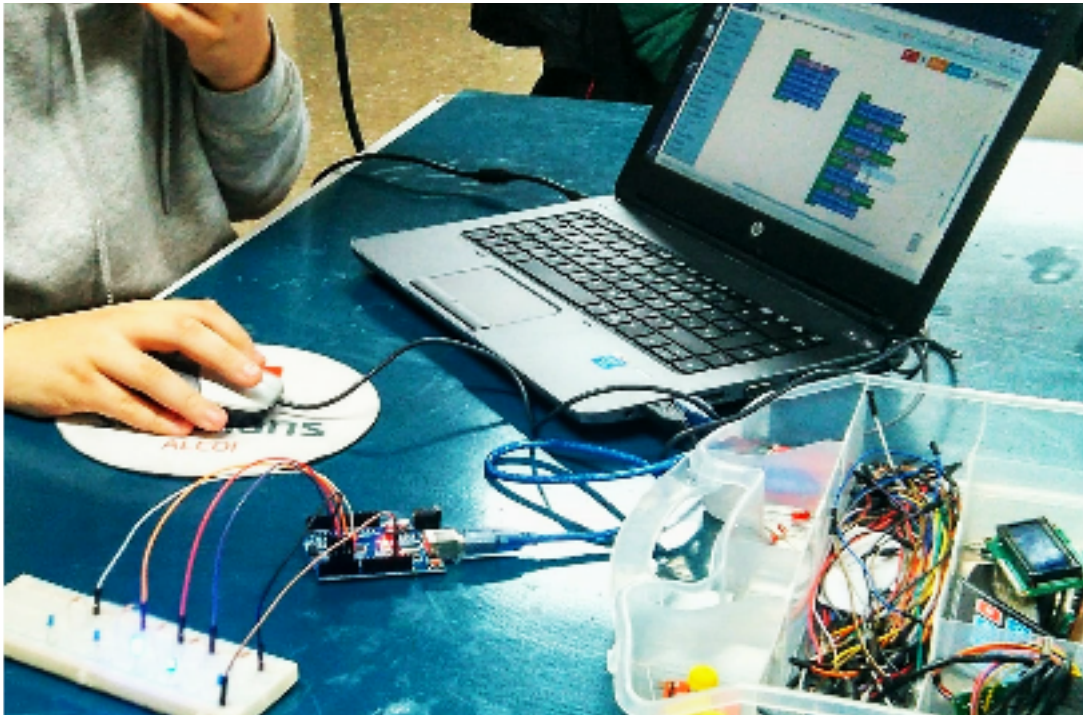


# 40

## PROYECTOS RESUELTOS



Juan José López Almendros

[arduinoblocks.com](http://arduinoblocks.com)

# PROYECTOS

A continuación se detallan 40 proyectos desarrollados en ArduinoBlocks con esquemas y programas de bloques.

La funcionalidad de cada proyecto está simplificada, el objetivo es mostrar las posibilidades de la plataforma con aplicaciones reales sencillas.

En la web de ArduinoBlocks podemos encontrar proyectos realizados por otros usuarios (proyectos compartidos) que nos sirvan también como inspiración o punto de partida para nuestros propios proyectos.

Los 40 proyectos de este libro se encuentran compartidos y accesibles en la web de ArduinoBlocks.  
<http://www.arduinoblocks.com/web/site/search>

## Listado de proyectos resueltos:

- P01 - Secuenciador de luces
- P02 - Simulación amanecer y anochecer
- P03 - Lámpara con regulación manual
- P04 - Semáforo
- P05 - Timbre
- P06 - Control inteligente de iluminación
- P07 - Encendido automático por movimiento
- P08 - Contador manual
- P09 - Cronómetro
- P10 - Fotómetro
- P11 - Iluminación crepuscular
- P12 - Encendido y apagado con palmada
- P13 - Termómetro
- P14 - Termostato
- P15 - Medidor de distancia
- P16 - Riego automático
- P17 - Lámpara multicolor con control IR
- P18 - Piano con teclado
- P19 - Sensor de aparcamiento
- P20 - Control pan/tilt con joystick
- P21 - Control de un led desde PC (consola)
- P22 - Control de relés por Bluetooth
- P23 - Estación meteorológica Bluetooth
- P24 - Control de led por voz (Android+Bluetooth)
- P25 - Control domótico (Android+Bluetooth)

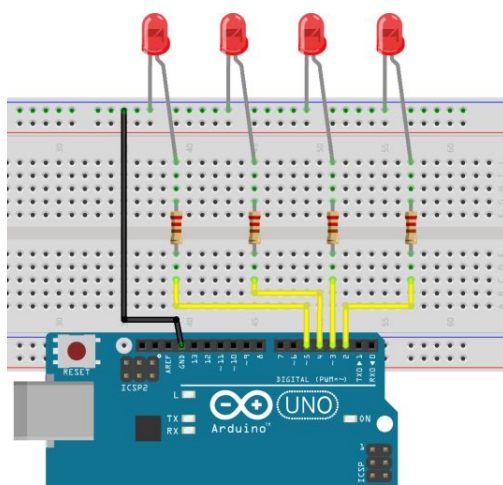
- P26 - Visualización GPS en LCD
- P27 - Aviso por exceso de velocidad
- P28 - Alarma por alejamiento
- P29 - Registrador GPS en tarjeta SD
- P30 - Registro de temperatura y humedad en tarjeta SD
- P31 - Control de servo con acelerómetro
- P32 - Sensor de caída con aviso a emergencias vía Bluetooth
- P33 - MQTT (IoT): Control de led RGB
- P34 - MQTT (IoT): Estación meteorológica
- P35 - MQTT (IoT): Control domótico
- P36 - Robot con servos controlador por Bluetooth
- P37 - Robot con motores DC - Bluetooth
- P38 - Robot con motores DC – Evita obstáculos
- P39 - Robot con motores DC – Sigue líneas
- P40 - Brazo robótico con servos – Control PC (consola)

## P01 - Secuenciador de luces

Un secuenciador es capaz de repetir ciclos de encendido y apagado de leds siguiendo un orden para lograr un efecto visual llamativo y divertido. Podemos utilizar nuestro secuenciador de luces para decorar el árbol de Navidad, realizar carteles luminosos llamativos o simplemente para animar una fiesta con los amigos.

### Material necesario:

- 4 x leds de los colores deseados.
- 4 x resistencias de 220 Ω.
- Placa de prototipos, cables de interconexión.



### Conexiones:

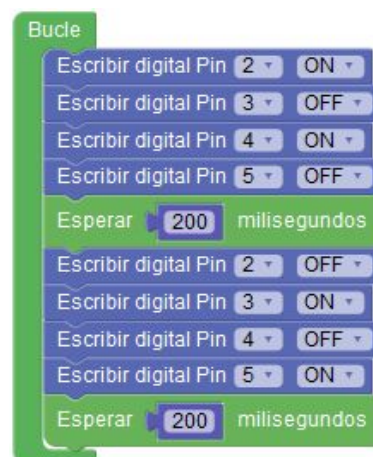
- Led 1 = Pin 2
- Led 2 = Pin 3
- Led 3 = Pin 4
- Led 4 = Pin 5

Programa ArduinoBlocks:

### *Secuencia 1:*



### *Secuencia 2:*





## P02 - Simulación de amanecer y anochecer

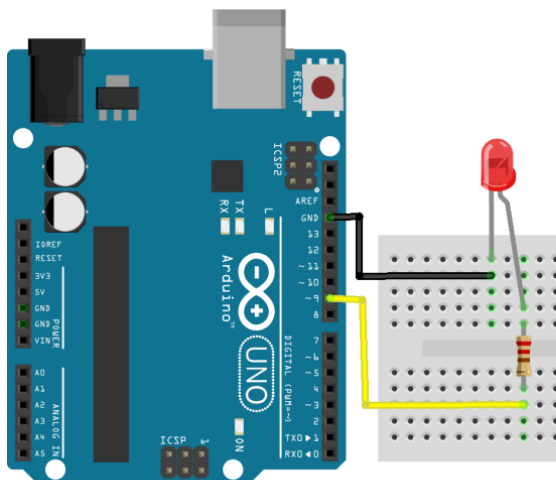
Con este proyecto vamos a simular el ciclo solar de anochecer y amanecer donde la luz disminuye o aumenta progresivamente de forma suave.

Aplicaciones de ejemplo:

- Belén Navideño con simulación de día/noche
- Aviario para cría de aves

### Material necesario:

- 1 x led
- 1 x resistencia de 220  $\Omega$ .
- Placa de prototipos, cables de interconexión.



Conexiones:  
Led = Pin ~9

Programa ArduinoBlocks:

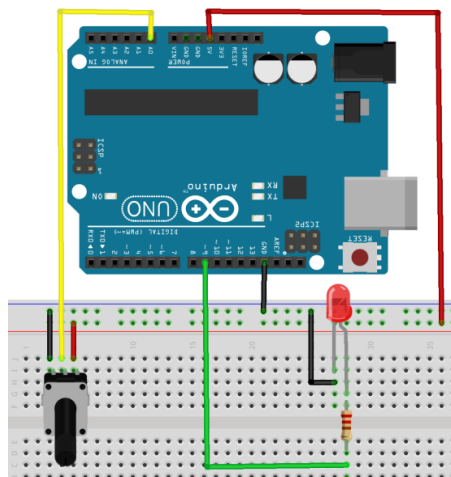


## P03 - Lámpara con regulación de intensidad manual

Mediante el uso de un potenciómetro rotativo vamos a controlar la intensidad de iluminación de un led.

### Material necesario:

- 1 x led
- 1 x resistencia de 220  $\Omega$ .
- 1 x potenciómetro 10k
- Placa de prototipos, cables de interconexión.



### Conexiones:

Led = Pin ~9

Potenciómetro = Pin A0

Programa utilizando el bloque de potenciómetro (0-100%):

```

Bucle
  Establecer potenciómetro = Potenciómetro % Pin A0
  Establecer intensidad = mapear potenciómetro de 0 - 100 a 0 - 255
  Escribir analógica (PWM) Pin 9 Valor potenciómetro
  
```

Programa utilizando el bloque genérico de lectura de entrada analógica (0-1023):

```

Bucle
  Establecer potenciómetro = Leer analógica Pin A0
  Establecer intensidad = mapear potenciómetro de 0 - 1023 a 255 - 0
  Escribir analógica (PWM) Pin 9 Valor potenciómetro
  
```

Modificando el mapeo del valor leído al valor enviado al led podemos invertir el sentido de giro para aumentar o disminuir la intensidad del led en sentido contrario:

```

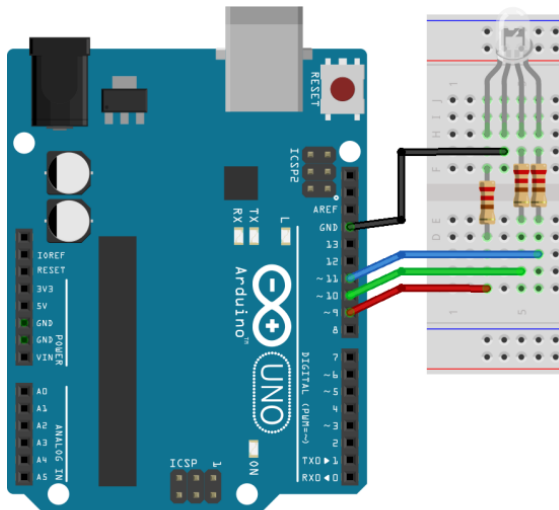
Establecer intensidad = mapear potenciómetro de 0 - 100 a 255 - 0
  
```

## P04 - Semáforo

Con la ayuda de un led RGB vamos a realizar un proyecto que simule el funcionamiento de un semáforo. El semáforo tendrá un tiempo en verde para permitir el paso, un tiempo pequeño en naranja parpadeando marcando peligro y un tiempo en rojo prohibiendo el paso.

### Material necesario:

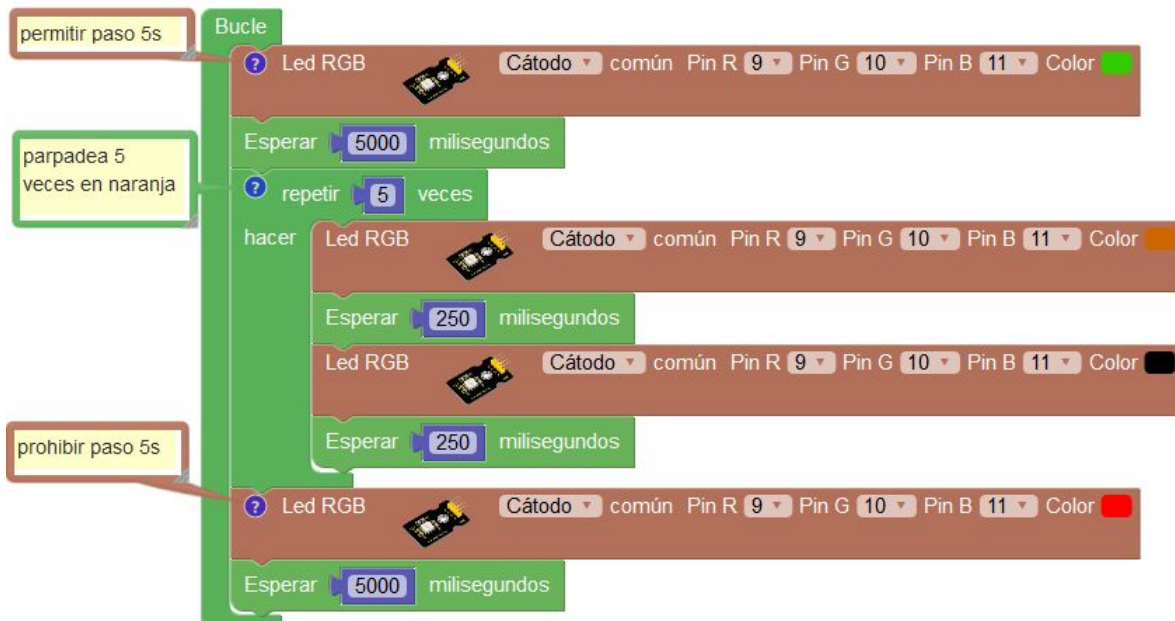
- 1 x led RGB de cátodo común
- 3 x resistencia de 220  $\Omega$ .
- Placa de prototipos, cables de interconexión.



### Conexiones:

Led R = Pin ~9  
Led G = Pin ~10  
Led B = Pin ~11

### Programa ArduinoBlocks:

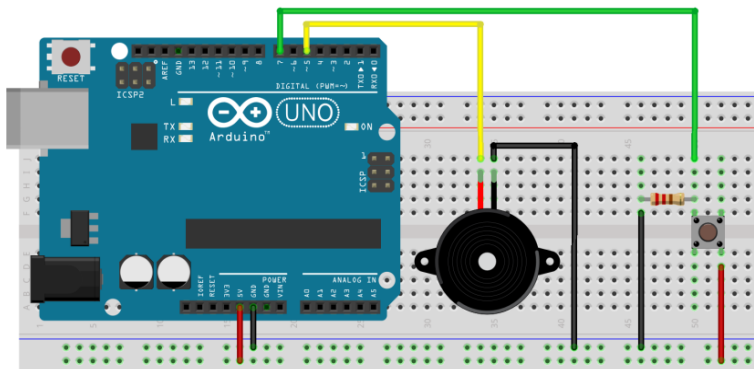


## P05 - Timbre

Con este sencillo proyecto vamos a realizar un timbre para casa, al detectar el pulsador presionado reproduciremos una melodía con el zumbador.

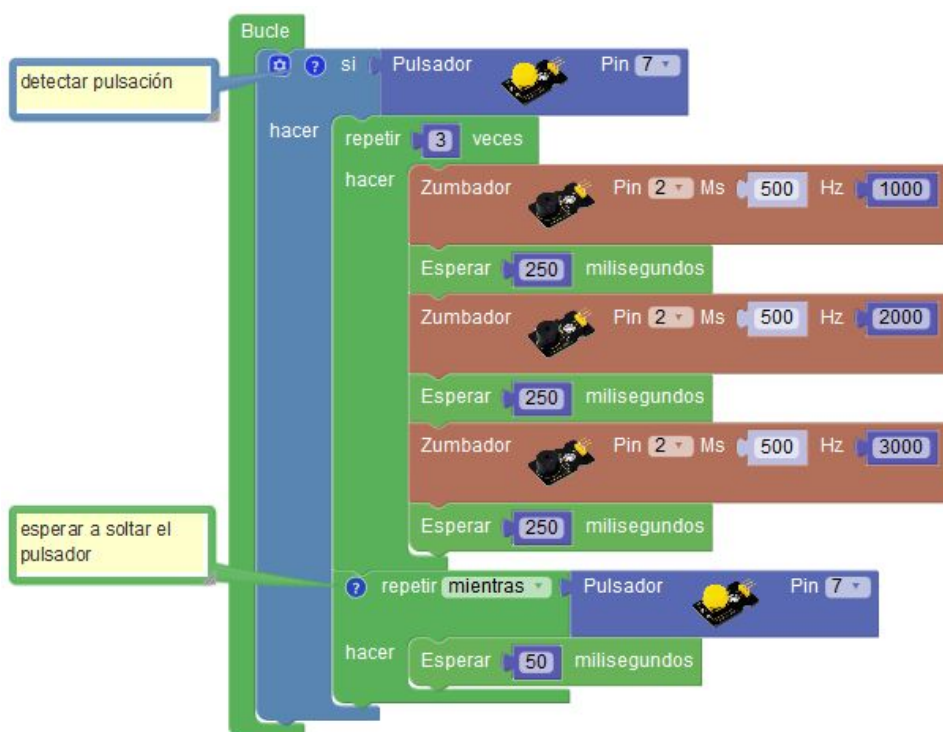
### Material necesario:

- 1 x zumbador
- 1 x pulsador
- 1 x resistencia 10 kΩ
- Placa de prototipos, cables de interconexión.



Conexiones:  
Zumbador = Pin 5  
Pulsador = Pin 7

Programa ArduinoBlocks:



Si el pulsador funciona de forma lógica inversa (normal = "ON" / pulsado = "OFF") sólo haría falta negar el estado del pulsador:

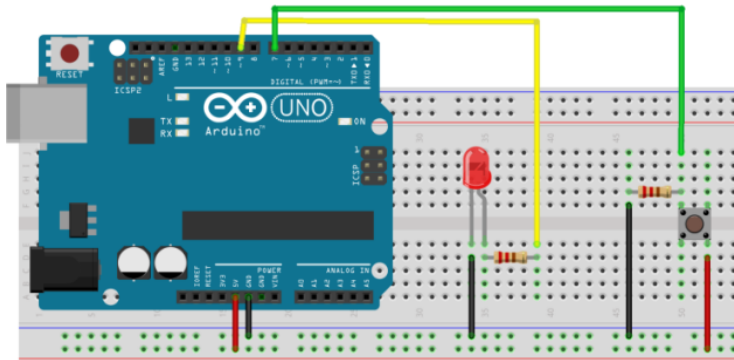


## P06 - Control inteligente de iluminación

Vamos a realizar un proyecto donde con un único pulsador podemos encender, apagar o regular la intensidad de un led. Con una pulsación corta encenderemos o apagaremos el led. Con una pulsación larga aumentaremos en pequeños incrementos la intensidad de iluminación del led.

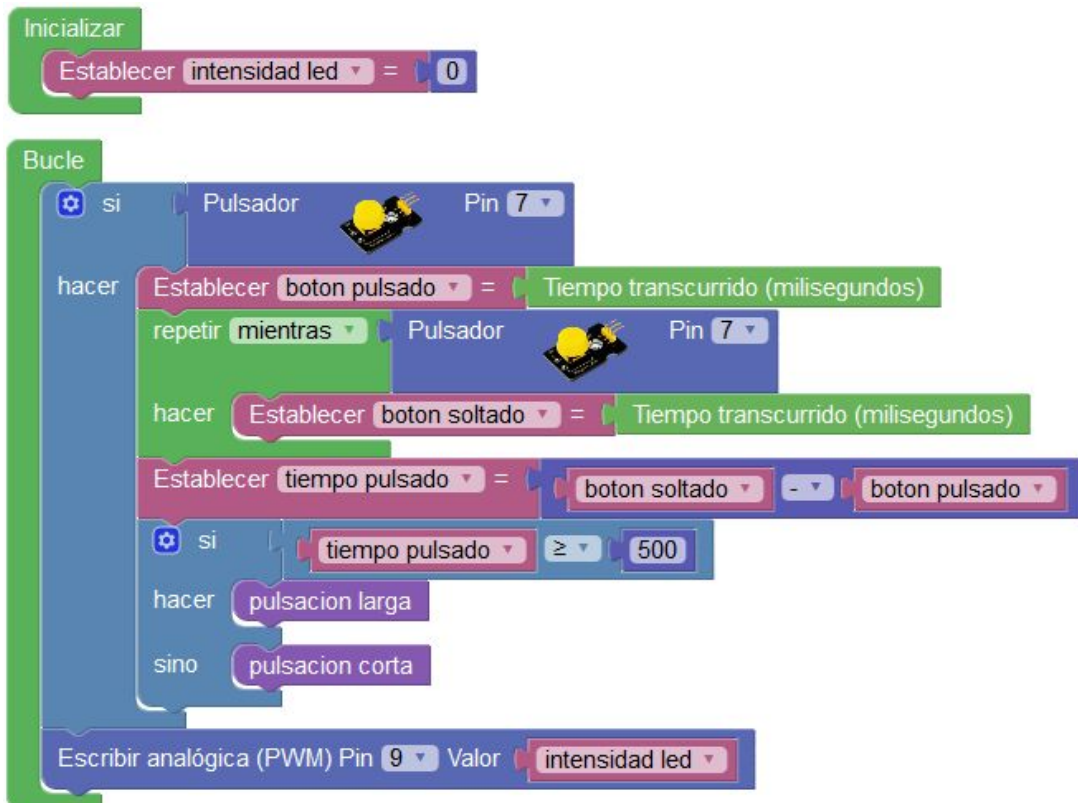
### Material necesario:

- 1 x led
- 1 x pulsador
- 1 x resistencia de  $220\Omega$
- 1 x resistencia  $10\text{ k}\Omega$
- Placa de prototipos, cables de interconexión.



