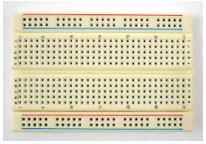
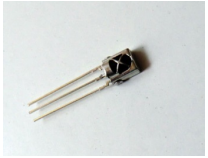
LOS MANDOS DE INFRARROJOS

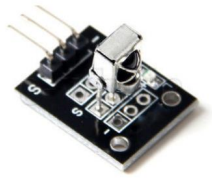
Controlando Arduino con un mando a distancia IR

OBJETIVOS

- Desmitificar la luza infrarroja.
- Presentar los receptores de infrarojos
- Aprender a usar un mando IR con tu ARduino
- Programas de ejemplo
- Instrucción switch case

MATERIAL REQUERIDO.

	Arduino UNO o equivalente.
 	Una Protoboard más cables.
	Un receptor de infrarrojos integrado como el AX 1838HS.

	Independiente o con un montaje Keyes

LOS MANDOS DE INFRARROJOS

Estamos tan acostumbrados a los **mandos a distancia infrarrojos** que no dedicamos un momento a pensar en ellos y simplemente nos parece normal, algo que a nuestros abuelos les hubiera parecido magia.

Nos parece normal que los equipos electrónicos, televisores, cadenas de música, aires acondicionados, respondan a nuestras instrucciones sin levantarnos del sofá, pero esto no ha sido siempre así, es más, es relativamente reciente, que estos mandos se popularizaron.

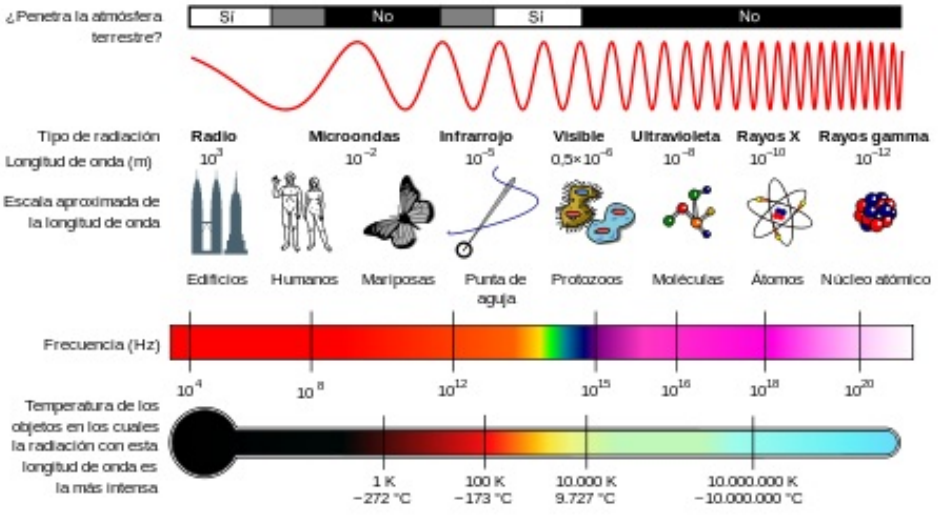
Hemos oído que funcionan por infrarrojos y poco más. Me pregunto cuántos usuarios responderían correctamente a la pregunta de que son los infrarrojos y cuál es el principio de funcionamiento de un mando a distancia de este tipo.

Para centrar un poco las ideas, empecemos diciendo que las ondas electromagnéticas se caracterizan, principalmente, por su frecuencia o lo que es lo mismo, por el inverso de ésta que es la longitud de onda.

En la parte más baja del espectro, de menor frecuencia, están los infrasonidos, y luego vienen las ondas de sonido claro. Un poco más arriba tenemos los ultrasonidos (Nombres sorprendentes, que básicamente significan por debajo y por encima de lo que se oye.

Si seguimos subiendo en la frecuencia nos encontramos con las ondas de radio y luego con las microondas. Después como veis en este grafico que he pillado de la Wikipedia viene el infrarrojo (Por debajo del rojo), el espectro de luz visible con los familiares colores (Que solo son una forma en que nuestro cerebro percibe las frecuencias de luz) y luego el ultravioleta (Por encima del violeta, eso es echarle imaginación a los nombres, si señor).

