librería
#include <IRremoteInt.h>

int ledTest = 13;

int ReceptorIR = 2;                         //Seteo del receptor infrarojo.
IRrecv receptorIr(ReceptorIR);              // Creamos un objeto de la clase IRrecv llamado
                                            "receptorIr".


decode_results codigoLeido;                 //declaramos 2 variables de tipo decode_results
decode_results codigoLeidoNuevo;            llamadas "codigoLeido" y "codigoLeidoNuevo"

int ledRojo = 9;                            //Seteo del led RGB pines y variables.
int ledVerde = 10;
int ledAzul = 6;

int potenciaRojo = 0;                       //Los valores de potencia serán 0 = apagado; 127
int potenciaVerde = 0;                      que es un 50%; y 255 que sería un 100%
int potenciaAzul = 0;

void setup() {

  Serial.begin(9600);
  receptorIr.enableIRIn();                  // Habilitamos al receptor para que comience a
                                            intentar detectar señales infrarrojas.

  pinMode(ledRojo, OUTPUT);
  pinMode(ledVerde, OUTPUT);
  pinMode(ledAzul, OUTPUT);

  analogWrite(ledRojo, 0);
  analogWrite(ledVerde, 0);
  analogWrite(ledAzul, 0);

  pinMode(ledTest, OUTPUT);
  digitalWrite(ledTest, LOW);

  digitalWrite(ledTest, HIGH);              //Saludo inicial
  delay(500);
  digitalWrite(ledTest, LOW);
  delay(500);
  digitalWrite(ledTest, HIGH);
  delay(500);
  digitalWrite(ledTest, LOW);
  delay(500);
  digitalWrite(ledTest, HIGH);
  delay(500);
  digitalWrite(ledTest, LOW);
  delay(500);
}
```