Conexiones:  
Led = Pin ~9  
Pulsador = Pin 7

Programa ArduinoBlocks:





```

para pulsacion corta
  si
    intensidad led = 0
  hacer
    Establecer intensidad led = 255
  sino
    Establecer intensidad led = 0

para pulsacion larga
  cambiar intensidad led por 15
  Establecer intensidad led = limitar intensidad led entre 0 y 255
  
```

Si el pulsador funciona de forma lógica inversa (normal = "ON" / pulsado = "OFF") sólo haría falta negar el estado del pulsador:

```

no Pulsador Pin 7
  
```

Si queremos ajustar el tiempo para la pulsación larga podemos modificar el valor fijo de 500 por otro valor a nuestro gusto. Por ejemplo para detectar pulsaciones más largas, por ejemplo de 3 o más segundos, realizaríamos el siguiente cambio:

```

si
  tiempo pulsado ≥ 3000
  hacer
    pulsacion larga
  sino
    pulsacion corta
  
```

Encadenando varias condiciones podríamos detectar pulsaciones de distintos tiempos:

```

si
  tiempo pulsado ≥ 5000
  hacer
    pulsacion extra larga
  sino si
    tiempo pulsado ≥ 2000
  hacer
    pulsacion larga
  sino
    pulsacion corta
  
```

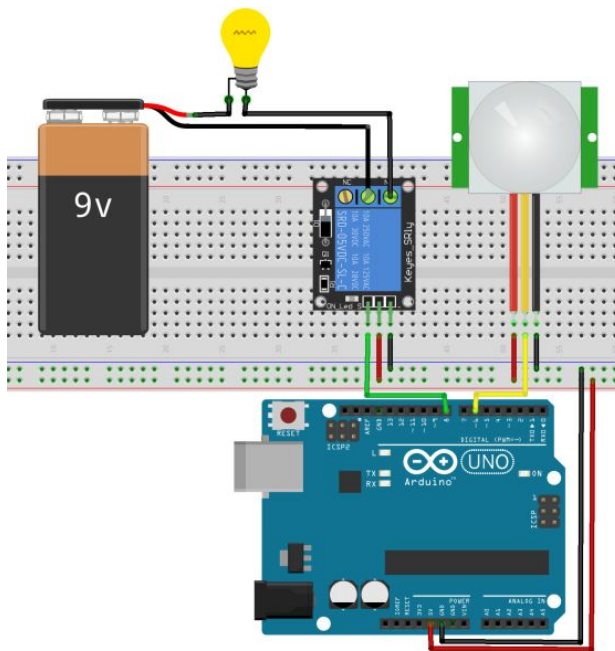


## P07 - Encendido automático por movimiento

El ahorro energético es cada vez más importante. Por eso con este sistema además de comodidad evitamos dejarnos la luz encendida. El sistema automáticamente activará la luz cuando detecte la presencia de una persona y transcurrido un tiempo a partir de dejar de detectar la presencia la luz se apagará.

### Material necesario:

- 1 x módulo de relé
- 1 x módulo de detección de movimiento PIR
- Placa de prototipos, cables de interconexión.



### Conexiones:

Relé = Pin 8  
Sensor PIR = Pin 6

Programa ArduinoBlocks:



**IMPORTANTE:** Los detectores PIR en muchos casos incorporan unos potenciómetros para ajustar el retardo y la sensibilidad. Un mal ajuste puede hacer que nuestro montaje no funcione de la forma deseada.

## P08 - Contador manual

Mediante un pulsador iremos incrementando un contador que se visualizará en una pantalla LCD. Si hacemos una pulsación larga (5s o más) se reiniciará el contador para empezar una nueva cuenta.



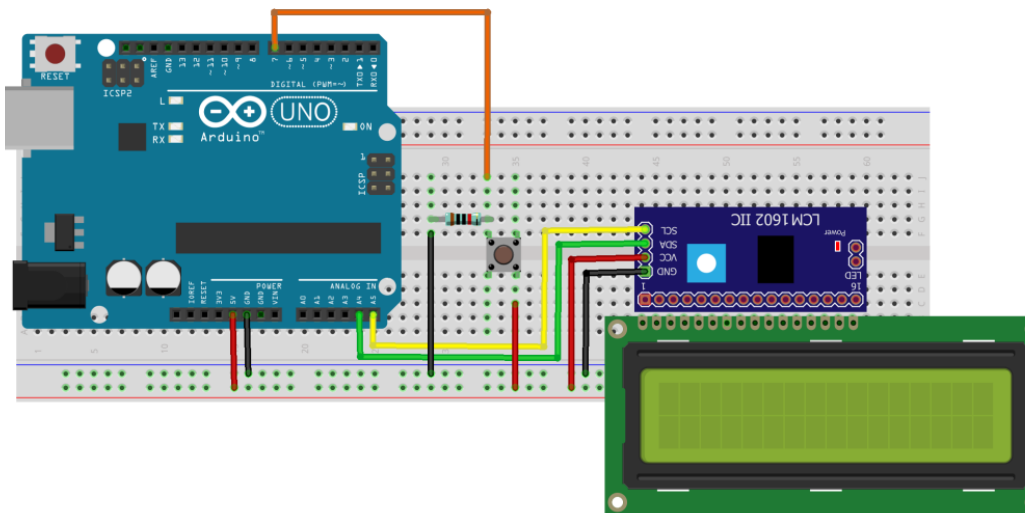
### Material necesario:

- 1 x pantalla LCD 2x16 (con módulo de conexión i2c)
- 1 x pulsador
- 1 x resistencia 10k $\Omega$
- Placa de prototipos, cables de interconexión.

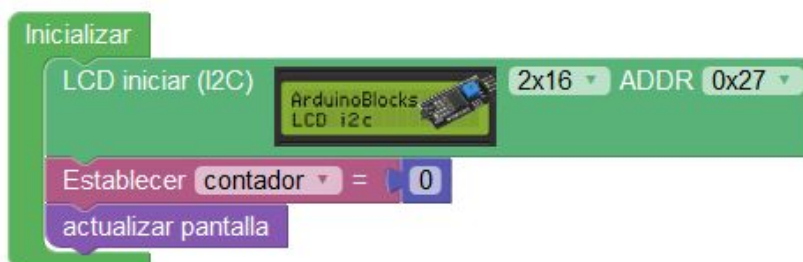
### Conexiones

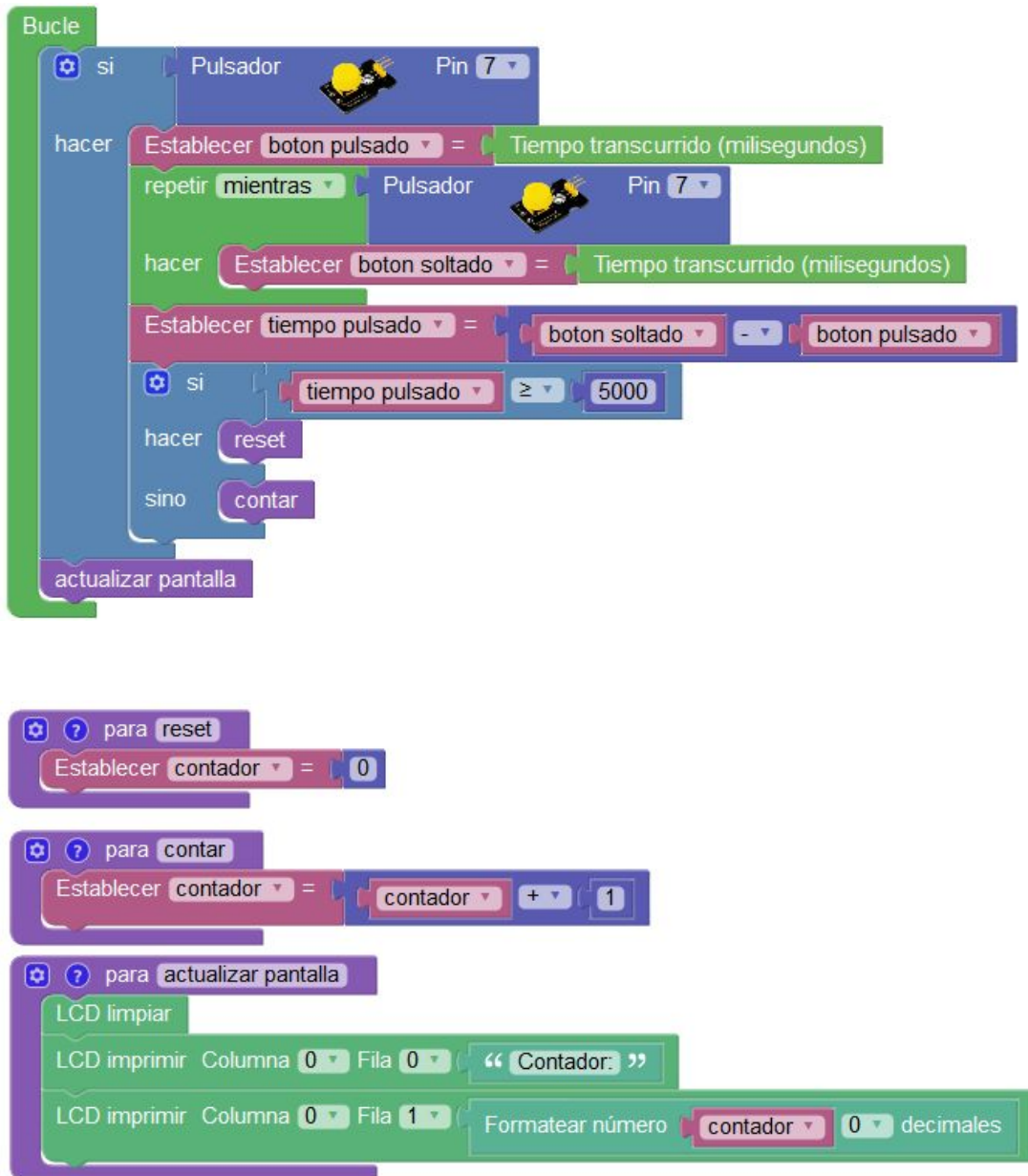
Pulsador = Pin 7

LCD (i2c) = Pin SDA (A4) , Pin SCL (A5)



Programa ArduinoBlocks:



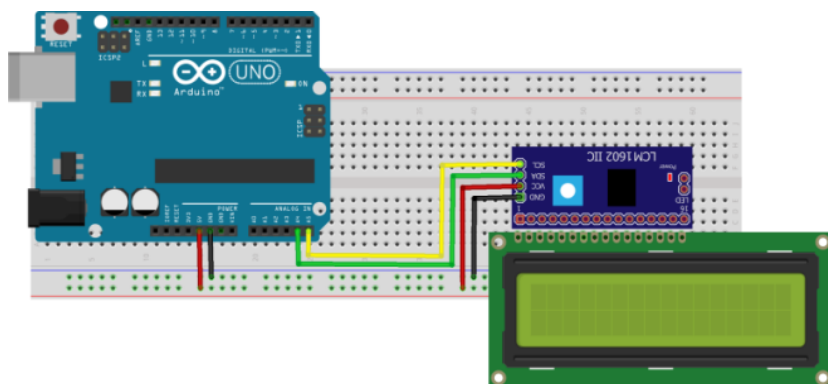


## P09 – Cronómetro / Cuenta atrás

Con sólo una pantalla LCD crearemos un cronómetro capaz de contar horas, minutos y segundos. Con el mismo montaje se implementan los programas de cuenta hacia adelante y de cuenta hacia atrás.

### Material necesario:

- 1 x pantalla LCD 2x16 (con módulo de conexión i2c)
- Placa de prototipos, cables de interconexión.



### Conexiones LCD:

Pin SDA (A4)  
Pin SCL (A5)

Programa ArduinoBlocks:

```

Inicializar
  LCD iniciar (I2C) 2x16 ADDR 0x27
  Establecer segundos = 0
  Establecer minutos = 0
  Establecer horas = 0
  actualizar pantalla

Bucle
  Ejecutar cada 1000 ms
  Establecer segundos = segundos + 1
  si segundos = 60
  hacer
    Establecer segundos = 0
    Establecer minutos = minutos + 1
    si minutos = 60
    hacer
      Establecer minutos = 0
      Establecer horas = horas + 1
  actualizar pantalla
  
```

```

para actualizar pantalla
  LCD limpiar
  LCD imprimir Columna 0 Fila 0 " Cronometro: "
  LCD imprimir Columna 0 Fila 1 " H: "
  LCD imprimir Columna 2 Fila 1 Formatear número horas 0 decimales
  LCD imprimir Columna 5 Fila 1 " M: "
  LCD imprimir Columna 7 Fila 1 Formatear número minutos 0 decimales
  LCD imprimir Columna 10 Fila 1 " S: "
  LCD imprimir Columna 12 Fila 1 Formatear número segundos 0 decimales
  
```

La versión para la cuenta atrás:  
*(iniciamos las variables horas, minutos y segundos al valor inicial deseado)*

```

Inicializar
  LCD iniciar (I2C) ArduinoBlocks LCD i2c 2x16 ADDR 0x27
  Establecer segundos = 15
  Establecer minutos = 3
  Establecer horas = 1
  actualizar pantalla

Bucle
  Ejecutar cada 1000 ms
  Establecer segundos = segundos - 1
  si segundos = -1
  hacer
    Establecer segundos = 59
    Establecer minutos = minutos - 1
    si minutos = -1
    hacer
      Establecer minutos = 59
      Establecer horas = horas - 1
      si horas = -1
      hacer
        fin
  actualizar pantalla
  
```

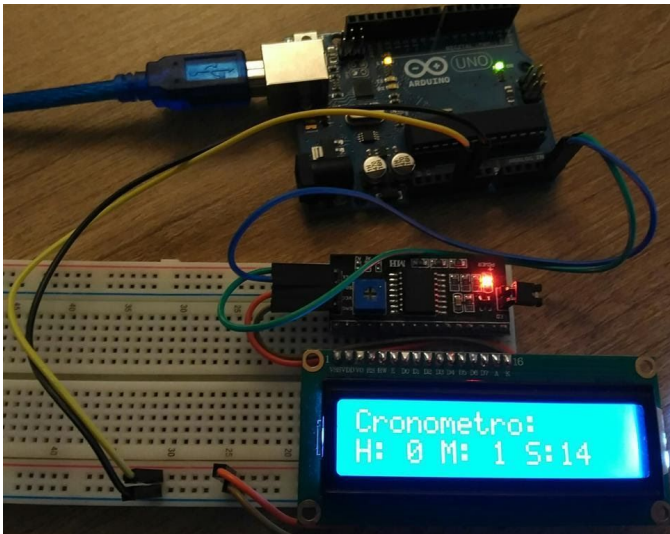


```

para actualizar pantalla
  LCD limpiar
  LCD imprimir Columna 0 Fila 0 " Cronometro: "
  LCD imprimir Columna 0 Fila 1 " H: "
  LCD imprimir Columna 2 Fila 1 Formatear número horas 0 decimales
  LCD imprimir Columna 5 Fila 1 " M: "
  LCD imprimir Columna 7 Fila 1 Formatear número minutos 0 decimales
  LCD imprimir Columna 10 Fila 1 " S: "
  LCD imprimir Columna 12 Fila 1 Formatear número segundos 0 decimales

para fin
  LCD limpiar
  LCD imprimir Columna 0 Fila 0 " FIN "
  Esperar por siempre (fin)
  
```

Vista real del montaje en funcionamiento:





## P10 - Fotómetro

Un fotómetro es un dispositivo que nos permite medir la cantidad de luz ambiente. El valor se mostrará en una pantalla LCD.

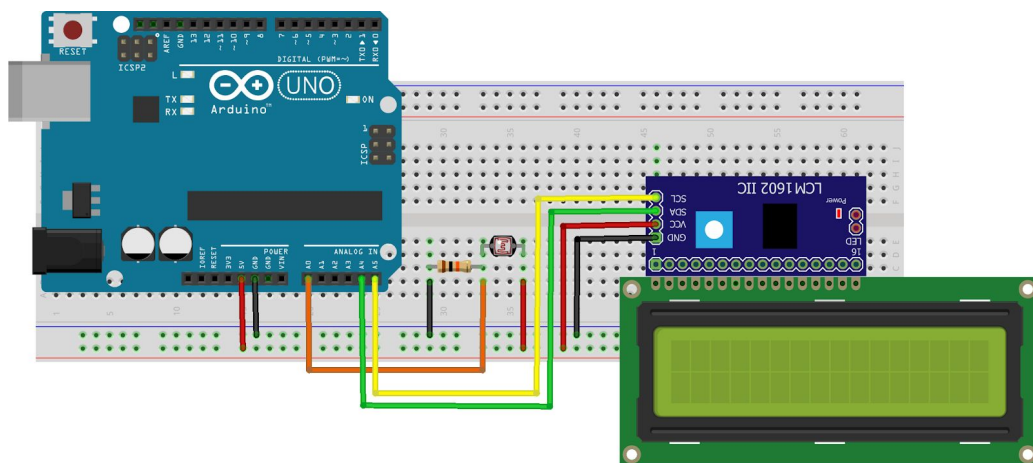
### Material necesario:

- 1 x pantalla LCD 2x16 (con módulo de conexión i2c)
- 1 x resistencia LDR 10k $\Omega$
- 1 x resistencia 10k $\Omega$
- Placa de prototipos, cables de interconexión

### Conexiones:

LDR = Pin A0

LCD (i2c) = Pin SDA (A4) , Pin SCL (A5)



Programa ArduinoBlocks:



## P11 - Iluminación crepuscular

Un sistema de iluminación crepuscular permite el encendido automático de un sistema de iluminación cuando no hay suficiente luz ambiente natural (atardecer/anochece) y de igual forma su apagado al tener la suficiente luz natural (amanecer).

Mediante un potenciómetro podremos ajustar el nivel de luz ambiente (umbral) al que queremos que se encienda o apague el sistema de iluminación.