Más arriba en la escala encontramos los rayos X (Cuando el alemán Roentgen descubrió esta radiación a finales del XIX, no tenía ni idea de que era, y le parecía una buena idea llamarlos Rayos X, o sea desconocidos, y todavía hoy seguimos con este cachondeo) y por último los rayos gamma o cósmicos.



La capacidad energética de la radiación crece rápidamente con la frecuencia y por eso los rayos X y los gamma son muy dañinos para los seres vivos.

Una manera de describir los **infrarrojos**, seria como una luz con un color diferente, que no vemos, pero luz a pesar de todo.

Una curiosidad no muy conocida, es que seguramente la cámara de tu teléfono móvil, o tableta es capaz de ver, y mostrarte, la radiación IR de tus mandos a distancia. Aquí os pongo un mini video grabado con mi móvil, un iPhone 4S:

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.
- Como solemos ver la tele a oscuras, no se ve cuando usamos el mando.
- Tiene relativamente poco alcance, pero no solemos ver la tele o tener la cadena de música más allá de 2 o 3 metros.

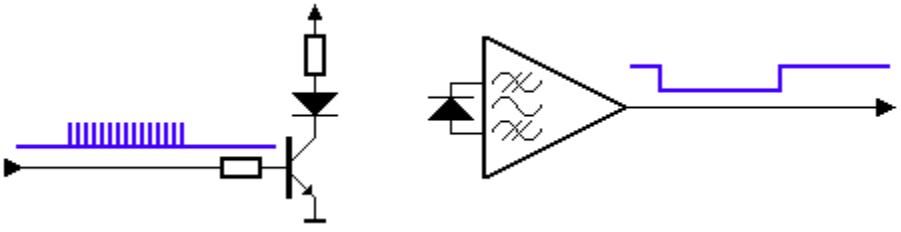
Así que es práctico, sencillo y barato (lo que les encanta a los fabricantes), aunque tienen también inconvenientes.

El principal es que cualquier cosa con una cierta temperatura, incluidos nosotros, emitimos radiación infrarroja. Es por eso que las cámaras IR que veis en las pelis, pueden mostrar nítidamente en plena oscuridad a una persona o animal.

Y esto podría interferir con el mando a distancia IR, lo mismo que cosas como la calefacción, el sol y demás cosas calientes. Así que la solución para evitarlo es modular la señal con una portadora.

La idea básica es mandar un tren de ondas estable (La portadora) y mezclarlo con la información que queremos enviar (La señal). Este mismo principio se usa con la radio y casi con cualquier señal radioeléctrica que se envié por el aire.

He encontrado este Gif en la página de sbprojects y me ha parecido ideal para mostrar el proceso de modulación y transmisión.



El emisor es un sencillo transistor que gobiernan un LED infrarrojo muy similar a los LEDs rojitos normales que hemos usado hasta ahora, solo que diseñados para emitir luz en un color que no vemos.

Un pequeño procesador en el mando gobierna, la generación de la señal y la mezcla con la portadora, para garantizar que nuestra tele no recibe órdenes espurias.

- El proceso de mezclar una señal con la portadora se le llama modular.
- El inverso, extraer la señal de una onda RF y obtener la señal limpia se llama demodular.

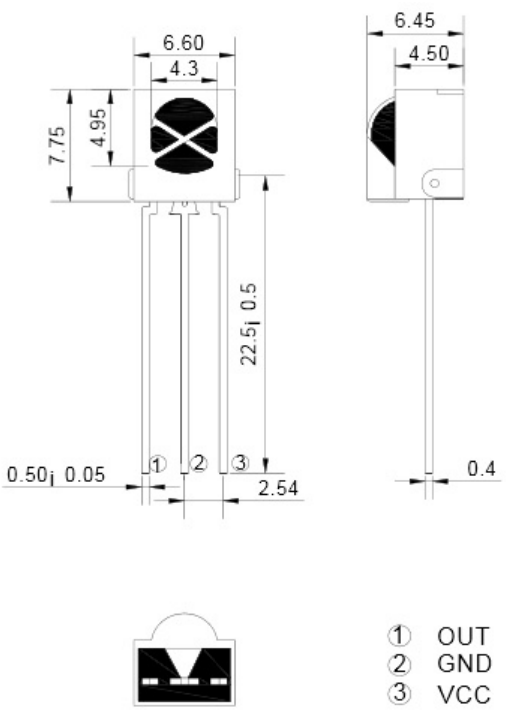
El receptor tiene un poco más de complejidad porque, las normas de emisión de IR para mandos a distancia, aparecieron como setas tras una tormenta y poco menos que cada fabricante presento su propia norma.