```
51  void loop() {
52   if (receptorIr.decode(&codigoLeidoNuevo))                    //Si recibimos algún dato, lo guardamos
53   {                                                            en codigoLeidoNuevo.
54
55    receptorIr.resume();                                        // Le indica al receptor que ya guarda-
56   }                                                            mos el nuevo código leído y ya puede
57                                                                esperar al siguiente
58
59   if (codigoLeidoNuevo.value != codigoLeido.value &&           // Sólo vamos a realizar una acción si el
     codigoLeidoNuevo.value != 0xFFFFFFFF)                        valor leído ha cambiado y si el pulsador
60   {                                                            no se mantiene pulsado (0xFFFFFFFF)
61
62    digitalWrite(ledTest, HIGH);                                // Hacemos parpadear nuestro ledTest
63    delay(50);                                                  para que sepamos que recibimos un
64    digitalWrite(ledTest, LOW);                                 código desde el control.
65
66    Serial.println(codigoLeido.value, HEX);                     // Mostramos el valor por el monitor
67                                                                serie.
68
69    codigoLeido = codigoLeidoNuevo;
70                                                                // switch compara el valor de una varia-
71                                                                ble con los valores escritos en cada
72                                                                instrucción case. Cuando encuentra un
73                                                                "caso" (Case) cuyo valor coincide con el
74                                                                de la variable, ejecuta el código especifi-
75                                                                cado en él.
76
77   switch (codigoLeido.value) {
78     case 0xFF30CF:                                             //botón 1
79       analogWrite(ledRojo, 0);                                 //Apagamos el led Rojo
80       break;
81     case 0xFF18E7:                                             //botón 2
82       analogWrite(ledRojo, 127);                               //led Rojo al 50%
83       break;
84     case 0xFF7A85:                                             //botón 3
85       analogWrite(ledRojo, 255);                               //led Rojo al 100%
86       break;
87     case 0xFF10EF:                                             //botón 4
88       analogWrite(ledVerde, 0);                                //Apagamos el led Verde
89       break;
90     case 0xFF38C7:                                             //botón 5
91       analogWrite(ledVerde, 127);                              //led Verde al 50%
92       break;
93     case 0xFF5AA5:                                             //botón 6
94       analogWrite(ledVerde, 255);                              //led Verde al 100%
95       break;
96     case 0xFF42BD:                                             //botón 7
97       analogWrite(ledAzul, 0);                                 //Apagamos el led Azul
98       break;
99     case 0xFF4AB5:                                             //botón 8
100      analogWrite(ledAzul, 127);                               //led Azul al 50%
101      break;
102    case 0xFF52AD:                                             //botón 9
103      analogWrite(ledAzul, 255);                               //led Azul al 100%
104      break;
```

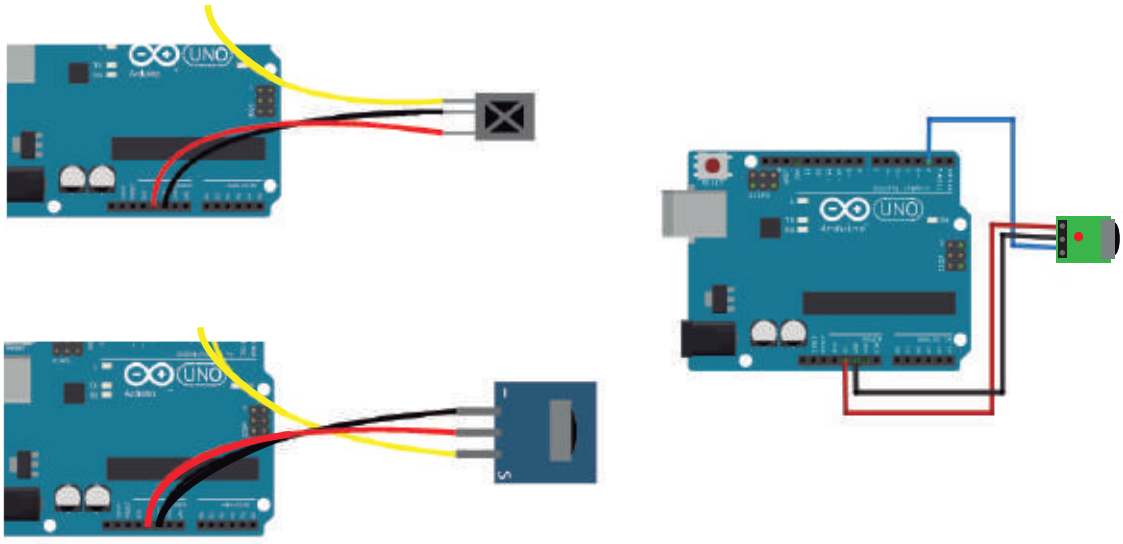
```
105       case 0xFFA25D:                    //botón CH-
106         analogWrite(ledRojo, 0);        //Apagamos el led Rojo
107         analogWrite(ledVerde, 0);       //Apagamos el led Verde
108         analogWrite(ledAzul, 0);        //Apagamos el led Azul
109         break;
110       case 0xFFE21D:                    //botón CH+   LUZ BLANCA
111         analogWrite(ledRojo, 255);      //Rojo 100%
112         analogWrite(ledVerde, 255);     //Verde 100%
113         analogWrite(ledAzul, 255);      //Azul 100%
114         break;
115
116       default:                          //Si el código recibido no es ninguno de
117         digitalWrite(ledTest, HIGH);    los que buscamos hacemos parpadear
118         delay(200);                     el led de prueba
119         digitalWrite(ledTest, LOW);
120         delay(200);
121         digitalWrite(ledTest, HIGH);
122         delay(200);
123         digitalWrite(ledTest, LOW);
124         break;
125     }
126   }
127 }
128
```

MATERIAL COMPLEMENTARIO

Testing de componentes

Receptor infrarrojo

Para probarlo hay que armar un circuito sencillo; escoge el ejemplo A, B o C según el modelo de receptor que tengas en tu kit.



Una vez armado el ensamble conectar el arduino a la PC, copiar el programa de testing en el IDE (es el mismo para ambos componentes) y ejecutarlo. Abrir el monitor serie, y disparar hacia el sensor con cualquier control remoto que tengamos a mano. Verificar que nos muestre los códigos recibidos en el monitor serie.

Código para probar el receptor infrarrojo