### Material necesario:

- 1 x resistencia LDR  $k\Omega$
- 1 x resistencia  $10k\Omega$
- 1 x potenciómetro rotativo  $10k\Omega$
- 1 x led
- 1 x resistencia  $220\Omega$
- 1 x módulo de relé
- Placa de prototipos, cables de interconexión

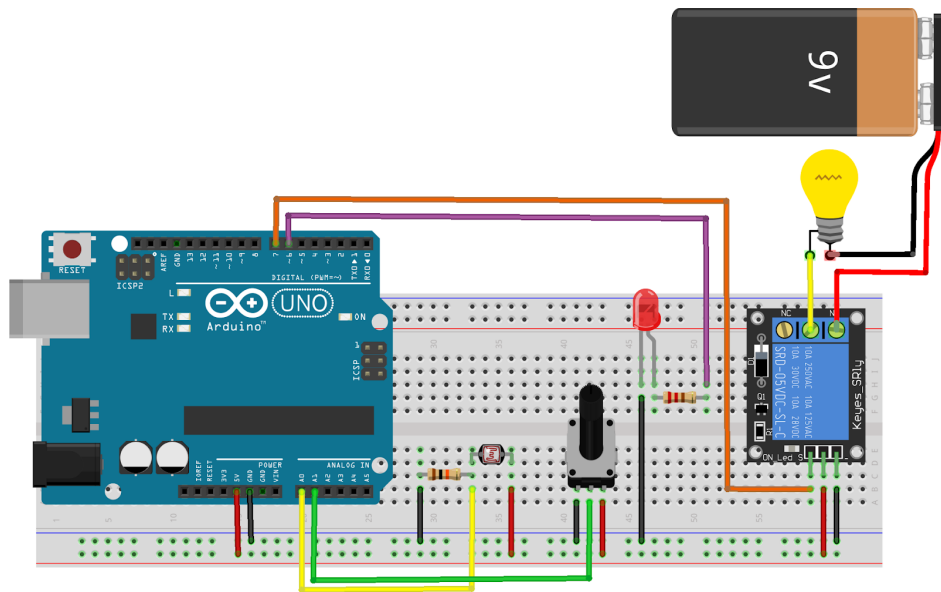
### Conexiones:

Potenciómetro = Pin A1

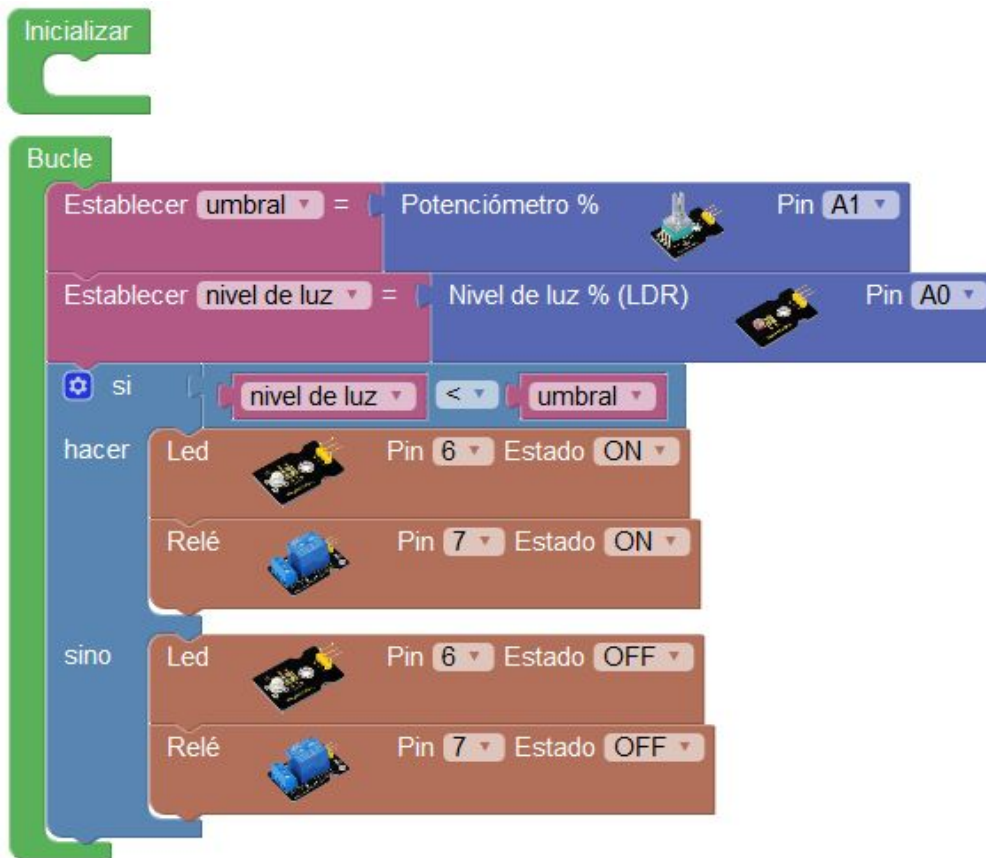
LDR = Pin A0

Led = Pin 6

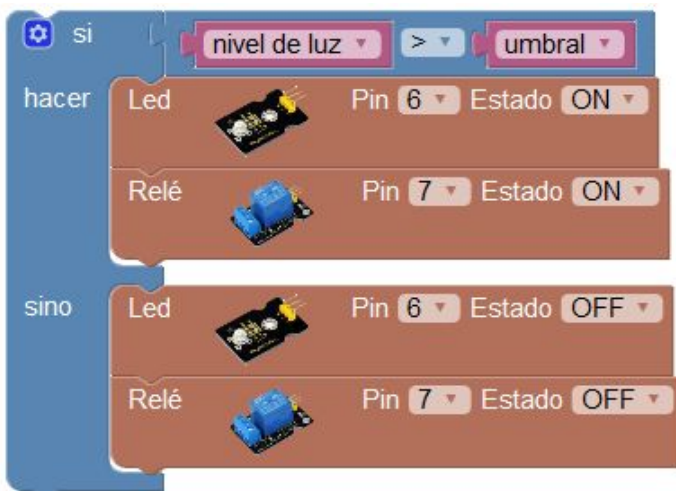
Relé = Pin 7



Programa ArduinoBlocks:



En algún caso, si conectamos la LDR de forma distinta , el valor numérico obtenido será inverso al nivel de luz ambiente (a más luz menor número) por lo que el ajuste sería al contrario:



## P12 - Encendido / Apagado con palmada

Este montaje nos permitirá sorprender a nuestros invitados en casa. Con un sonido intenso como el de una palmada podemos encender y apagar la luz de nuestra habitación.

Este sencillo sistema nos permite controlar la luz sin movernos del sofá. Como habrás comprobado no sólo sirve una palmada, cualquier sonido que supere el umbral configurado activará el sistema (la palmada nunca falla).

### Material necesario:

- 1 x módulo de sensor de sonido
- 1 x potenciómetro rotativo 10k $\Omega$
- 1 x led
- 1 x resistencia 220 $\Omega$
- 1 x módulo de relé
- Placa de prototipos, cables de interconexión

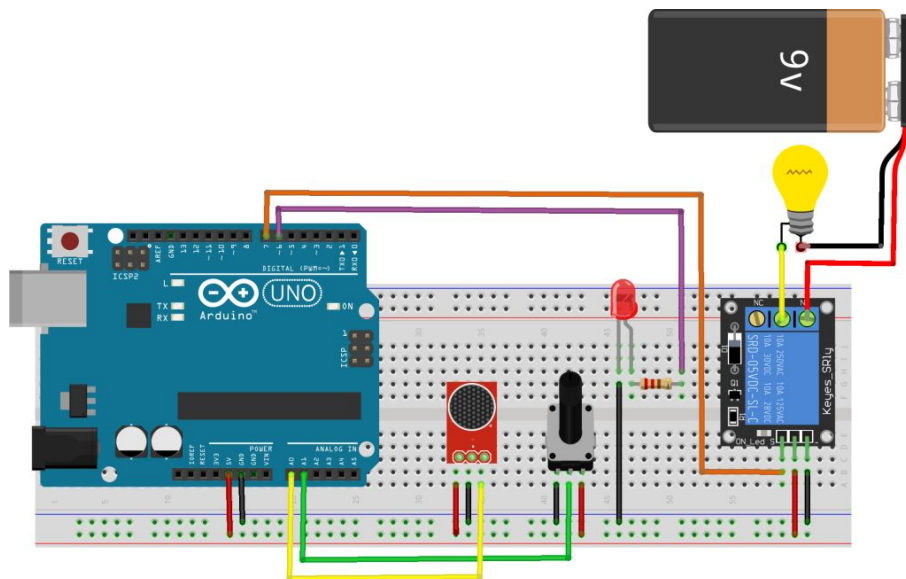
### Conexiones:

Sensor de sonido = Pin A0

Potenciómetro = Pin A1

Led = Pin 6

Relé = Pin 7



Programa ArduinoBlocks:

```

Inicializar
  Establecer luz encendida = Off

Bucle
  Establecer umbral = Potenciómetro % Pin A1
  Establecer nivel de sonido = Nivel de sonido % Pin A0

  si nivel de sonido ≥ umbral
  hacer
    si luz encendida
    hacer
      Led Pin 6 Estado OFF
      Relé Pin 7 Estado OFF
      Establecer luz encendida = Off
    sino
      Led Pin 6 Estado ON
      Relé Pin 7 Estado ON
      Establecer luz encendida = On
  Esperar 2000 milisegundos
  
```

La espera de 2000 ms es para evitar encendidos y apagados muy consecutivos. En caso de detectar una palmada (o sonido fuerte) se esperará 2000ms hasta volver a poder detectar otro nuevo sonido. Este valor se puede ajustar.

Los módulos de sensor de sonido con salida analógica normalmente tienen un potenciómetro para ajustar la sensibilidad.



## P13 - Termómetro

Construye tu propio termómetro digital con este sencillo montaje. La temperatura se visualiza cómodamente en la pantalla LCD.

El sensor de temperatura utilizado es una resistencia NTC.

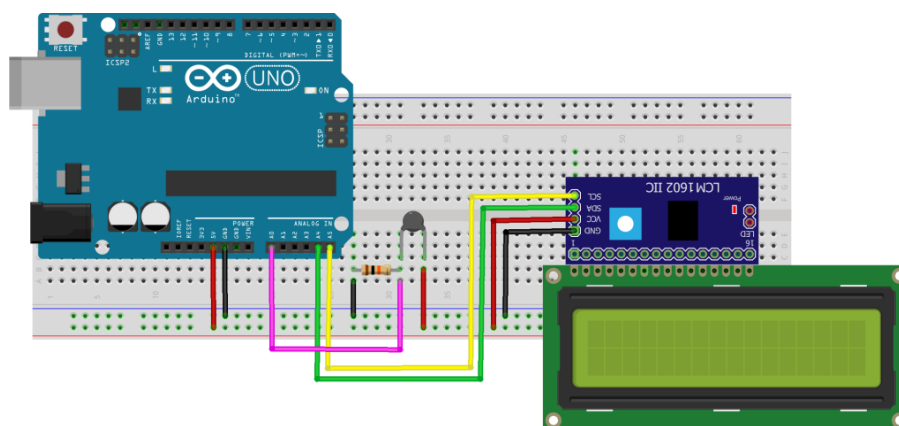
### Material necesario:

- 1 x pantalla LCD 2x16 (con módulo de conexión i2c)
- 1 x resistencia NTC 10k $\Omega$
- 1 x resistencia 10k $\Omega$
- Placa de prototipos, cables de interconexión

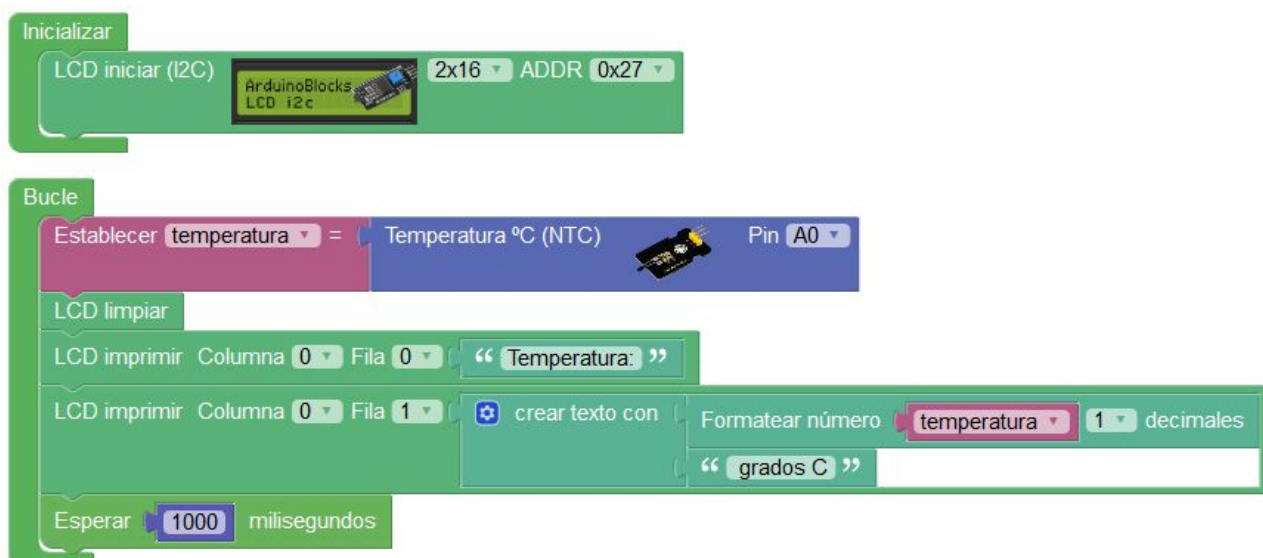
### Conexiones:

Resistencia NTC = Pin A0

LCD (i2c) = Pin SDA (A4) , Pin SCL (A5)



Programa ArduinoBlocks:





## P14 - Termostato

Un termostato permite controlar un sistema de calefacción (o de refrigeración) para actuar y conseguir la temperatura deseada en función de la temperatura ambiente.

El termostato de este montaje permite controlar un sistema de calefacción, activando la caldera (o cualquier otra fuente de calor como un radiador eléctrico) para conseguir la temperatura deseada cuando la temperatura ambiente sea inferior a la temperatura deseada (en el caso de un sistema de refrigeración sería al revés).

### Material necesario:

- 1 x pantalla LCD 2x16 (con módulo de conexión i2c)
- 1 x sensor DHT-11
- 1 x potenciómetro 10k $\Omega$
- 1 x módulo de relé
- Placa de prototipos, cables de interconexión

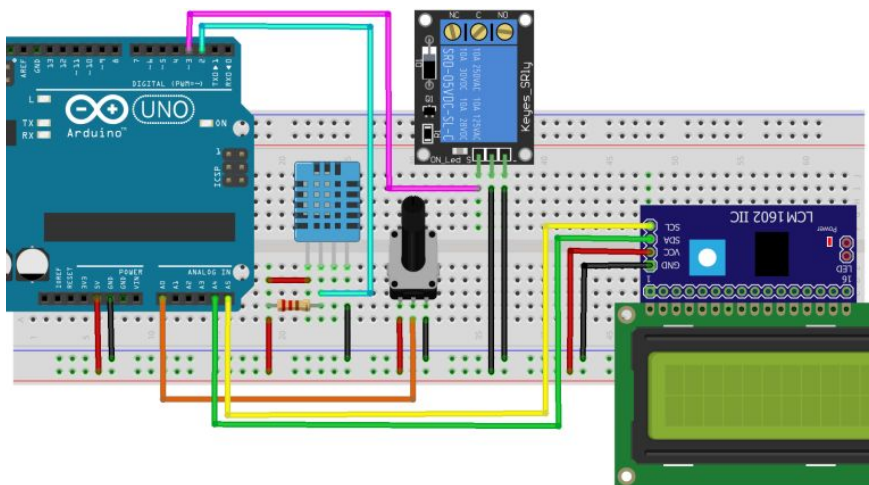
### Conexiones:

Potenciómetro = Pin A0

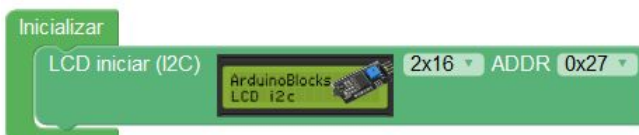
Sensor DHT-11 = Pin 2

Relé = Pin 3

LCD (i2c): Pin SDA (A4) / Pin SCL (A5)



Programa ArduinoBlocks:



```

Bucle
  Establecer potenciómetro = Potenciómetro % Pin A0
  Establecer temperatura deseada = mapear potenciómetro de 0 - 100 a 15 - 25
  Establecer temperatura = DHT-11 Temperatura °C Pin 2

  LCD limpiar
  LCD imprimir Columna 0 Fila 0 crear texto con " T Ambiente: "
  Formatear número temperatura 0 decimales
  LCD imprimir Columna 0 Fila 1 crear texto con " T Deseada: "
  Formatear número temperatura deseada 0 decimales

  si temperatura < temperatura deseada
  hacer Relé Pin 3 Estado ON
  sino Relé Pin 3 Estado OFF

  Esperar 500 milisegundos
  
```

Con el potenciómetro podemos ajustar un valor entre 15 y 25° C

Si deseamos cambiar este rango debemos modificar el mapeo al rango deseado. Por ejemplo si queremos poder ajustar entre 5 y 40 ° C:

```

Establecer temperatura deseada = mapear potenciómetro de 0 - 100 a 5 - 40
  
```

Si quisiéramos realizar un termostato para enfriar (activando un ventilador o aire acondicionado), el funcionamiento sería inverso:

```

si temperatura > temperatura deseada
hacer Relé Pin 3 Estado ON
sino Relé Pin 3 Estado OFF
  
```

## P15 - Medidor de distancia

Mediante el sensor de ultrasonidos y la pantalla LCD podemos realizar un dispositivo capaz de medir y visualizar la distancia hasta el objeto más cercano mostrando la distancia en cm.

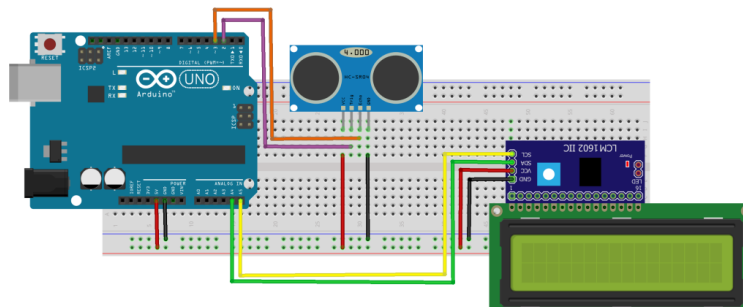
### Material necesario:

- 1 x pantalla LCD 2x16 (con módulo de conexión i2c)
- 1 x sensor de ultrasonidos HC-SR04
- Placa de prototipos, cables de interconexión

### Conexiones:

Sensor HC-SR04: Trigger = Pin 2 , Echo = Pin 3

LCD (i2c) = Pin SDA (A4) , Pin SCL (A5)



Programa ArduinoBlocks:

```
Inicializar
  LCD iniciar (I2C) 2x16 ADDR 0x27

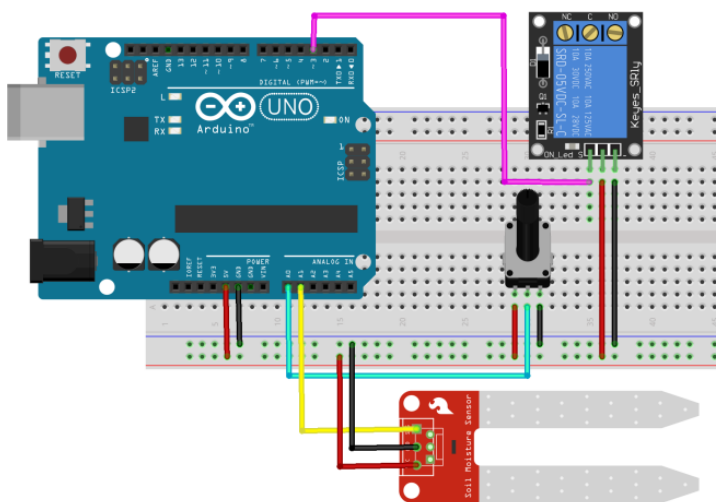
Bucle
  LCD limpiar
  LCD imprimir Columna 0 Fila 0 " Distancia: "
  Establecer distancia = Distancia (cm) [Trigger] 2 [Echo] 3
  si distancia > 0
  hacer
    LCD imprimir Columna 0 Fila 1 crear texto con distancia " cm "
  sino
    LCD imprimir Columna 0 Fila 1 "-No detectado-"
  Esperar 500 milisegundos
```

## P16 - Riego automático

Mediante el sensor de humedad detectaremos el nivel de humedad de la tierra. Si el nivel de humedad es inferior al ajustado mediante un potenciómetro se activará una electroválvula (para regar) a través de un relé. El sistema comprueba la humedad una vez por minuto.

### Material necesario:

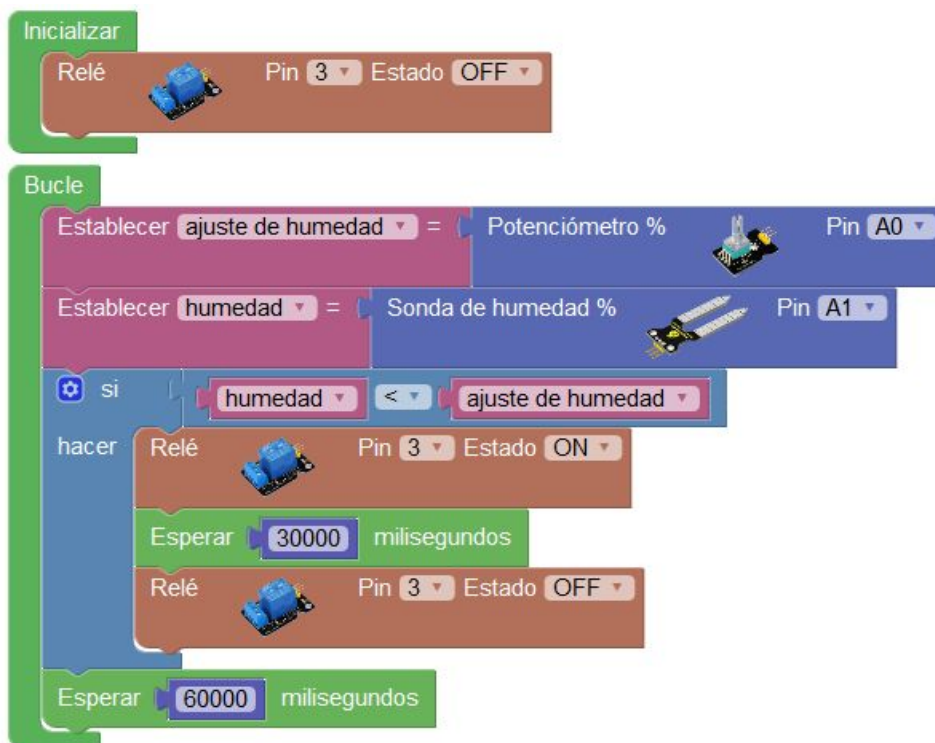
- 1 x sensor de humedad
- 1 x potenciómetro rotativo
- 1 x módulo de relé
- Placa de prototipos, cables de interconexión



### Conexiones:

Sensor humedad = Pin A1  
 Potenciómetro = Pin A0  
 Relé = Pin 3

Programa ArduinoBlocks:

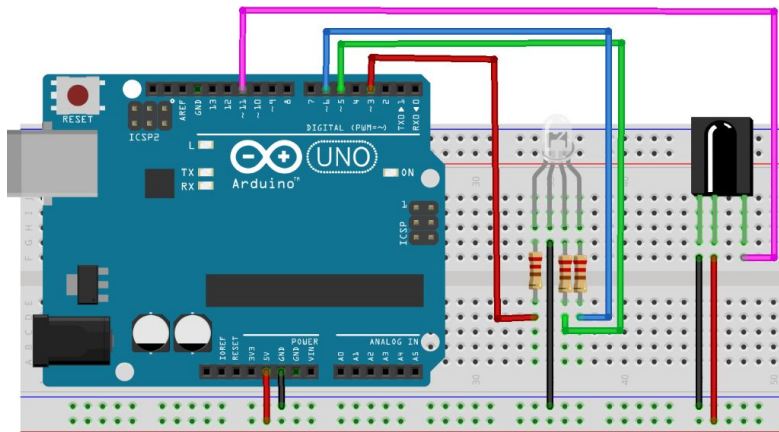


## P17 - Lámpara multicolor controlada con mando IR

Esta lámpara permitirá ajustar el ambiente a cada momento. Con la ayuda de un mando a distancia IR podremos cambiar el color y la intensidad a nuestro gusto.