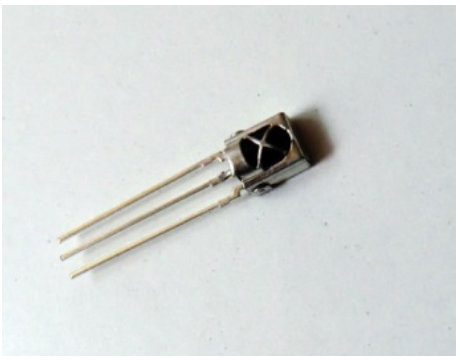
Al final la industria acabo diseñando unos receptores capaces de recibir y demodular casi cualquier cosa, y como se venden como churros (Porque nos encanta mantener nuestro culo pegado al sofá) valen muy poco y son un prodigio tecnológico.

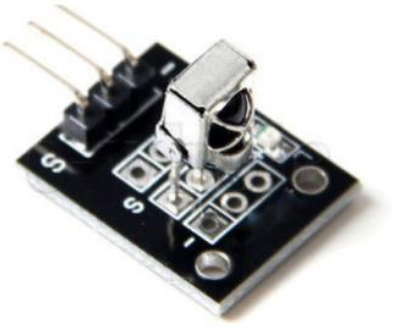
Un receptor IR típico actual, incluye el receptor, amplificador demodulador y lo que se te ocurra encapsulado y listo para usarse. No os imagináis la cantidad de electrónica que va incluido esta piececita de aspecto inocente.

RECEPTORES DE IR

Un típico receptor de infrarrojos, es el AX-1838HS, que se consigue por poco más de unos euros. En su hoja de normas encontráis este diagrama, y son relativamente fáciles de encontrar, en dos formas. Independientes y montados en un soporte para utilizar sin complicaciones con nuestros Duinos.







El esquema de conexión nuevamente es trivial, pues el AX-1838HS solo tiene 3 pines: Vcc, GND y señal. Como vamos a usar una interrupción para leer la señal es imprescindible, que llevéis el pin de señal a el pin2 de Arduino (La interrupción 0) o al pin3 (La interrupción 1).

Yo lo voy a conectar al pin 2 y así lo reflejan los programas y los ejemplos.

PROGRAMA DE CONTROL

Para poderlos usar de una forma cómoda, vamos a descargarnos alguna librería, que nos facilite la vida. De las varias disponibles me ha gustado la de Nico Hood http: //nicohood.wordpress.com/, por varios motivos.

- Es ligera y muy rápida en decodificar las señales IR y además usa muy poca memoria.
- Aunque no decodifica todo, acepta las normas de NEC y Panasonic y compatibles, y además tiene un modo de haz lo que puedas que parece ir bastante bien. Lo que nos da un 85% de probabilidades de que reconozca tu mando
- Funciona por interrupciones, lo que la hace muy ligera y rápida y así vemos un buen ejemplo de uso de estas.
- Es de las pocas que es capaz de decodificar las señales simultáneas de varios mandos diferentes, con diferentes protocolos cada uno.