```
#include <IRremote.h>
#include <IRremoteInt.h>

int ledTest = 13;
int ReceptorIR = 4;
IRrecv receptorIr(ReceptorIR);
decode_results codigoLeido;

void setup() {
  Serial.begin(9600);
  receptorIr.enableIRIn();
  pinMode(ledTest, OUTPUT);
  digitalWrite(ledTest, LOW);
}

void loop() {
  if (receptorIr.decode(&codigoLeido))
  {
    Serial.print("0x");
    Serial.println(codigoLeido.value, HEX);
    delay(50);
    receptorIr.resume();
  }
}
```

ACTIVIDADES Y EJERCITACIÓN

Cap. 13: Actividades complementarias

En base a los contenidos vistos en el **"Capítulo 13: Control remoto"** realiza las siguientes actividades complementarias.

Actividad 1

Desafío práctico: Circuito básico

Modifica el código del **"Circuito básico"** para que en lugar de mostrar el código del botón apretado, éste muestre el nombre de la tecla. Por ejemplo: **"Botón 1"**, **"Vol +"**, etc.

Recomendaciones:
- Usa las estructuras **IF / ELSE IF**.
- Puedes usar como referencia la tabla de valores incluída en el material **"Control Remoto IR"**.

Actividad 2

Desafío práctico: Circuito básico + RGB

Modifica el código para utilizar las teclas sobrantes del control remoto y así lograr a través de ellas que se generen otros colores o funciones del led. Por ejemplo: conseguir que el led parpadee, que se ponga amarillo, etc.

Actividad 3

Desafío práctico: Circuito básico + RGB

Modifica el código para que al apretar los botones del control, la intensidad de los colores aumente o disminuya de manera gradual. Por ejemplo, si presionamos el número 1 que disminuya la intensidad del azul hasta apagarse, y si presionamos el 3, que aumente la intensidad hasta llegar al límite de 255. Igual con los otros colores.

Cap. 13: Actividades complementarias

En base a los contenidos vistos en el **"Capítulo 13: Control remoto"** realiza las siguientes actividades complementarias.

Actividad 1

Desafío práctico: Circuito básico

```
#include <IRremote.h>
#include <IRremoteInt.h>

int ReceptorIR = 2;
IRrecv receptorIr(ReceptorIR);
decode_results codigoLeido;

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

void loop() {
  if (receptorIr.decode(&codigoLeido)) {
    if (codigoLeido.value == 0xFD08F7) Serial.println("Botón 1");
    else if (codigoLeido.value == 0xFD8877) Serial.println("Botón 2");
    else if (codigoLeido.value == 0xFD48B7) Serial.println("Botón 3");
    else if (codigoLeido.value == 0xFD28D7) Serial.println("Botón 4");
    else if (codigoLeido.value == 0xFDA857) Serial.println("Botón 5");
    else if (codigoLeido.value == 0xFD6897) Serial.println("Botón 6");
    else if (codigoLeido.value == 0xFD18E7) Serial.println("Botón 7");
    else if (codigoLeido.value == 0xFD9867) Serial.println("Botón 8");
    else if (codigoLeido.value == 0xFD58A7) Serial.println("Botón 9");
    delay(50);
    receptorIr.resume();
  }
}
```

Actividad 2

Desafío práctico: Circuito básico + RGB

```
#include <IRremote.h>
#include <IRremoteInt.h>

int ReceptorIR = 2;
IRrecv receptorIr(ReceptorIR);
decode_results codigoLeido;
decode_results codigoLeidoNuevo;

int ledRojo = 9;
int ledVerde = 6;
int ledAzul = 10;

int potenciaRojo = 0;
int potenciaVerde = 0;
int potenciaAzul = 0;
```

```
int color= 0; // si es 1 es rojo, si es 2 es verde y si es 3 es azul.

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

  pinMode(ledRojo, OUTPUT);
  pinMode(ledVerde, OUTPUT);
  pinMode(ledAzul, OUTPUT);

  analogWrite(ledRojo, 0);
  analogWrite(ledVerde, 0);
  analogWrite(ledAzul, 0);
}

void loop() {
  if (receptorIr.decode(&codigoLeidoNuevo)) {
    receptorIr.resume();
  }