### Material necesario:

- 1 x led RGB (cátodo común)
- 1 x sensor IR
- 1 x mando IR
- Placa de prototipos, cables de interconexión

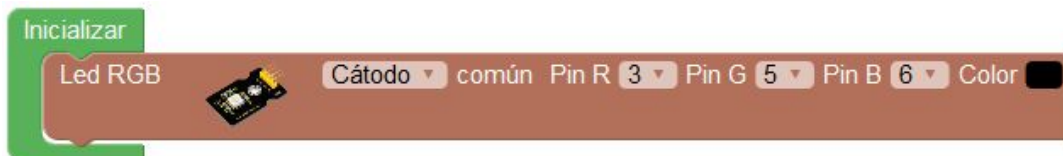


**Conexiones:**  
 Led R = Pin ~3  
 Led G = Pin ~5  
 Led B = Pin ~6  
 Sensor IR = Pin 11



Tecla "1"	16724175
Tecla "2"	16718055
Tecla "3"	16743045
Tecla "4"	16716015

Programa ArduinoBlocks:



```

Bucle
  Establecer codigo mando = Receptor de IR Pin 11
  si
    Número entero sin signo codigo mando = 16724175
  hacer
    Led RGB Cátodo común Pin R 3 Pin G 5 Pin B 6 Color rojo
  si
    Número entero sin signo codigo mando = 16718055
  hacer
    Led RGB Cátodo común Pin R 3 Pin G 5 Pin B 6 Color azul
  si
    Número entero sin signo codigo mando = 16743045
  hacer
    Led RGB Cátodo común Pin R 3 Pin G 5 Pin B 6 Color verde
  si
    Número entero sin signo codigo mando = 16716015
  hacer
    Led RGB Cátodo común Pin R 3 Pin G 5 Pin B 6 Color negro
  
```

Antes de realizar este montaje es recomendable obtener los códigos para cada botón del mando a distancia utilizado. Esto se puede realizar fácilmente con este programa para obtener los códigos de cada botón por la consola serie:

```

Inicializar
  Enviar " IR codes 1.0 " Salto de línea

Bucle
  Establecer ir code = Receptor de IR Pin 11
  si
    ir code ≠ 0
  hacer
    Enviar crear texto con " Rx code: " Salto de línea
    Número entero sin signo ir code
  
```



## P18 - Piano con teclado

Un sencillo piano digital para poder tocar y componer nuestras propias melodías. Cada tecla del keypad reproducirá un tono en el zumbador conectado.

### Material necesario:

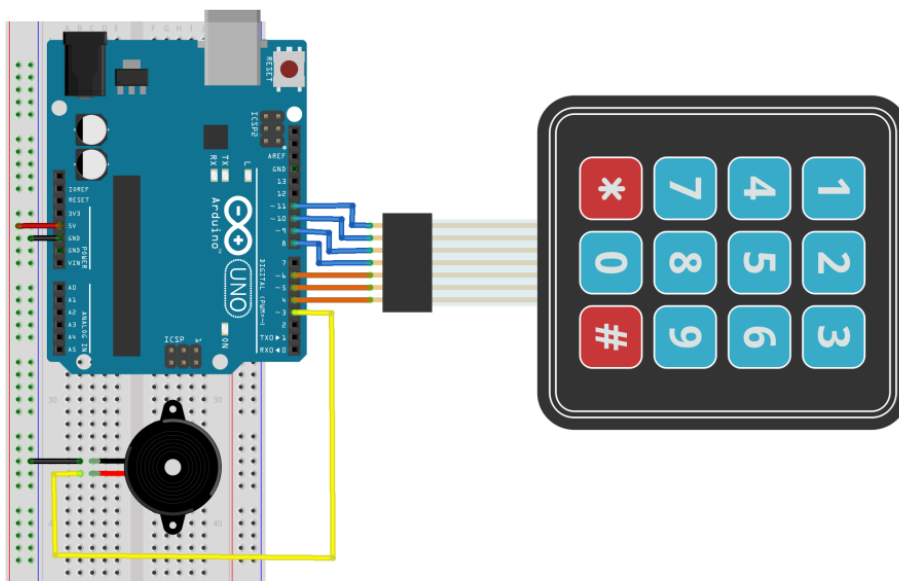
- 1 x zumbador pasivo
- 1 x keypad 3x4
- Placa de prototipos, cables de interconexión

### Conexiones:

Keypad: Fila 1 = Pin 11, Fila 2 = Pin 10, Fila 3 = Pin 9, Fila 4 = Pin 8

Keypad: Col-1 = Pin 6, Col-2 = Pin 5, Col-3 = Pin 4

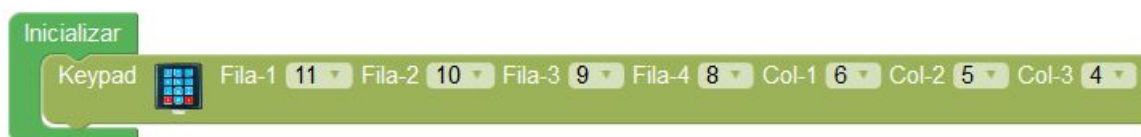
Zumbador = Pin ~3

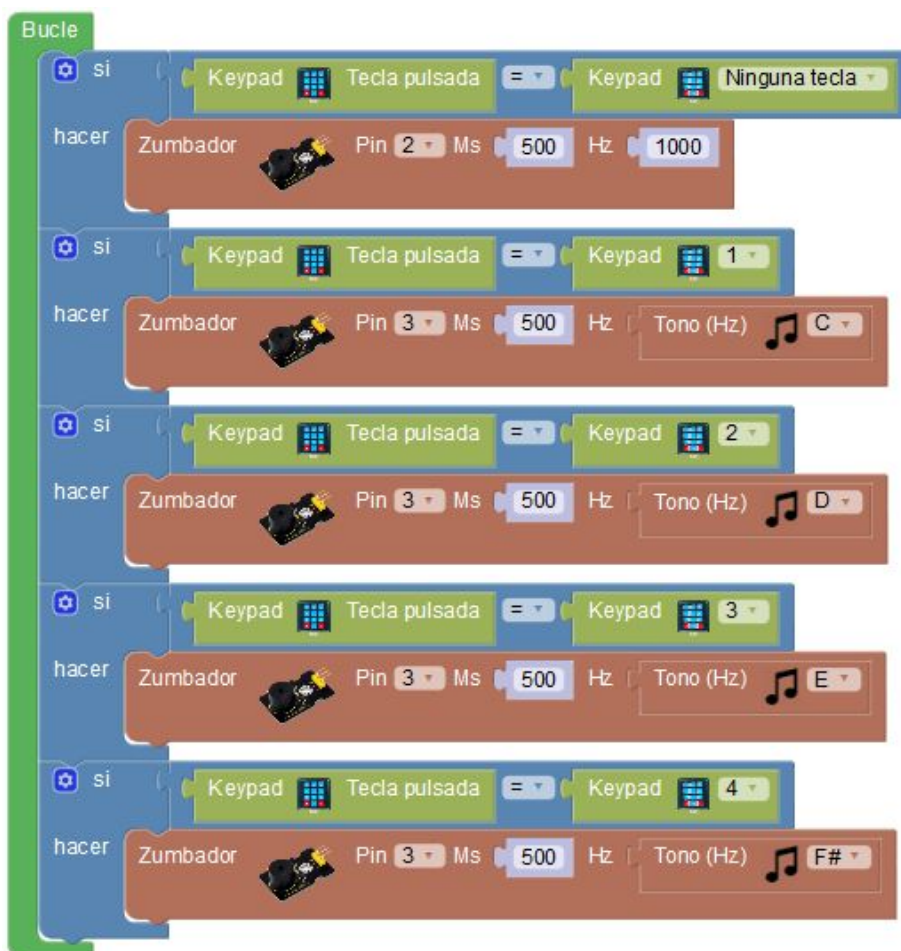


Ejemplos de zumbadores pasivo para utilizar en este proyecto.



Programa ArduinoBlocks:



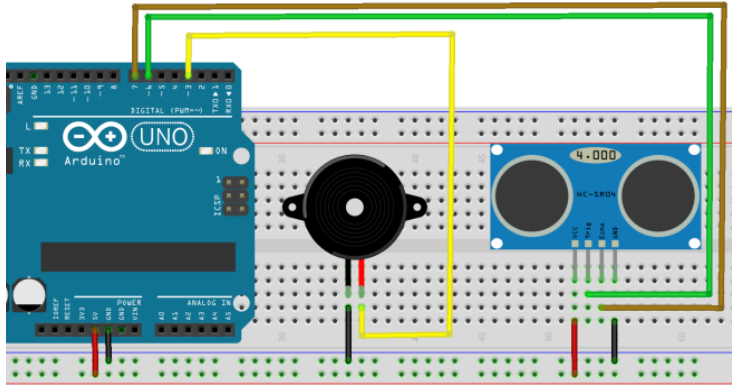


## P19 - Sensor de aparcamiento

Hoy en día todos los coches modernos incorporan un sensor de aparcamiento. Este sensor nos permite detectar los objetos que se encuentran delante y detrás del vehículo para evitar colisionar. De una forma sencilla podemos crear nuestro propio sensor de aparcamiento.

### Material necesario:

- 1 x zumbador
- 1 x sensor de ultrasonidos HC-SR04
- Placa de prototipos, cables de interconexión



### Conexiones:

Sensor HC-SR04:

Trigger = Pin 6

Echo = Pin 7

Zumbador = Pin ~3

Programa ArduinoBlocks:



## P20 - Control Tilt/Pan on joystick

El movimiento conocido como pan/tilt (horizontal/vertical) permite controlar la posición en dos ejes. Este tipo de controles se utiliza comúnmente para mover cámaras de seguridad, detectores de obstáculos, etc.

Existen pequeños mecanismo que implementan la función de pan/tilt gracias a la integración de dos servos.



### Material necesario:

- 2 x micro-servos + mecanismo pan/tilt
- 1 x módulo joystick
- Placa de prototipos, cables de interconexión

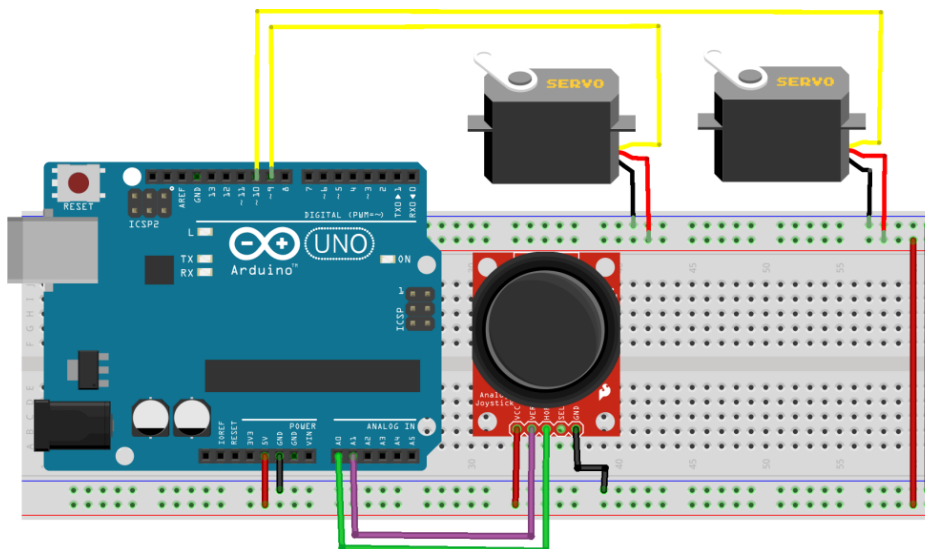
### Conexiones:

Servo pan = Pin ~9

Servo tilt = Pin ~10

Joystick X = Pin A0

Joystick Y = Pin A1



Programa ArduinoBlocks:

```

Inicializar
  Establecer pan = 90
  Establecer tilt = 90
  Establecer retardo velocidad = 50

Bucle
  leer joystick
  Servo Pin 9 Grados pan Retardo (ms) 0
  Servo Pin 10 Grados tilt Retardo (ms) 0
  Esperar retardo velocidad milisegundos

para leer joystick
  si Posición del Joystick % Pin A0 X < 25
  hacer Establecer pan = pan - 1
  si Posición del Joystick % Pin A0 X > 75
  hacer Establecer pan = pan + 1
  si Posición del Joystick % Pin A1 Y < 25
  hacer Establecer tilt = tilt - 1
  si Posición del Joystick % Pin A1 Y > 75
  hacer Establecer tilt = tilt + 1
  Establecer pan = limitar pan entre 0 y 180
  Establecer tilt = limitar tilt entre 0 y 180
  
```



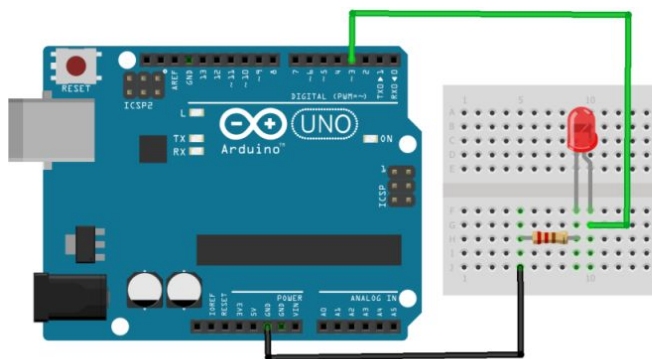
## P21 - Control de un LED desde PC (consola)

Vamos a realizar el control de la iluminación de un led controlando su intensidad desde un PC (desde el monitor serie del PC)

La idea es sencilla, recibimos un valor a través del terminal serie entre Arduino y el PC y cambiamos la intensidad del led al valor recibido (recuerda que como es un valor para la salida PWM debe ser un valor entre 0 a 255)

### Material necesario:

- 1 x led
- 1 x resistencia 220Ω
- Placa de prototipos, cables de interconexión



Conexiones:  
Led = Pin ~3

Programa ArduinoBlocks:

```

Inicializar
  Iniciar Baudios 9600
  Enviar "Control de iluminación 1.0" Salto de línea
  Enviar "Envía un valor entre 0 y 255 para regular el led" Salto de línea

Bucle
  si ¿Datos recibidos?
  hacer
    Establecer intensidad = Recibir como número Hasta salto de línea
    Establecer intensidad = limitar intensidad entre 0 y 255
    Led intensidad (PWM) Pin 3 Valor intensidad
    Enviar crear texto con "Intensidad = " Salto de línea
    intensidad
  
```



*Ejemplo de control desde la consola de ArduinoBlocks:*

### ArduinoBlocks :: Consola serie

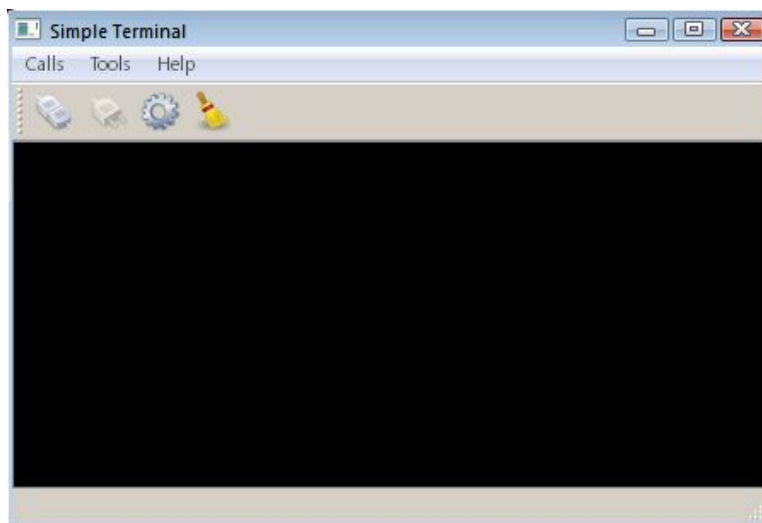
Baudrate: 9600

125

Control de iluminacion 1.0

Envia un valor entre 0 y 255 para regular el led  
Intensidad = 125.00

*Podemos utilizar otras aplicaciones de consola serie:*



## P22 - Control de relés por Bluetooth

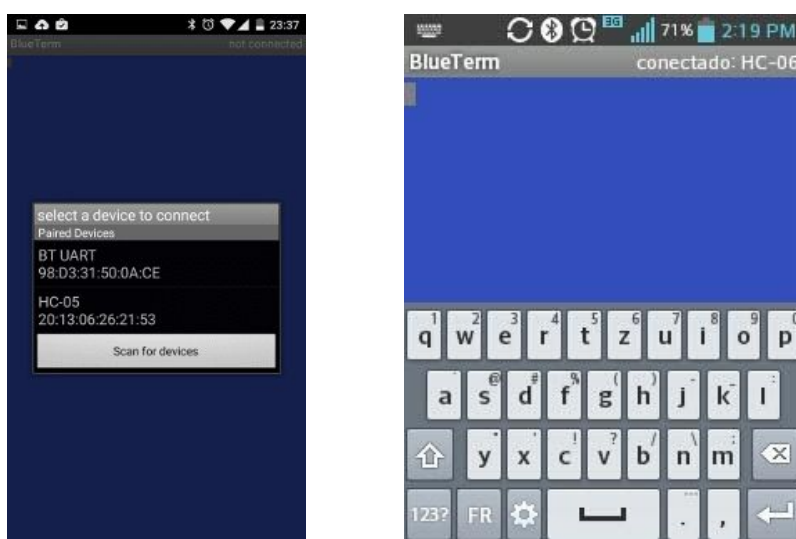
La comunicación inalámbrica Bluetooth apareció en los dispositivos móviles hace varios años y nos permite de una forma sencilla y rápida transferir información entre dispositivos a una distancia de hasta 100m.

Existen diferentes módulos para Arduino que nos permiten la utilización de la conexión Bluetooth, ArduinoBlocks es compatible con el módulo HC-06.

*(repasa la utilización del módulo Bluetooth en el apartado 3.3.4)*

Para el envío y recepción de datos utilizaremos una consola serie bluetooth desde algún dispositivo móvil como un Smartphone o Tablet.