Lo primero es descargarnos la librería IRLremote.zip, y después instalarla siguiendo el procedimiento habitual que vimos en sesiones anteriores. Una vez hecho para usarla como siempre usamos el importar librería, que os pondrá:

```
#include "IRLremote.h"
```

Y para inicializarla:

```
IRLbegin<IR_ALL>(interruptIR);
```

Vamos a empezar por un ejemplo que nos recomienda el autor para reconocer el mando a distancia que usamos. Podeis cargarlo con ejemplos IRLremote\ReceiveInterrupt

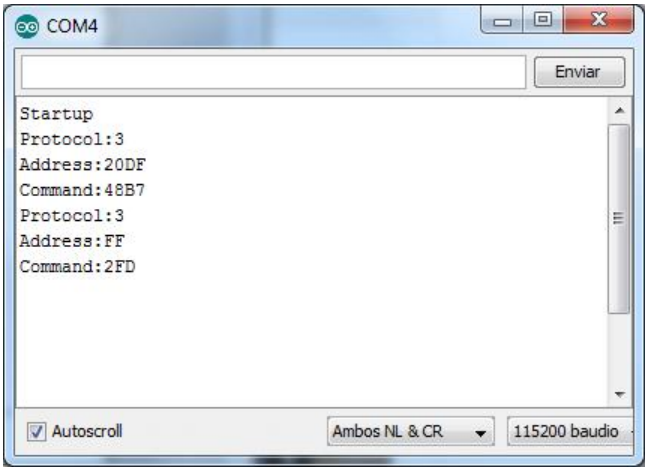
```
#include "IRLremote.h"
const int interruptIR = 0;                // Arduino interrupcion 0: Pin 2

uint8_t IRProtocol = 0;  // Variables para recibir los datos
uint16_t IRAddress = 0;
uint32_t IRCommand = 0;

void setup()
   {     Serial.begin(115200);  // Fijate en la velocidad
         Serial.println("Startup");
         IRLbegin<IR_ALL>(interruptIR);
   }
void loop()
   {
         uint8_t oldSREG = SREG;  // Parar las interrupciones
         cli();
         if (IRProtocol)          // Si reconoce el protocolo
            {
                 Serial.print("Protocol:");
                 Serial.println(IRProtocol);
                 Serial.print("Address:");
                 Serial.println(IRAddress, HEX);
                 Serial.print("Command:");
                 Serial.println(IRCommand, HEX);
                 IRProtocol = 0;
            }
         SREG = oldSREG;
    }

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

Este programa, simplemente tratará de leer tu mando a distancia y enviar esa información a la consola. Asegúrate de que has puesto la velocidad a 115200:



Como veréis la rutina de servicio de la interrupción simplemente recoge los valores del protocolo que amablemente reconoce sobre la marcha la librería. Los valores que presenta son

- Nos informa del modelo de señal que usa nuestro mando. En este momento reconoce 4 protocolos y el autor pide la colaboración ciudadana, para aumentar el número de protocolos disponibles con la librería. Actualmente son:
  - IR_NO_PROTOCOL , 0
  - IR_USER, 1
  - IR_ALL, 2
  - IR_NEC, 3.
  - IR_PANASONIC, 4
- Address, dirección de tu mando. Cada mando lleva una dirección para identificarle. Podrías usar más de un mando a distancia para controlar lo que sea, y reconocerlos de forma diferenciada.
- La orden o identificación del botón pulsado en el mando. Cada botón tiene un código diferente y estos son los códigos que usaras en tu programa para lanzar acciones diferentes en función del botón pulsado.