  if (codigoLeidoNuevo.value != codigoLeido.value && codigoLeidoNuevo.value !=
0xFFFFFFFF)  {
  Serial.println(codigoLeido.value, HEX);
   codigoLeido = codigoLeidoNuevo;
      switch (codigoLeido.value) {
      case 0xFD08F7: //número 1
        color =1;
        analogWrite(ledRojo, 0);
        break;
      case 0xFD8877: //número 2
        color =1;
        analogWrite(ledRojo, 127);
        break;
      case 0xFD48B7: //número 3
        color =1;
        analogWrite(ledRojo, 255);
        break;
      case 0xFD28D7: //número 4
        color =2;
        analogWrite(ledVerde, 0);
        break;
      case 0xFDA857: //número 5
        color =2;
        analogWrite(ledVerde, 127);
        break;
      case 0xFD6897: //número 6
        color =2;
        analogWrite(ledVerde, 255);
        break;
      case 0xFD18E7: //número 7
        color =3;
        analogWrite(ledAzul, 0);
        break;
      case 0xFD9867: //número 8
        color =3;
        analogWrite(ledAzul, 127);
        break;
```

```
      case 0xFD58A7: //número 9
        color =3;
        analogWrite(ledAzul, 255);
        break;
      case 0xFD40BF: //FUNC/STOP
        analogWrite(ledRojo, 0);
        analogWrite(ledVerde, 0);
        analogWrite(ledAzul, 0);
        break;
      case 0xFD00FF: //Encendido
        analogWrite(ledRojo, 255);
        analogWrite(ledVerde, 255);
        analogWrite(ledAzul, 255);
        break;
      case 0xFD807F: //vol +
        if (color == 1) parpadeo(ledRojo);
        else if (color ==2) parpadeo(ledVerde);
        else if (color ==3) parpadeo(ledAzul);
        break;
    }
  }
}

void parpadeo(int pin){
  for (int i=0; i<=3; i++){
  digitalWrite(pin,HIGH);
  delay(1000);
  digitalWrite(pin,LOW);
  delay(1000);
  }
}
```

Actividad 3

Desafío práctico: Circuito básico + RGB

```
#include <IRremote.h>
#include <IRremoteInt.h>

int ReceptorIR = 2;
IRrecv receptorIr(ReceptorIR);
decode_results codigoLeido;

int ledRojo = 9;
int ledVerde = 6;
int ledAzul = 10;

int potenciaRojo = 255;
int potenciaVerde = 255;
int potenciaAzul = 255;

void setup() {
  Serial.begin(9600);
  receptorIr.enableIRIn();
```

```
  pinMode(ledRojo, OUTPUT);
  pinMode(ledVerde, OUTPUT);
  pinMode(ledAzul, OUTPUT);

  analogWrite(ledRojo, 255);
  analogWrite(ledVerde, 255);
  analogWrite(ledAzul, 255);
}

void loop() {
   if (receptorIr.decode(&codigoLeido)) {   // eliminamos la verificación de código distinto
    Serial.println(codigoLeido.value, HEX);

    switch (codigoLeido.value) {
     case 0xFD08F7: //número 1
        if (potenciaRojo > 1) potenciaRojo = potenciaRojo -1; //validamos que potencia no baje de 1
        analogWrite(ledRojo, potenciaRojo);
       break;
     case 0xFD48B7: //número 3
        if (potenciaRojo <255) potenciaRojo = potenciaRojo +1; /* validamos que potencia no sea
                                                                  mayor a 255 */
        analogWrite(ledRojo, potenciaRojo);
       break;
     case 0xFD28D7: //número 4
        if (potenciaVerde > 1) potenciaVerde = potenciaVerde -1;
        analogWrite(ledVerde, potenciaVerde);
       break;
     case 0xFD6897: //número 6
        if (potenciaVerde <255) potenciaVerde = potenciaVerde +1;
        analogWrite(ledVerde, potenciaVerde);
       break;
     case 0xFD18E7: //número 7
        if (potenciaAzul > 1) potenciaAzul = potenciaAzul -1;
        analogWrite(ledAzul, potenciaAzul);
       break;
     case 0xFD58A7: //número 9
       if (potenciaAzul <255) potenciaAzul = potenciaAzul +1;
       analogWrite(ledAzul, potenciaAzul);
       break;
     case 0xFD00FF: //Apagado
       analogWrite(ledRojo, 255);
       potenciaRojo = 255;
       analogWrite(ledVerde, 255);
       potenciaVerde=255;
       analogWrite(ledAzul, 255);
       potenciaAzul =255;
       break;
    }
    receptorIr.resume();
  }
}
```