*Ejemplo: Aplicación "BlueTerm" para dispositivos Android*



<https://play.google.com/store/apps/details?id=es.pymasde.blueterm&hl=es>

### Material necesario:

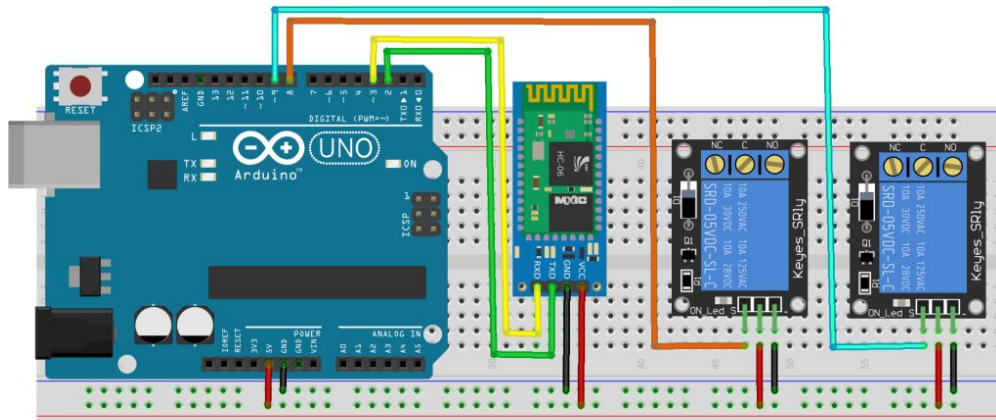
- 1 x módulo Bluetooth HC-06
- 2 x módulo relé
- Dispositivo móvil con conexión Bluetooth
- Placa de prototipos, cables de interconexión

### Conexiones:

Bluetooth RX = Pin 2, TX = Pin 3

Relé 1 = Pin 8

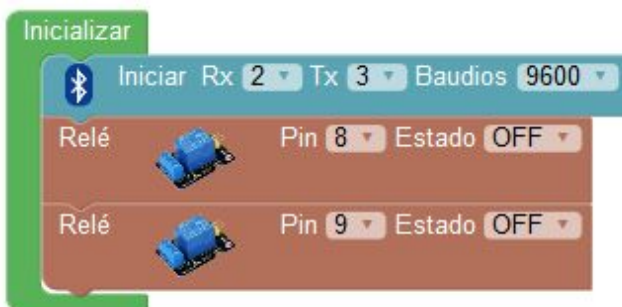
Relé 1 = Pin 9



Para el control por Bluetooth vamos a implementar un protocolo de comunicación muy sencillo donde cada comando es un número que realizará una función:

<b>Comando (número)</b>	<b>Función</b>
1	Relé 1 = ON
2	Relé 1 = OFF
3	Relé 2 = ON
4	Relé 2 = OFF

Programa ArduinoBlocks:



```
Bucle
si ¿Datos recibidos?
hacer Establecer comando = Recibir como número Hasta salto de línea
si comando = 1
hacer Relé Pin 8 Estado ON
si comando = 2
hacer Relé Pin 8 Estado OFF
si comando = 3
hacer Relé Pin 9 Estado ON
si comando = 4
hacer Relé Pin 9 Estado OFF
```

Módulo de relés utilizado en el proyecto:



## P23 - Estación meteorológica Bluetooth

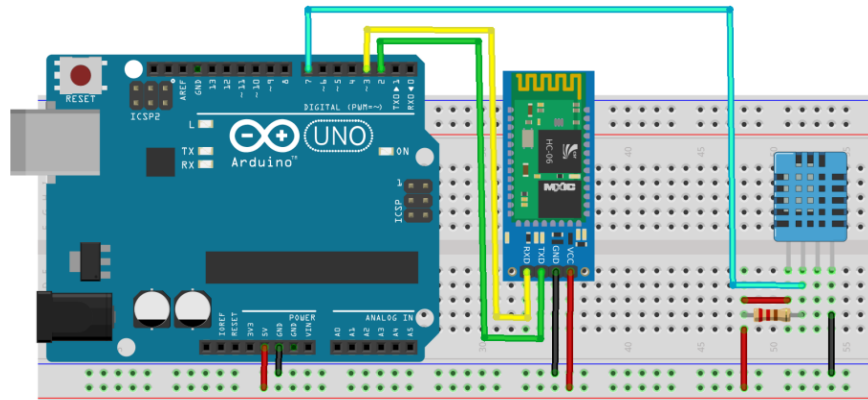
En proyectos anteriores hemos visto como obtener la temperatura y la humedad fácilmente con el sensor DHT-11. Con este proyecto vamos a aplicar esta idea pero pudiendo monitorizar los datos de temperatura y humedad remotamente, así podemos tener la central meteorológica en un lugar alejado, como por ejemplo en la terraza de casa, y los datos los podemos visualizar cómodamente en el interior en un dispositivo móvil con conexión Bluetooth.

### Material necesario:

- 1 x sensor DHT-11
- 1 x módulo Bluetooth HC-06
- 1 x dispositivo móvil con conexión Bluetooth
- Placa de prototipos, cables de interconexión

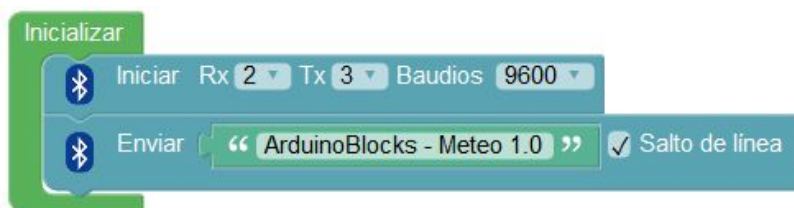
### Conexiones:

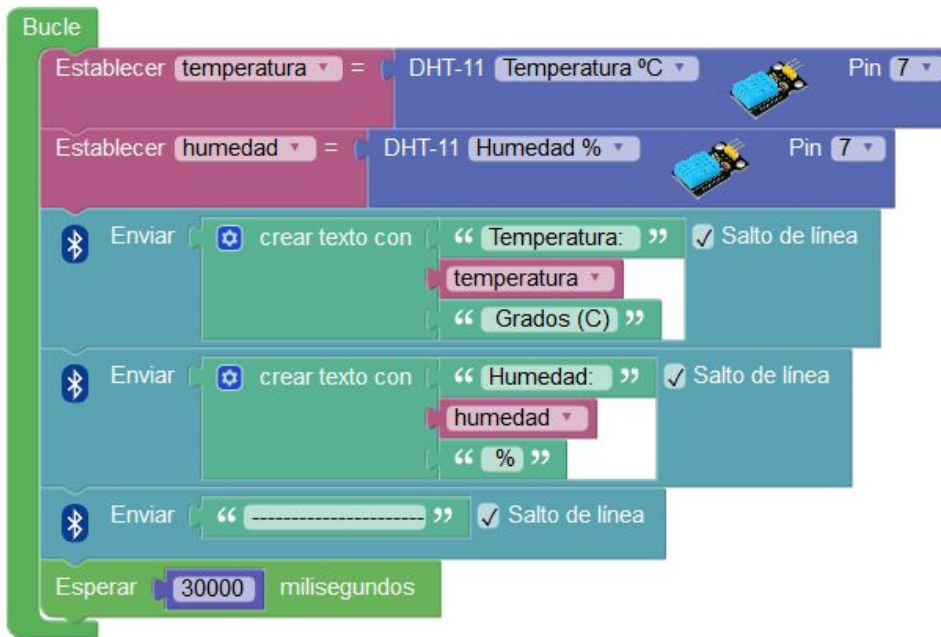
Bluetooth RX = Pin 2 / TX = Pin 3  
Sensor DHT-11 = Pin 7



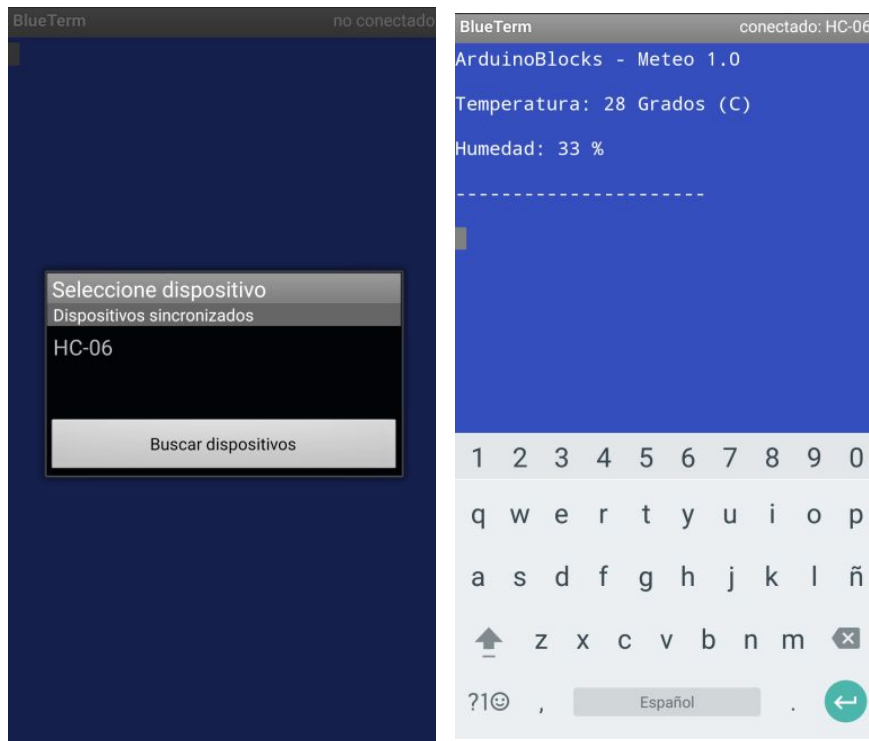
El programa es sencillo, cada 30 segundos enviamos a través de la conexión Bluetooth los datos de temperatura y humedad. En una aplicación tipo "BlueTerm" de Android podemos recibir y visualizar los datos en tiempo real.

Programa ArduinoBlocks:





Visualización desde la aplicación BlueTerm en Android:





## P24 - Control de LED por voz (Android+Bluetooth)

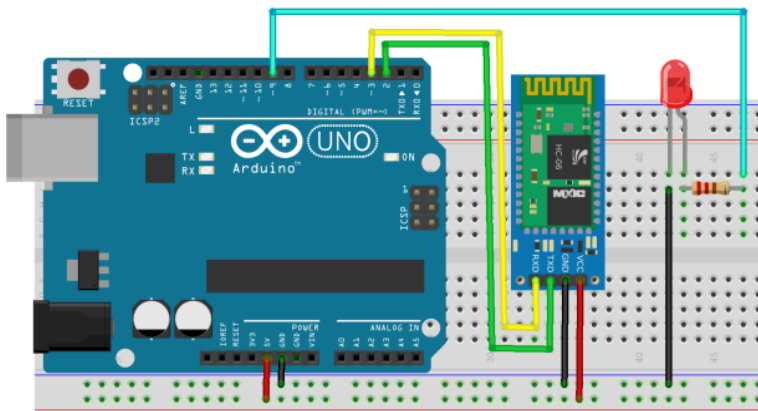
El terminal Bluetooth es muy útil y fácil de utilizar pero en algunas ocasiones necesitamos crear nuestra propia interfaz de control en el dispositivo. Para crear aplicaciones fácilmente en Android disponemos de la extraordinaria herramienta *AppInventor*. Esta plataforma nos permite crear aplicaciones Android de forma visual y programarla con lenguaje de bloques. Lo único que necesitamos es una cuenta de Google para poder utilizarla.

El siguiente proyecto utiliza una aplicación muy sencilla creada en *AppInventor* que reconoce la voz y enviará un comando u otro a través de Bluetooth para encender o apagar un led.

Comando de voz	Comando enviado	Acción de Arduino
“encender”	1	Encender el led
“apagar”	2	Apagar el led
“parpadear”	3	Parpadea 3 segundos

### Material necesario:

- 1 x led
- 1 x resistencia 220Ω
- 1 x módulo Bluetooth HC-06
- 1 x dispositivo móvil con sistema Android
- Placa de prototipos, cables de interconexión



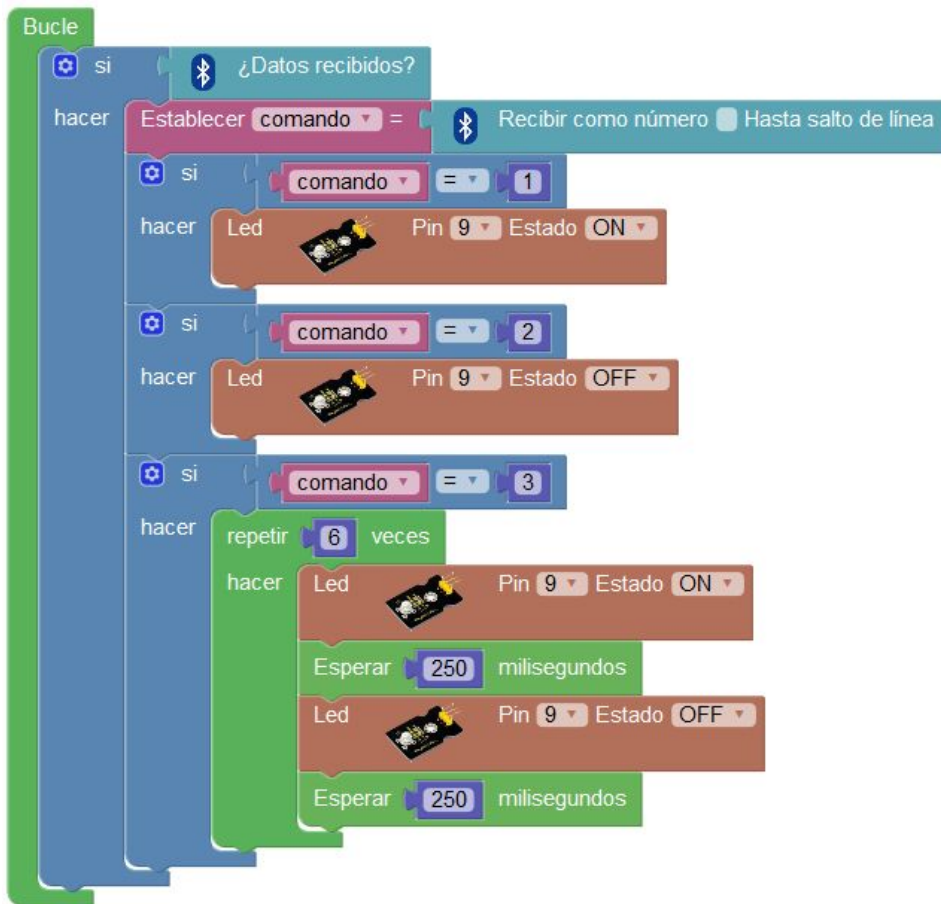
### Conexiones:

Bluetooth:  
RX = Pin 2  
TX = Pin 3

Led = Pin ~9

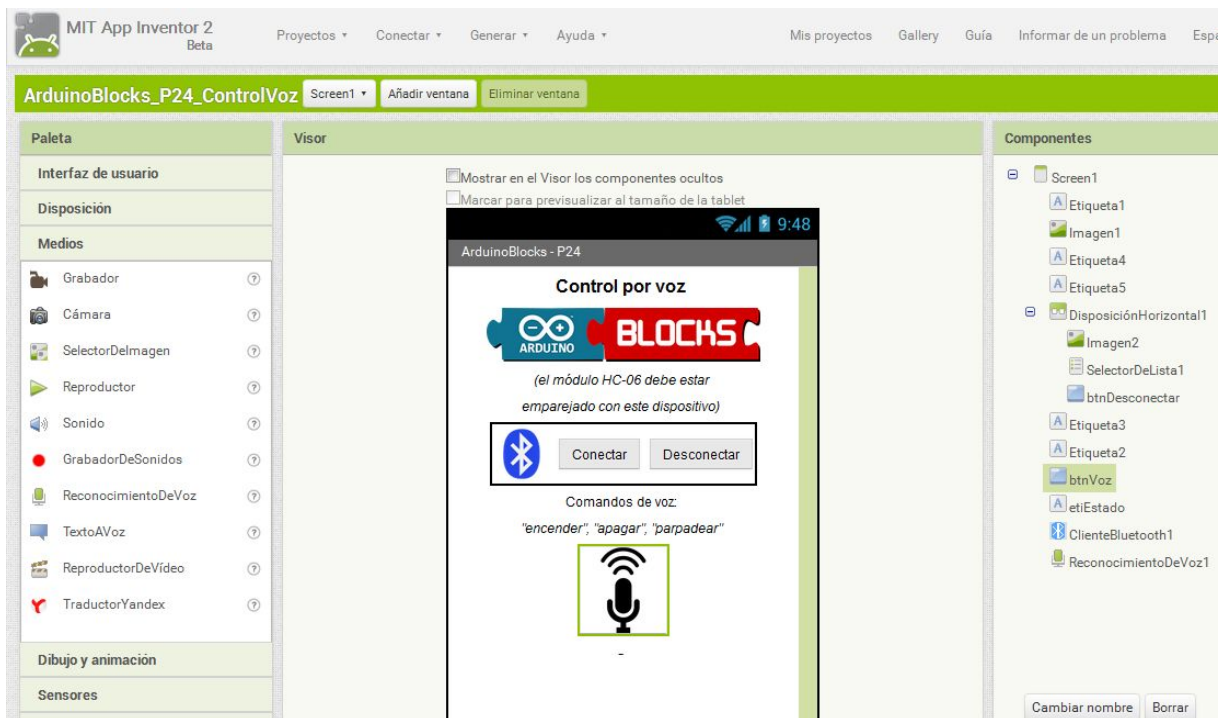
### Programa ArduinoBlocks:





*(no se ha marcado la opción "Hasta salto de línea" en la recepción Bluetooth porque desde la aplicación Android no enviamos salto de línea)*

Diseño de la interfaz de la aplicación Android con *AppInventor*:



Código de la aplicación Android con *AppInventor*:

```

cuando SelectorDeLista1 .AntesDeSelección
ejecutar poner SelectorDeLista1 . Elementos como ClienteBluetooth1 . DireccionesYNombres

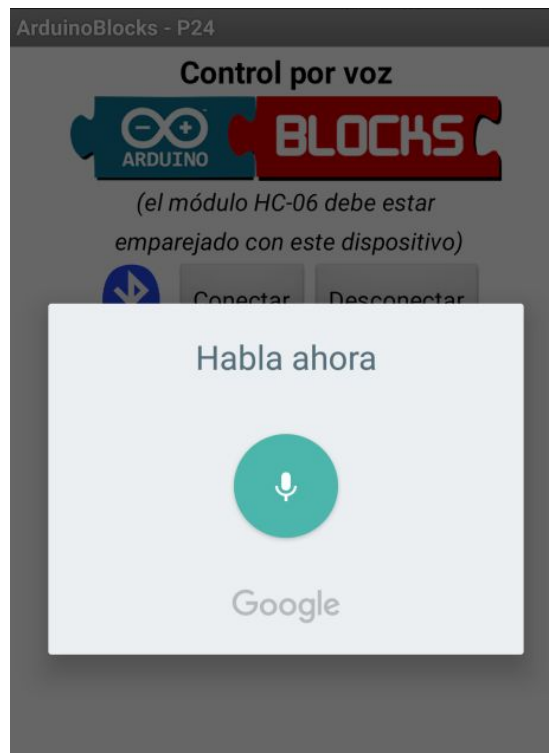
cuando SelectorDeLista1 .DespuésDeSelección
ejecutar si
    llamar ClienteBluetooth1 .Conectar
        dirección SelectorDeLista1 . Selección
    entonces poner etiEstado . Texto como " Conectado! "
    si no poner etiEstado . Texto como " Error conectando! "

cuando btnDesconectar .Clic
ejecutar llamar ClienteBluetooth1 .Desconectar
    poner etiEstado . Texto como " Desconectado "

cuando btnVoz .Clic
ejecutar si
    entonces llamar ReconocimientoDeVoz1 .ObtenerTexto

cuando ReconocimientoDeVoz1 .DespuésDeObtenerTexto
Resultado
ejecutar si
    entonces poner etiEstado . Texto como unir " Has dicho: "
        ReconocimientoDeVoz1 . Resultado
        si
            ReconocimientoDeVoz1 . Resultado = " encender "
            entonces llamar ClienteBluetooth1 .EnviarTexto
                texto " 1 "
            si
            ReconocimientoDeVoz1 . Resultado = " apagar "
            entonces llamar ClienteBluetooth1 .EnviarTexto
                texto " 2 "
            si
            ReconocimientoDeVoz1 . Resultado = " parpadear "
            entonces llamar ClienteBluetooth1 .EnviarTexto
                texto " 3 "
        si no poner etiEstado . Texto como " No estás conectado! "
    
```

Aplicación ApplInventor funcionando:



## P25 - Control domótico (Android+Bluetooth)

Con todo lo aprendido en los proyectos anteriores vamos a implementar un proyecto un poco más complejo implementando un control domótico para casa.

La domótica es la aplicación de la automatización y la robótica en el hogar. La domótica debe cumplir funciones de confort, seguridad y ahorro energético.

Cada día aparecen más sistemas domóticos pero gracias a Arduino podemos crearnos nuestros sistemas propios de automatización y programar el funcionamiento de la aplicación según nuestras necesidades.

Este proyecto implementa un sistema muy sencillo de automatización, pero nos sirve como una aproximación para entender cómo funcionan estos sistemas.