Yo he probado con 5 mandos IR diferentes (Tengo muchos trastos) y me ha reconocido correctamente 4 a la primera. Los que han entrado sin problemas son los de Panasonic, Sony, LG y Keyes.

El que no sabía lo que era, ha sido un mando viejo de una TV Schneider que soy incapaz de saber de dónde ha salido.

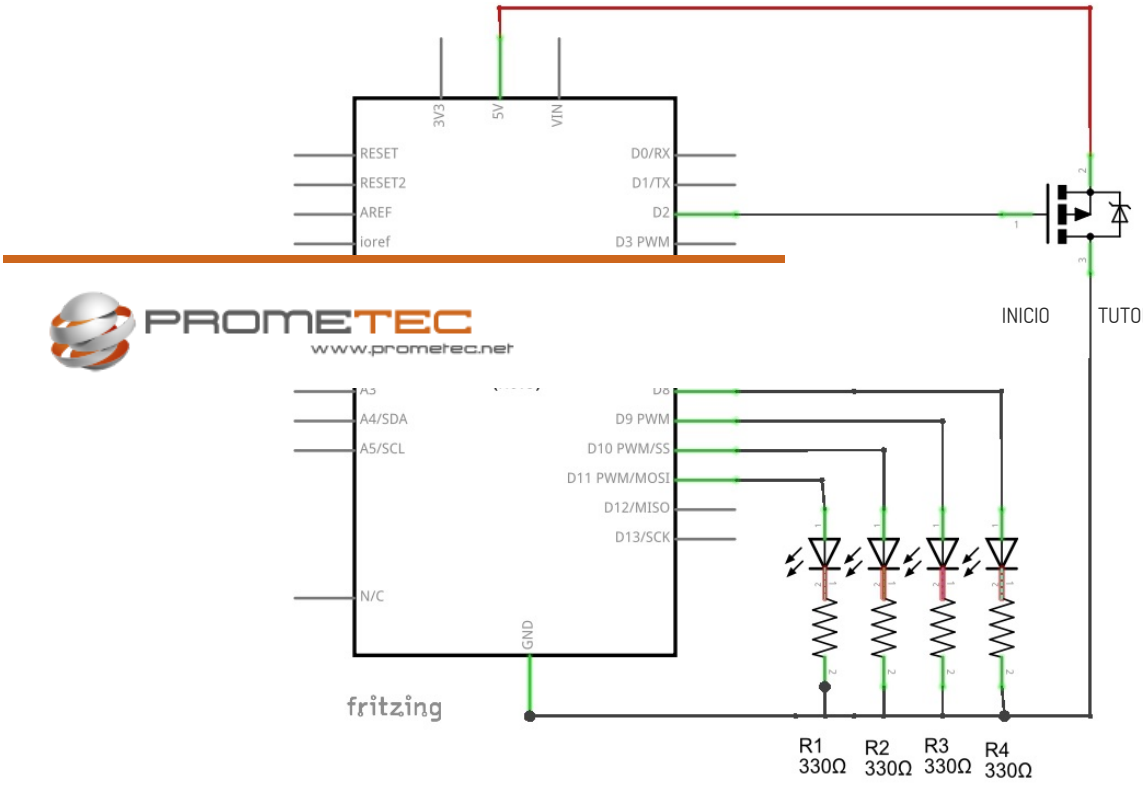
- Conviene comentar que si tienes una tele, o VHS o similares que ya no usas, puedes tirar el aparto si quieres pero conserva el mando porque hay muchas probabilidades de que puedas leerlo con tu Arduino.
- Además tus antiguos cacharros con mando a distancia incluyen esto receptores de IR que puedes desarmar y reutilizar para tus cosas. (Os recomiendo desarmar solo los aparatos viejos e inútiles, no los que están funcionando, que luego no quiero responsabilidades).

Y en este programa está ya implícito todo lo necesario para utilizar un mando IR en tus proyectos. EL resto de la sesión será sencillamente un ejemplo de cómo gestionar esos golpes de teclas con u Swuitch que es una instrucción C++ que hasta ahora no habíamos usado y que nos viene al pelo.

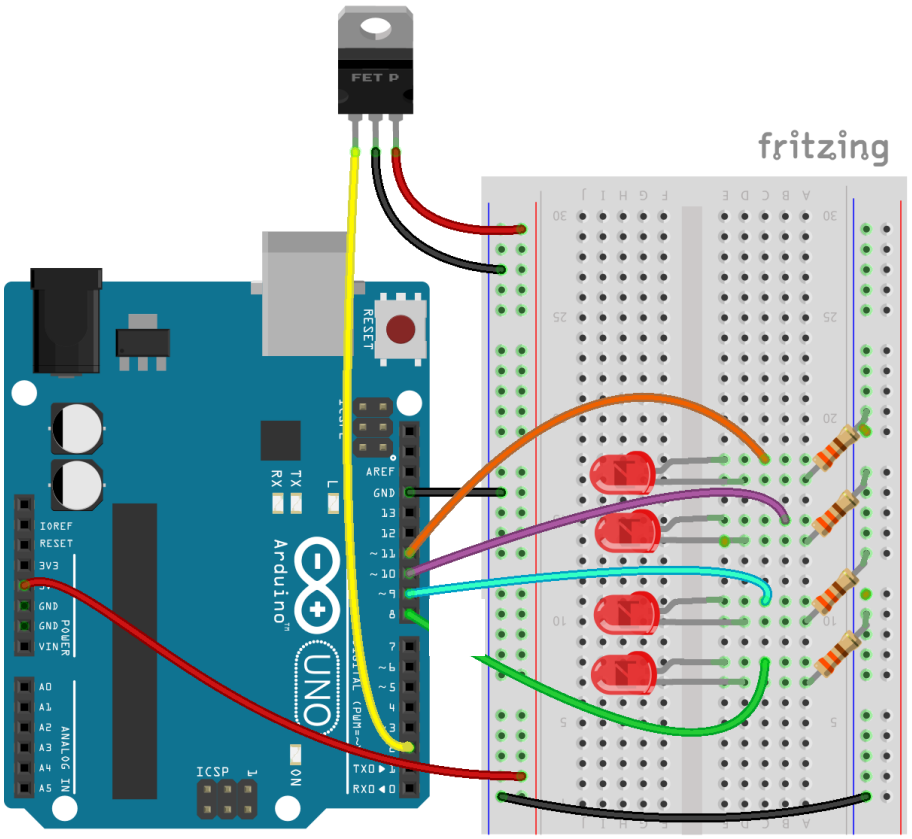
Vamos a montar un circuito con Arduino y 4 LEDs de colores para encerderlos de uno en uno o apagarlos todos de golpe, mediante el mando IR a distancia.

ESQUEMA DE CONEXIÓN

La conexión será muy sencilla:



Y el esquema de protoboard, cuesta más dibujarlo que montarlo:



PROGRAMA DE CONTROL

Al principio de esta sesión vimos que ya recibíamos las instrucciones del mando (Espero) y nos ponía en la consola el valor del protocolo, dirección del mando y Commando que corresponde al botón pulsado. Estos comandos son lo que necesitais para diparar vuestro programa

Si tienes mas de un mando y quieres usarlos, tendras que controlar también la dirección de cada mando.

En primer lugar correr el programa anterior para anotar los códigos que corresponden a cada tecla de vuestro mando, y haceros una tabla. No hace falta que anoteis todas las teclas por ahora, pero si al menos las que vayamos a usar.

En mi mando LG el resultado es este:

ARRIBA	ABAJO	IZQUIERDA	DERECHA	OK
0x02FD	0x827D	0xE01F	0x609F	0x22DD

Y con esto ya podemos escribir el programa de control. Vamos a utilizar la instrucción switch case, que es muy comoda cuando tienes que comprobar muchos valores y tomar acciones en respuesta a una variable. LA idea básica es como esto:

```
switch (IRCommand)
   {   case 0x02FD:
            Serial.println("Arriba");
            digitalWrite(8, HIGH);
            break;
case 0x827D:
            Serial.println("Abajo");
            digitalWrite(9, HIGH);
            break;
case 0xE01F:
            Serial.println("Izquierda");
            digitalWrite(10, HIGH);
            break;
case 0x609F:
            Serial.println("Derecha");
            digitalWrite(11, HIGH);
            break;
case 0x22DD:
            Serial.println("OK");
            for (int k =0 ; k <12 ; k++)
                 digitalWrite(k, LOW);
             break;
}
```

Le pasamos a switch la variable, cuyo valor determina la acción, en este caso el comando pulsado. Y después se establece una clausula de case para cada valor posible finalizando con dos puntos. Escribe las instrucciones a ejecutar y finaliza con break.

He puesto a propósito lo del Serial.println() para que veais que podeis poner varias líneas sin necesidad de bloques que ya esta implícito en el case. Y por fin finaliza la clausula con un break.

Puedes incluir una ultima clausula default, sin otro valor para poner una acción que se ejecute si el valor de la variable no coincide con niguno de los caso comprobados mas arriba.

Aquí teneis un programa de ejemplo: **EJEMPLO 52_3**

```
#include "IRLremote.h"
const int interruptIR = 0;

uint8_t IRProtocol = 0;  // Variables para recibir los datos
uint16_t IRAddress = 0;
uint32_t IRCommand = 0;
```

```
void setup()
   {    Serial.begin(115200);          // Fijate en la velocidad
        for (int i = 8 ; i<12 ; i++)
        pinMode(i, OUTPUT);