El control domótico con Arduino y Android incluirá:

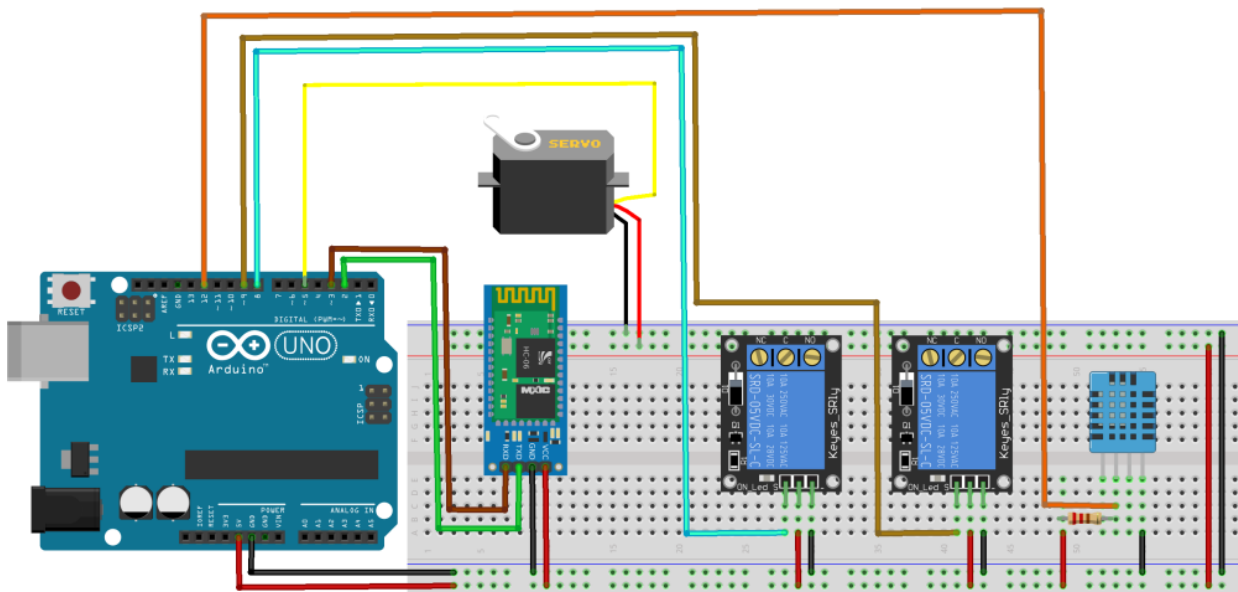
- Control de dos relés para iluminación ON/OFF
- Monitorización de temperatura desde dispositivo móvil
- Control de una persiana (simulada con un servo)
- Escenas (confort, apagar todo, simular presencia)
- Control desde dispositivo móvil Android vía Bluetooth

### Material necesario:

- 1 x módulo Bluetooth HC-06
- 1 x sensor DHT-11
- 1 x servomotor
- 2 x módulo relé
- Placa de prototipos
- Cables de interconexión

### Conexiones:

- Bluetooth RX = Pin 2
- Bluetooth TX = Pin 3
- Sensor DHT-11 = Pin 12
- Relé 1 = Pin 8
- Relé 2 = Pin 9
- Servo = Pin ~5





Programa ArduinoBlocks:

```

Inicializar
  Iniciar Rx 2 Tx 3 Baudios 9600
  Establecer ultimo envio = 0

Bucle
  si ¿Datos recibidos?
  hacer
    Establecer comando = Recibir como número Hasta salto de línea
    ejecutar comando
  Establecer diferencia = Tiempo transcurrido (milisegundos) - ultimo envio
  si diferencia ≥ 5000
  hacer
    Establecer ultimo envio = Tiempo transcurrido (milisegundos)
    Enviar DHT-11 Temperatura °C Pin 12 Salto de línea

para escena_comfort
  Relé Pin 8 Estado ON
  Relé Pin 9 Estado OFF
  Servo Pin 5 Grados Ángulo 90° Retardo (ms) 250

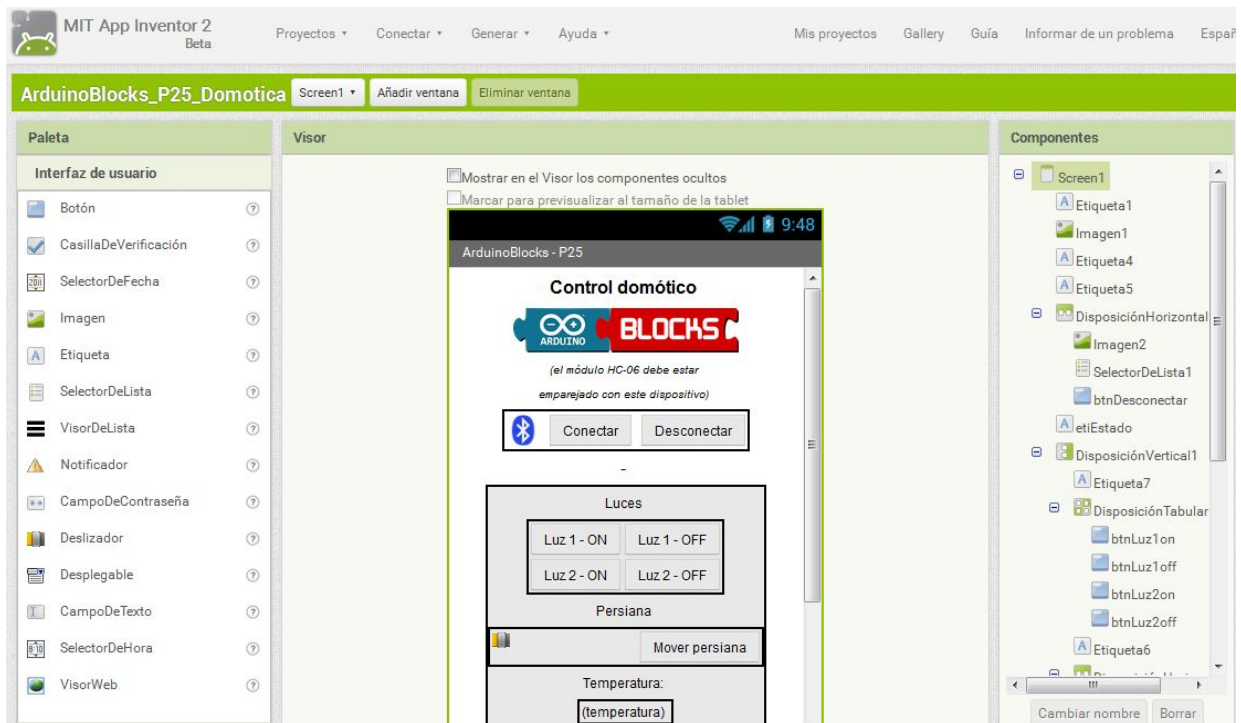
para escena_simulacion
  repetir 5 veces
  hacer
    Relé Pin 8 Estado ON
    Relé Pin 9 Estado OFF
    Esperar 500 milisegundos
    Relé Pin 8 Estado OFF
    Relé Pin 9 Estado ON
    Esperar 500 milisegundos
    Servo Pin 5 Grados entero aleatorio de 0 a 180 Retardo (ms) 100
    Esperar 500 milisegundos
  
```



```
para ejecutar comando
  si comando = 1
  hacer Relé Pin 8 Estado ON
  si comando = 2
  hacer Relé Pin 8 Estado OFF
  si comando = 3
  hacer Relé Pin 9 Estado ON
  si comando = 4
  hacer Relé Pin 9 Estado OFF
  si comando = 11
  hacer escena_off
  si comando = 12
  hacer escena_comfort
  si comando = 13
  hacer escena_simulacion
  si comando ≥ 100
  hacer establecer posicion persiana a mapear comando de 100 - 200 a 0 - 180
  Servo Pin 5 Grados posicion persiana Retardo (ms) 250
```

```
para escena_off
  Relé Pin 8 Estado OFF
  Relé Pin 9 Estado OFF
  Servo Pin 5 Grados Ángulo 0° Retardo (ms) 250
```

## Diseño de la interfaz de la aplicación *Android* con *AppInventor*:



## Código de la aplicación *Android* con *AppInventor*:

```

cuando SelectorDeLista1 . AntesDeSelección
ejecutar poner SelectorDeLista1 . Elementos como ClienteBluetooth1 . DireccionesYNombres

cuando SelectorDeLista1 . DespuésDeSelección
ejecutar si
  llamar ClienteBluetooth1 . Conectar
  dirección SelectorDeLista1 . Selección
  entonces poner etiEstado . Texto como " Conectado! "
  si no poner etiEstado . Texto como " Error conectando! "

cuando btnDesconectar . Clic
ejecutar llamar ClienteBluetooth1 . Desconectar
  poner etiEstado . Texto como " Desconectado "

cuando Reloj1 . Temporizador
ejecutar si
  entonces si
    llamar ClienteBluetooth1 . BytesDisponiblesParaRecibir > 0
    entonces poner etiTemp . Texto como llamar ClienteBluetooth1 . RecibirTexto
      númeroDeBytes llamar ClienteBluetooth1 . BytesDisponib
  
```

cuando **btnLuz1on** .Clic  
ejecutar si **ClienteBluetooth1** . Conectado  
entonces llamar **ClienteBluetooth1** .EnviarTexto  
texto " 1 "

cuando **btnLuz1off** .Clic  
ejecutar si **ClienteBluetooth1** . Conectado  
entonces llamar **ClienteBluetooth1** .EnviarTexto  
texto " 2 "

cuando **btnLuz2on** .Clic  
ejecutar si **ClienteBluetooth1** . Conectado  
entonces llamar **ClienteBluetooth1** .EnviarTexto  
texto " 3 "

cuando **btnLuz2off** .Clic  
ejecutar si **ClienteBluetooth1** . Conectado  
entonces llamar **ClienteBluetooth1** .EnviarTexto  
texto " 4 "

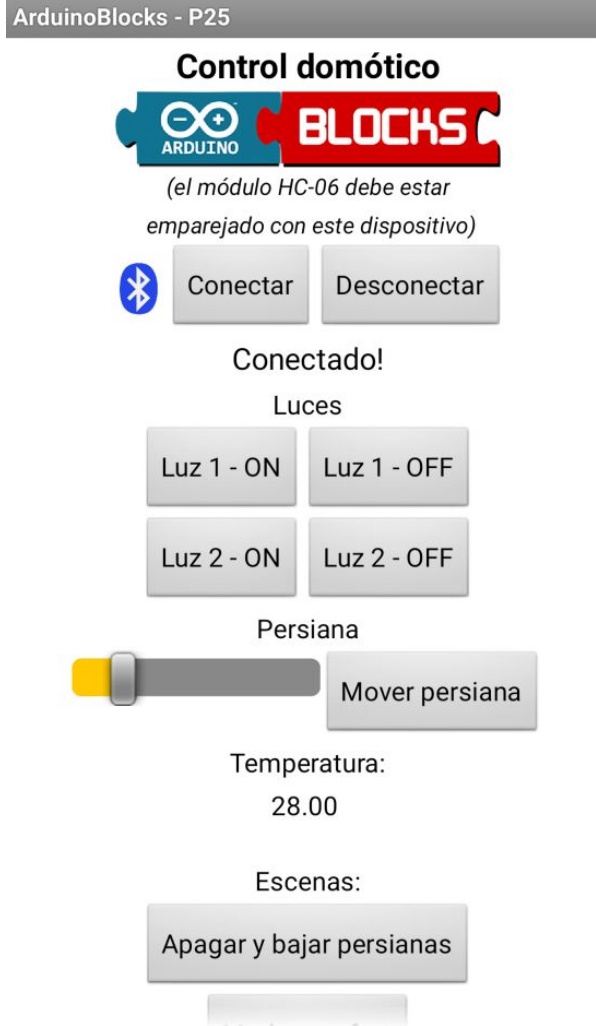
cuando **btnEscenaOff** .Clic  
ejecutar si **ClienteBluetooth1** . Conectado  
entonces llamar **ClienteBluetooth1** .EnviarTexto  
texto " 11 "

cuando **btnEscenaComfort** .Clic  
ejecutar si **ClienteBluetooth1** . Conectado  
entonces llamar **ClienteBluetooth1** .EnviarTexto  
texto " 12 "

cuando **btnEscenaSimulacion** .Clic  
ejecutar si **ClienteBluetooth1** . Conectado  
entonces llamar **ClienteBluetooth1** .EnviarTexto  
texto " 13 "

cuando **btnPersiana** .Clic  
ejecutar si **ClienteBluetooth1** . Conectado  
entonces llamar **ClienteBluetooth1** .EnviarTexto  
texto **persianaPos** . PosiciónDelPulgar

Aspecto final de la aplicación de control y monitorización:



## P26 - GPS con visualización en LCD

El siguiente proyecto nos permite visualizar la información de obtenida desde el módulo GPS en una pantalla LCD. Este proyecto nos permitiría por ejemplo añadir un indicador de velocidad, altitud, posición, etc. a nuestra bicicleta o motocicleta.

Cada 5s se mostrarán unos datos diferentes en la pantalla:

- Pantalla 1: Información de latitud y longitud
- Pantalla 2: Velocidad y altitud
- Pantalla 3: Fecha y hora recibida del satélite GPS

Para un correcto funcionamiento el módulo GPS debe estar preferiblemente en un espacio a cielo abierto y puede tardar unos minutos en obtener información válida.

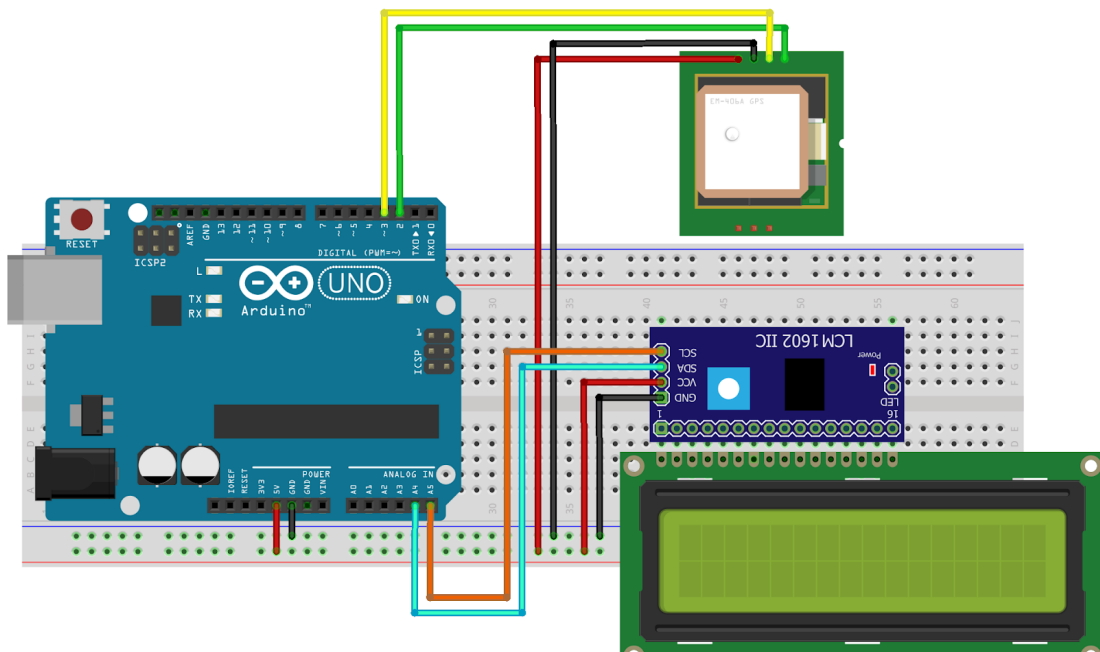
### Material necesario:

- 1 x módulo GPS
- 1 x LCD con conexión I2C
- 1 x Arduino UNO
- Placa de prototipos
- Cables de interconexión

### Conexiones:

- GPS RX = Pin 2
- GPS TX = Pin 3
- LCD = I2C

Esquema de conexión:





Programa ArduinoBlocks:

**Inicializar**

- LCD iniciar (I2C) **ArduinoBlocks LCD i2c** **2x16** ADDR **0x27**
- GPS iniciar Rx **2** Tx **3**
- Establecer **pantalla** = **0**

**Bucle**

Ejecutar cada **5000** ms

LCD limpiar

si **¿Datos válidos?**

hacer

- si **pantalla** = **0**
- hacer **info1**
- Establecer **pantalla** = **1**
- sino si **pantalla** = **1**
- hacer **info2**
- Establecer **pantalla** = **2**
- sino si **pantalla** = **2**
- hacer **info3**
- Establecer **pantalla** = **0**

sino **LCD imprimir** Columna **0** Fila **0** **“ Sin datos GPS ”**

**para info1**

LCD imprimir Columna **0** Fila **0** **“ Lat: ”**

crear texto con **“ Lat: ”**

Formatear número **Posición Latitud** **4** decimales

LCD imprimir Columna **0** Fila **1** **“ Lon: ”**

crear texto con **“ Lon: ”**

Formatear número **Posición Longitud** **4** decimales

**para info2**

LCD imprimir Columna **0** Fila **0** **“ Km/h: ”**

crear texto con **“ Km/h: ”**

**Velocidad Km/h**

LCD imprimir Columna **0** Fila **1** **“ Altitud: ”**

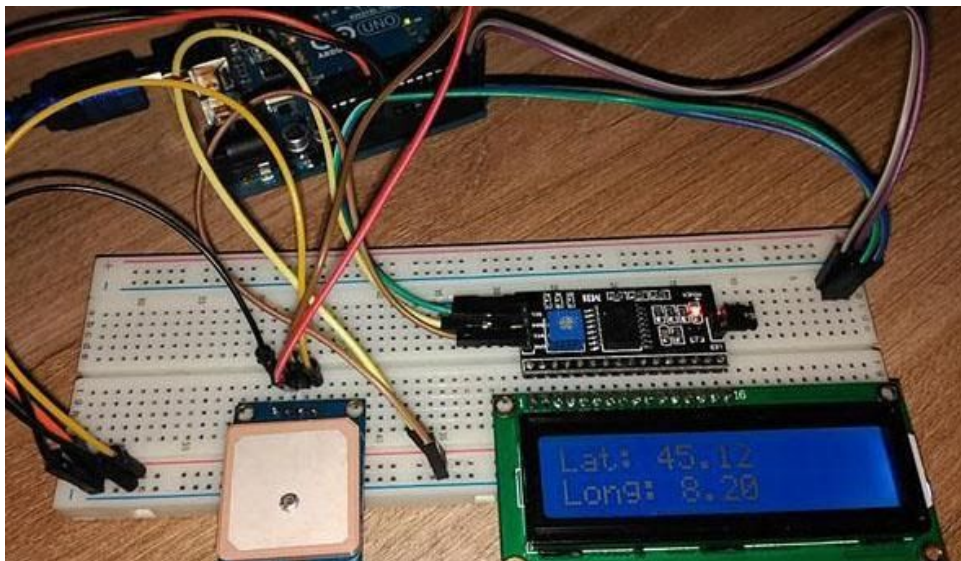
crear texto con **“ Altitud: ”**

**Altitud**





Imagen real del montaje:



## P27 - Aviso por exceso de velocidad

El siguiente montaje nos avisará cuando superemos una velocidad indicada. La velocidad máxima podemos ajustarla cambiando el valor de una variable en el programa. En caso de superar la velocidad indicada sonará un pitido producido por un zumbador. El valor de velocidad se obtendrá desde un módulo GPS.

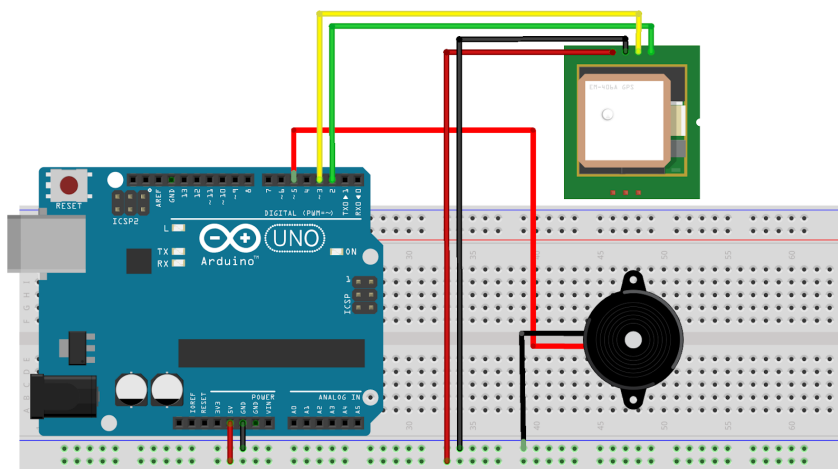
### Material necesario:

- 1 x módulo GPS
- 1 x Zumbador
- 1 x Arduino UNO
- Placa de prototipos y cables

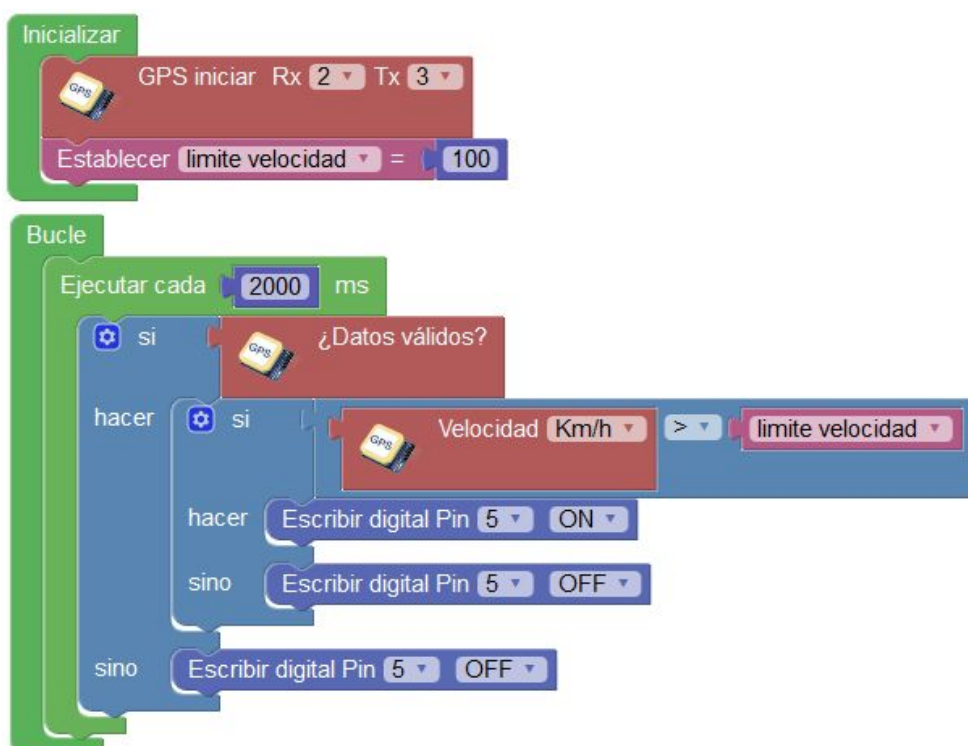
### Conexiones:

- GPS RX = Pin 2
- GPS TX = Pin 3
- Zumbador = Pin ~5

Esquema de conexión:



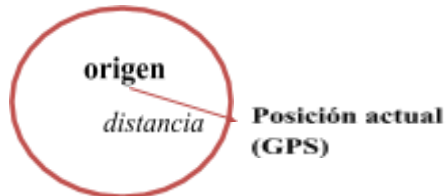
Programa ArduinoBlocks:



## P28 - Alarma por alejamiento

Mediante el GPS podemos conocer en todo momento la posición exacta en la que nos encontramos, de igual forma podemos calcular la distancia (en línea recta) respecto a una posición preestablecida de forma que sabemos si estamos lejos o cerca de ese punto.

Este montaje detecta la distancia respecto a un punto prefijado y activará un aviso sonoro (zumbador) en caso de alejarnos más de 500m de ese lugar. Este proyecto puede ser útil por ejemplo para evitar que niños o personas mayores se desorienten y se pierdan alejándose de una zona establecida.



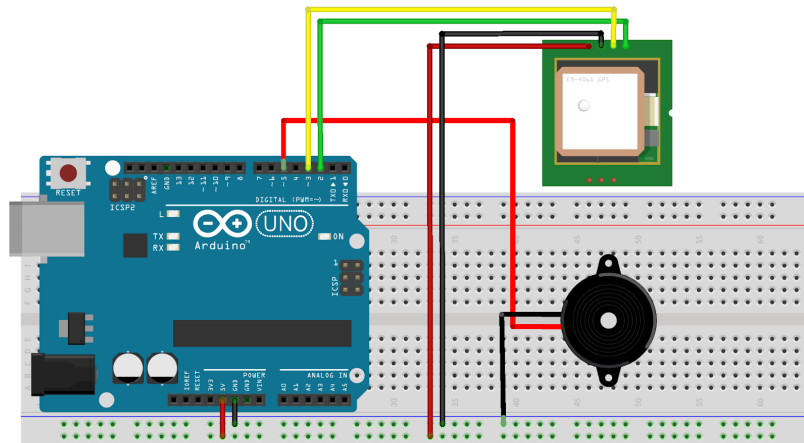
### Material necesario:

- 1 x módulo GPS
- 1 x Zumbador
- 1 x Arduino UNO
- Placa de prototipos
- Cables de interconexión

### Conexiones:

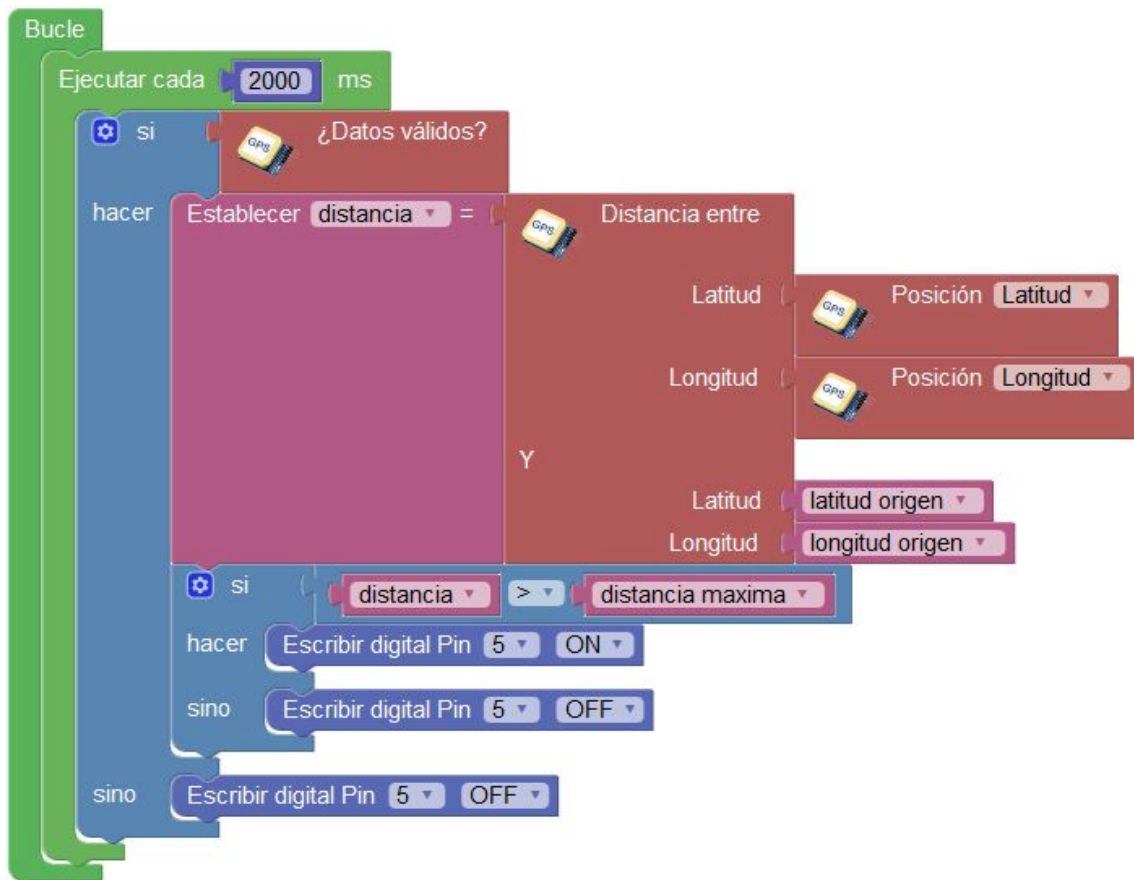
- GPS RX = Pin 2
- GPS TX = Pin 3
- Zumbador = Pin ~5

Esquema de conexión:



Programa ArduinoBlocks:





Para obtener las coordenadas GPS del punto origen podemos utilizar sitios webs como por ejemplo: <http://www.coordenadas-gps.com/> donde indicando una dirección o marcando sobre el mapa podemos obtener fácilmente la latitud y longitud del punto:

Dirección

**Obtener Coordenadas GPS**

---

GD (grados decimales)\*

Latitud

Longitud



## P29 - Registrador GPS en tarjeta SD

Un registrador GPS puede ser de gran utilidad para registrar nuestras rutas y su posterior procesamiento o visualización. Podemos utilizarlo por ejemplo para grabar nuestras rutas en bicicleta. El archivo generado en formato CSV se puede abrir fácilmente en aplicaciones de hoja de cálculo para su posterior procesamiento o con herramientas más potentes como por ejemplo la aplicación web GpsVisualizer o Google Maps.

Para poder visualizar el mapa con el recorrido grabado correctamente en la web GpsVisualizer (<http://www.gpsvisualizer.com/>) debemos generar el archivo CSV con un formato específico según nos indican en su sitio web:

```
type,latitude,longitude,alt
T,45.9874167,-76.8752333,79.8
T,45.9860000,-76.8737833,111.4
T,45.9850500,-76.8724833,107.9
T,45.9844000,-76.8716333,120.0
T,45.9839500,-76.8710000,117.5
```

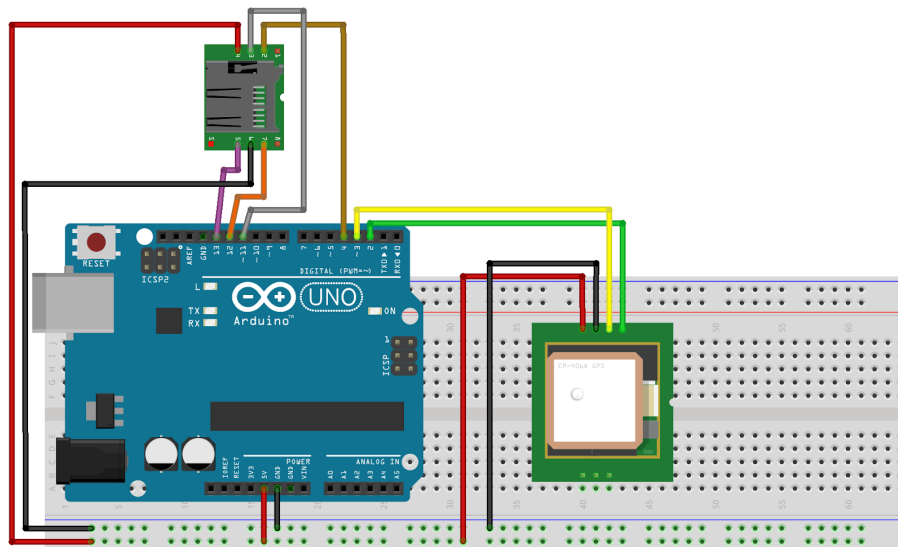
### Material necesario:

- 1 x módulo GPS
- 1 x módulo tarjeta SD
- 1 x Arduino UNO
- Placa de prototipos
- Cables de interconexión

### Conexiones:

- GPS RX = Pin 2
- GPS TX = Pin 3
- SD = SPI

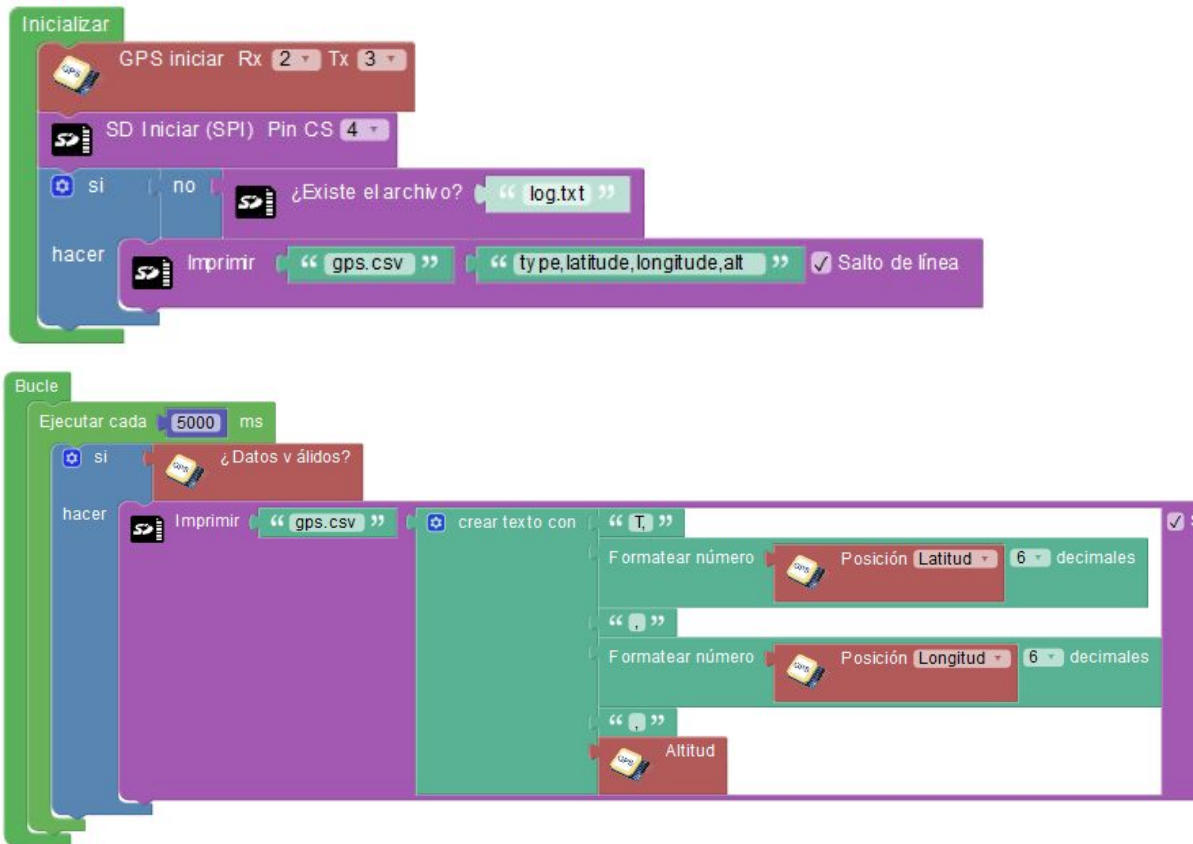
Esquema de conexión:



Programa ArduinoBlocks:

*Si el archivo "gps.csv" no existe en la tarjeta inicializa la primera línea con el texto "type,latitude,longitude,alt". Cada 5s si tenemos datos válidos desde el GPS se registra una nueva línea con la información de posición y altitud.*





Una vez finalizado el registro de datos podemos copiar el archivo gps.csv de la tarjeta de memoria a un PC y con la ayuda de la web GpsVisualizer obtendremos la ruta dibujada sobre el mapa:

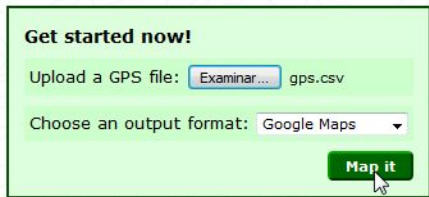
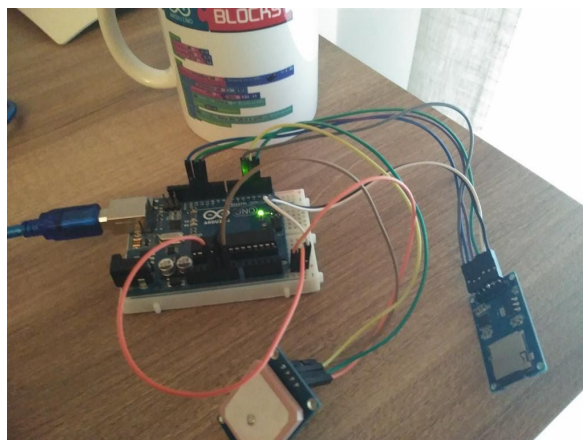
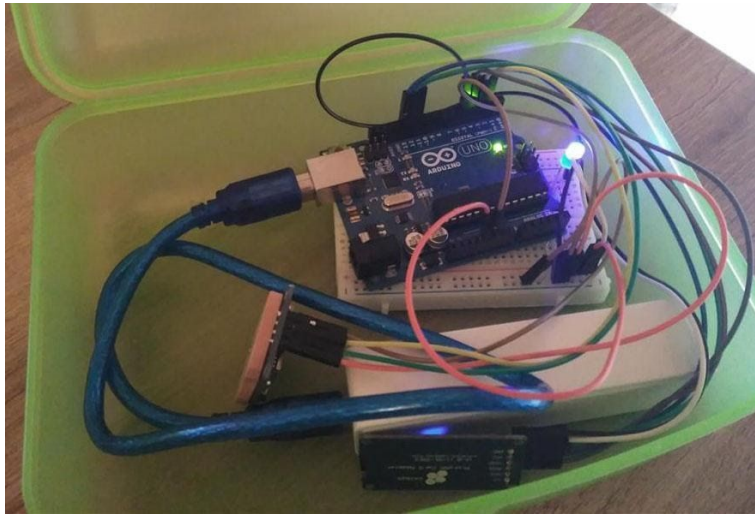


Imagen del montaje real con módulo GPS y tarjeta micro SD





Versión “empaquetada” y conectada a un power-bank USB como fuente de alimentación



## P30 - Registro de temperatura/humedad en SD

Controlar la temperatura y humedad de un lugar puede ser muy útil para comprobar si nuestro sistema de calefacción o refrigeración funciona bien. Normalmente no podemos estar visualizando los valores de temperatura y humedad en todo momento por lo que puede ser una gran idea registrar estos valores en un archivo para poder posteriormente visualizar los datos.

### Material necesario:

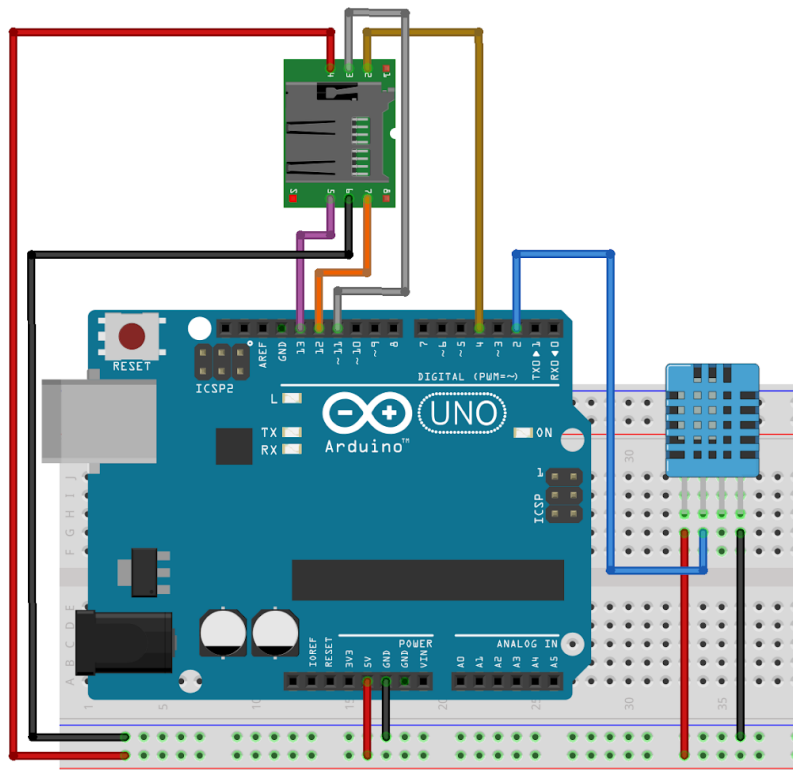
- 1 x sensor DHT11
- 1 x módulo tarjeta SD
- 1 x Arduino UNO
- Placa de prototipos
- Cables de interconexión

### Conexiones:

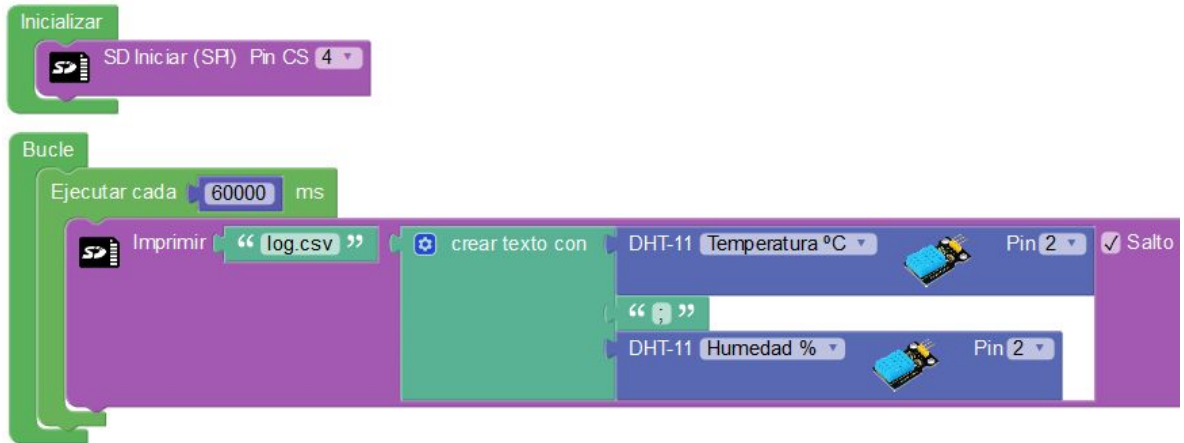
SD = SPI

DHT11 = Pin 2

Esquema de conexión:



Programa ArduinoBlocks:



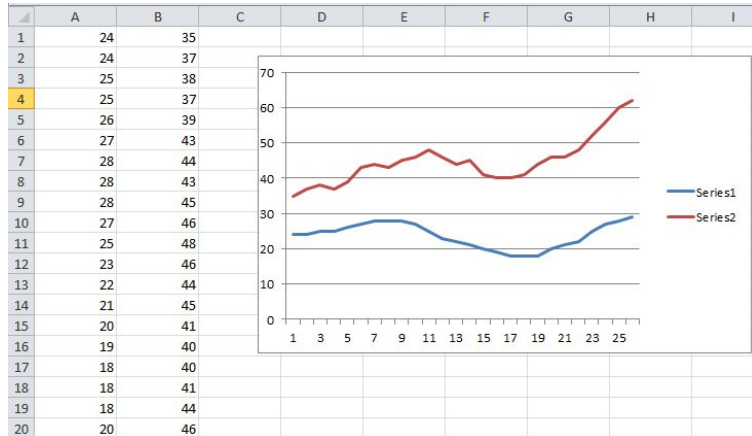
El archivo generado en la memoria SD se llama "log.csv".

Un ejemplo de los datos almacenados visualizados en un editor de texto plano y procesado en una aplicación de hoja de cálculo:

*Editor de texto: Importación a LibreOffice Calc o Excel y creación de gráficas con los valores:*

```

24.00;35.00
24.00;37.00
25.00;38.00
25.00;37.00
26.00;39.00
27.00;43.00
28.00;44.00
28.00;43.00
28.00;45.00
27.00;46.00
25.00;48.00
23.00;46.00
22.00;44.00
21.00;45.00
20.00;41.00
19.00;40.00
18.00;40.00
18.00;41.00
18.00;44.00
20.00;46.00
21.00;46.00
    
```



*Módulo de tarjeta micro SD (conexión SPI) utilizado en el proyecto:*



## P31 - Control de servo con acelerómetro

Este proyecto permite controlar un servo a partir de los movimientos detectados por un acelerómetro. Por ejemplo el acelerómetro podría estar fijado al dedo de una persona y con sus movimientos controlar el movimiento de un dedo robótico movido por un servo.