        IRLbegin<IR_ALL>(interruptIR);
   }
```

La unica novedad ha sido hasta aqui, definer como salidas los pines del 8 al 11

```
void loop()
   {    uint8_t oldSREG = SREG;  // Parar las interrupciones
        cli();
        if (IRProtocol)
          {
            switch (IRCommand) // Aqui tenemos la clausula switch con sus case:
                { case 0x02FD:
                      Serial.println("Arriba");
                      digitalWrite(8, HIGH);
                      break;
                  case 0x827D:
                      Serial.println("Abajo");
                      digitalWrite(9, HIGH);
                      break;
                  case 0xE01F:
                       Serial.println("Izquierda");
                       digitalWrite(10, HIGH);
                       break;
                  case 0x609F:
                       Serial.println("Derecha");
                       digitalWrite(11, HIGH);
                       break;
                  case 0x22DD:
                       Serial.println("OK");
                       for (int k =0 ; k <12 ; k++)
                            digitalWrite(k, LOW);
                       break;
                }
            IRProtocol = 0;
          }
       SREG = oldSREG;
    }
void IREvent(uint8_t protocol, uint16_t address, uint32_t command)
   {
        IRProtocol = protocol;  // Recogemos los valores
        IRAddress = address;
        IRCommand = command;
   }
```

La cláusula switch es muy útil para evitar if else encadenados que al final te hacen perder el hilo. Imagínate si tienes que comprobar todas las posibles teclas del mando.

Aquí teneís un minivideo con el resultado

Por ultimo y antes de terminar, comentar que el autor de la librería recomienda que si vamos a usar un solo mando, sustituyamos la líne

```
IRLbegin<IR_ALL>(interruptIR);
```

Por la que corresponda a nuestro modelo, porque ahorraremos en memoria y ganaremso en velocidad en la decodificación de las señales. En mi caso que me reporta el protocolo 3 , IR_NEC sería:

```
IRLbegin<IR_NEC >(interruptIR);
```

RESUMEN DE LA SESIÓN

- Tenemos una idea mas clara de lo que es la luz infrarroja.
- Hemos presentado los receptore de mandos IR AX-1838HS.
- Hemos visto que podemos usar los mandos que tengamos disponibles y que nuestro Arduino es capaz de leer la mayoria de ellos
- Vimos la instruccion switch case

ANTERIOR SIGUIENTE

(3) COMMENTS

Jose *05 mar 2015*

Has puesto los pines de GND y Vcc del sensor al revés en el esquema gráfico. Cuidado porque quemas el sensor (te lo digo por experiencia, vi cómo humeaba y ahora no funciona :D).
Gracias por los tutoriales, de lo mejorcito que he visto!
Un saludo,

Reply

Admin *06 mar 2015*

Muchas gracias pòr el aviso Jose,
Ya he corregido el diagrama para evitar futuros percances. Siento mucho lo de tu sensor

Reply

Andres *26 mar 2015*

Muy clarito todo. Mi proyecto personal es con un mando contrllar 8 Relay que a su vez controlaran 8 funcíones en un mismo aparato. Es unmagnetofono Revox,en aquellos tiempos no existian estos aditamentos, por lo tanto debo redefinir las funciones de algunas teclas.

A-On/Off esta es la unica que debe quedar encendida hasta que le vuelva a dar un segundo toque.

B- Las demas solo deben permencer acticas no mas de UN segundo para que estimulen la entrada de otro microcontrolador que si tiene el aparato.

Como veis ando recogiendo informacion de todos lados para llevar adelante el proyecto.

C- Como parte final del tema pretendo “clonar” el arduino con el programa y armarlo en una PCB para que quede definitiva en el aparato.

Si teneis alguna idea os agradeceria, es que donde mayores problemas tengo es en los ‘Codigos’ o sea la estructura gramitacal del mismo.

Pues muy Exito en todo!

Reply

GIVE A REPLY

Name (required)

Email (required)

URL

Message

Sí, agrégame a tu lista de correos.

Post comment

CATEGORIAS DE LOS PRODUCTOS

Selecciona una categoría