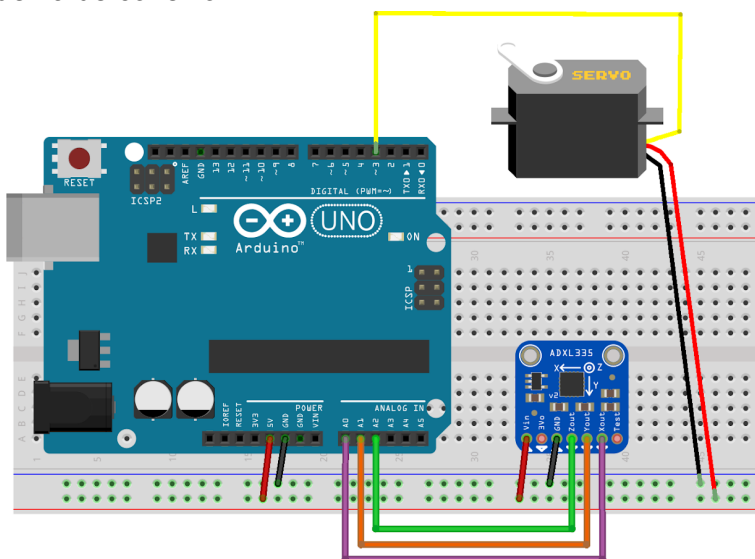
### Material necesario:

- 1 x Módulo acelerómetro ADXL335
- 1 x Servomotor
- 1 x Arduino UNO
- Placa de prototipos
- Cables de interconexión

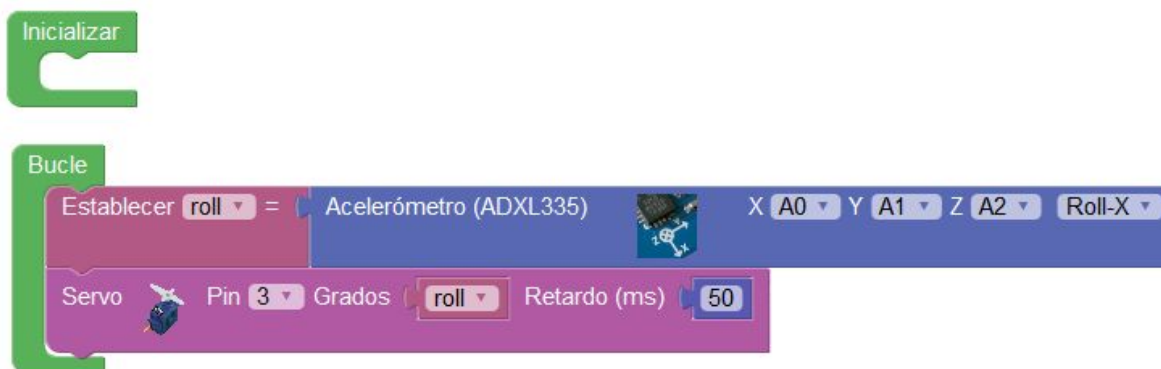
### Conexiones:

Servo = Pin ~3  
ADXL335 = A0,A1,A2

Esquema de conexión:



Programa ArduinoBlocks:



Vídeo del proyecto funcionando:  
<https://youtu.be/aeScceN1D7w>

## P32 - Sensor de caídas con aviso a emergencias

### (Android+Bluetooth)

Cuando se produce un impacto, se produce una desaceleración fuerte al para bruscamente un cuerpo. Esta desaceleración o “frenazo brusco” lo podemos detectar con un acelerómetro.

Siguiendo esta teoría vamos a realizar un proyecto en el que ante una caída se envíe una señal vía Bluetooth a una aplicación móvil Android que automáticamente llamará a un teléfono de emergencias.

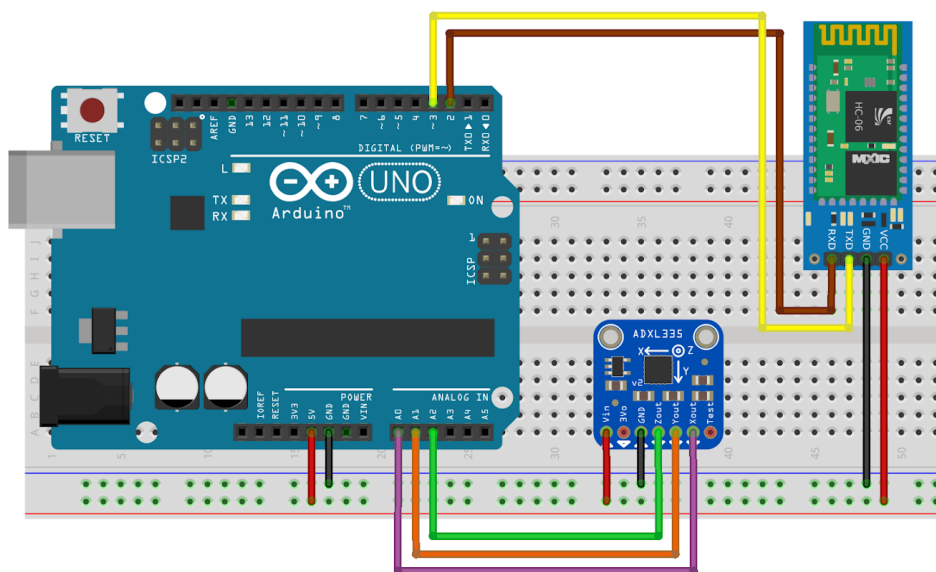
#### Material necesario:

- 1 x Módulo acelerómetro ADXL335
- 1 x Módulo Bluetooth HC-06
- 1 x Arduino UNO
- Placa de prototipos
- Cables de interconexión

#### Conexiones:

Bluetooth RX = Pin 2  
Bluetooth TX = Pin 3  
ADXL335 = A0,A1,A2

Esquema de conexión:



Programa ArduinoBlocks:

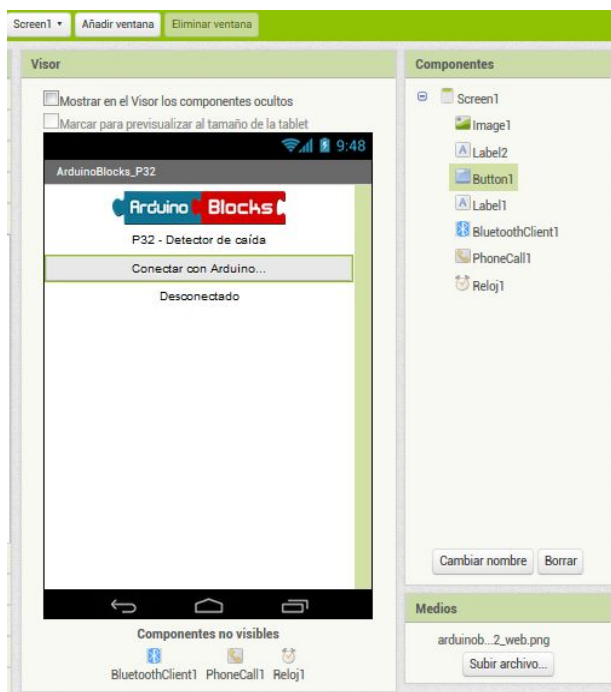
*El programa comprueba continuamente la aceleración en el eje Z, en caso de detectar un valor menor que -2 (una desaceleración fuerte) envía el valor 112 a través de la conexión Bluetooth.*

```

Inicializar
  Iniciar Rx 2 Tx 3 Baudios 9600
  Nombre "ArduinoBlocks" Código PIN "1234"

Bucle
  Establecer aZ = Acelerómetro (ADXL335) X A0 Y A1 Z A2 Accel-Z
  si aZ ≤ -2
  hacer
    Enviar byte 112
  Esperar 10000 milisegundos
  
```

Aplicación Android con AppInventor:



```

cuando Button1 .Clic
  ejecutar
    si
      llamar BluetoothClient1 .Conectar dirección "00:11:22:33:44:55"
    entonces poner Label1 .Texto como "Conectado"
    si no poner Label1 .Texto como "Error conectando"

cuando Reloj1 .Temporizador
  ejecutar
    si BluetoothClient1 .Conectado
    entonces
      si llamar BluetoothClient1 .BytesDisponiblesParaRecibir > 0
      entonces
        si llamar BluetoothClient1 .RecibirNúmero1ByteSinSigno = 112
        entonces llamar PhoneCall1 .LlamarPorTeléfono
  
```



## P33 - MQTT (IoT): Control de led RGB

Como ya hemos visto un led RGB nos permite obtener multitud de tonos de luz de diferentes colores simplemente combinando los valores de rojo, verde y azul de los que se compone. Este proyecto nos va a permitir controlar un led RGB desde el móvil y además vía internet, es decir, desde cualquier lugar del mundo con conexión a internet podremos ajustar nuestro led al color e intensidad deseado.

Para ello utilizaremos una Shield Ethernet que debemos conectar a internet mediante un router.

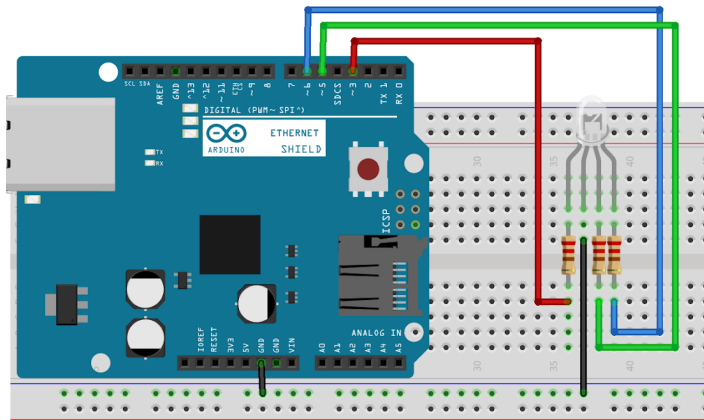
### Material necesario:

- 1 x Led RGB (cátodo común)
- 1 x Arduino Ethernet Shield
- 1 x Arduino UNO
- Placa de prototipos y cables

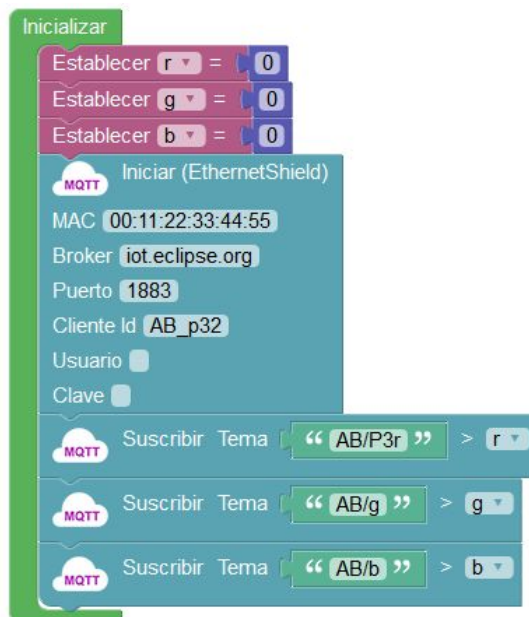
### Conexiones:

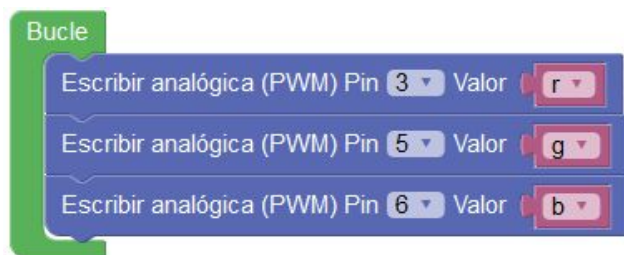
- RGB R = Pin ~3
- RGB G = Pin ~5
- RGB B = Pin ~6

Esquema de conexión:



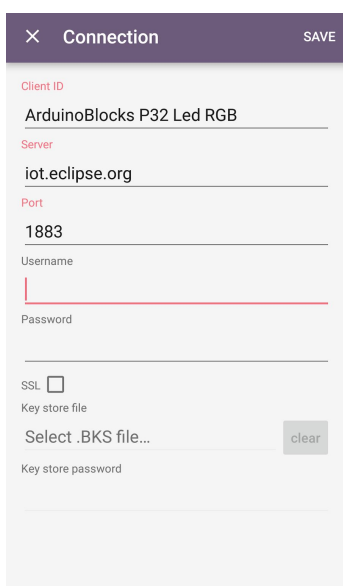
Programa ArduinoBlocks:



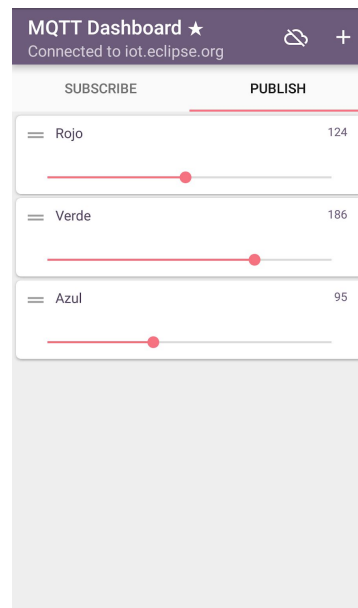
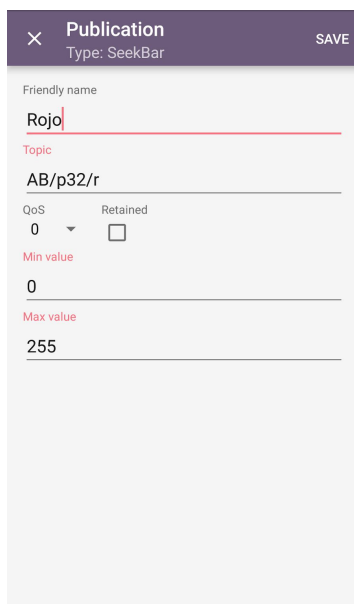


Para el control utilizaremos una aplicación como MQTT Dashboard (Android) o similar.

*Configuración de la conexión con el broker MQTT*



*Creación de las barras de desplazamiento asociadas a cada tema "AB/p32/r", "AB/p32/g", "AB/p32/b"*



**IMPORTANTE:** En este proyecto hemos utilizado un broker MQTT público y gratuito con fines experimentales, cualquier cliente que conecte a este broker y utilice los mismos temas (topics) podrá controlar o monitorizar nuestro proyecto. En un proyecto real debemos configurar nuestro propio broker MQTT seguro o utilizar uno público con seguridad (ver apdo. MQTT en el capítulo 3.3.13)

## P34 - MQTT (IoT): Estación meteorológica

Gracias al protocolo MQTT vamos a implementar una sencilla estación meteorológica cuyos datos serán publicados por internet y visualizados con una aplicación cliente MQTT en un dispositivo móvil desde cualquier parte del mundo.

Los datos son actualizados cada 30s.

### Material necesario:

- 1 x Arduino Ethernet Shield
- 1 x Arduino UNO
- 1 x DHT-11
- 1 x LDR
- 1 x Sensor de lluvia
- Placa de prototipos y cables

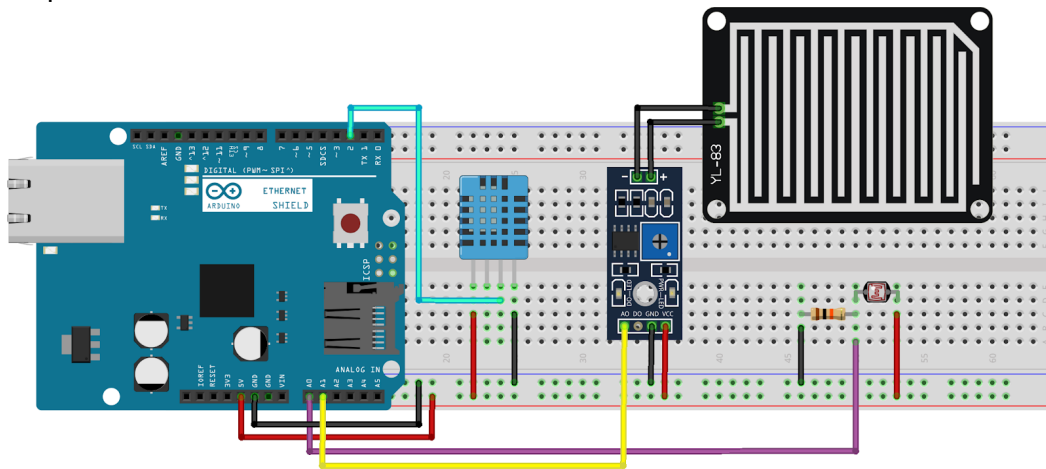
### Conexiones:

DHT-11 = Pin 2

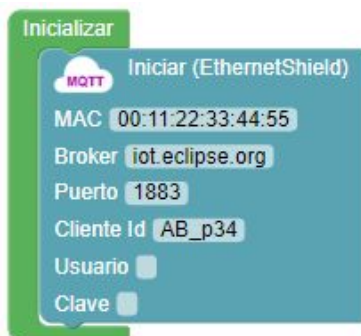
LDR = Pin A0

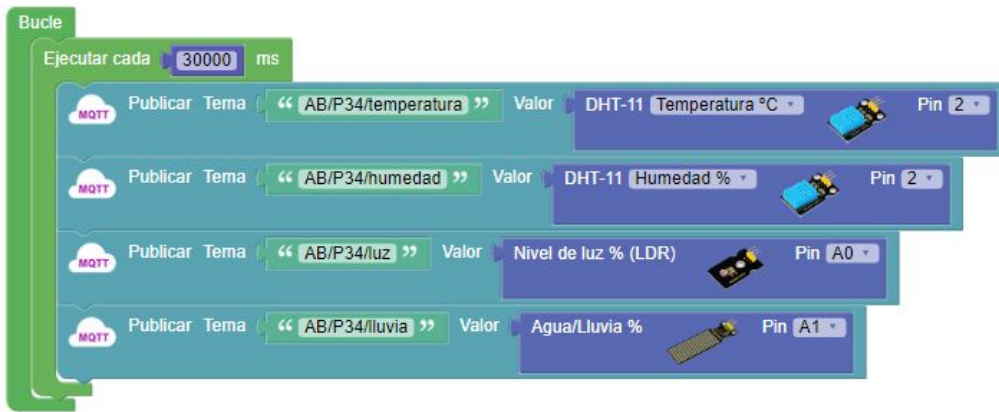
Sensor lluvia = Pin A1

Esquema de conexión:

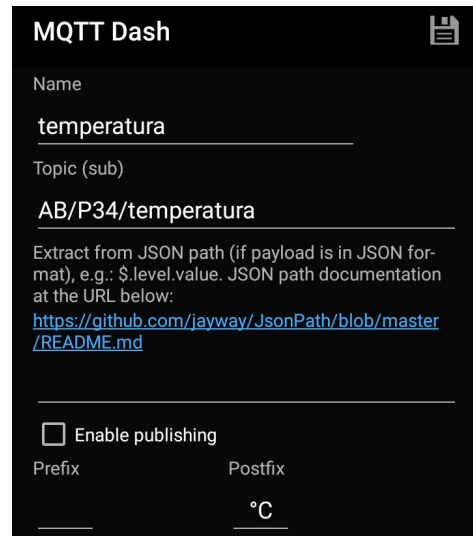
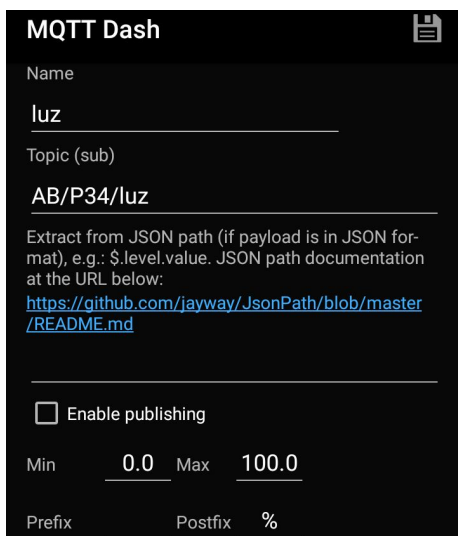


Programa ArduinoBlocks:





Ejemplo de monitorización desde aplicación MQTT Dash (Android):



<https://play.google.com/store/apps/details?id=net.routix.mqttdash&hl=es>

## P35 - MQTT (IoT): Control domótico

En proyectos anteriores hemos realizado sencillas simulaciones de una instalación domótica controlada por Bluetooth. En este proyecto implementamos una funcionalidad similar con la ventaja del protocolo MQTT vía internet y del sencillo y cómodo control desde un terminal móvil.

Vamos a controlar dos puntos de luz accionados por relé y la posición de una persiana simulada con un servo.

Por otro lado detectaremos la presencia con un sensor PIR y el nivel de luz ambiente con una LDR.

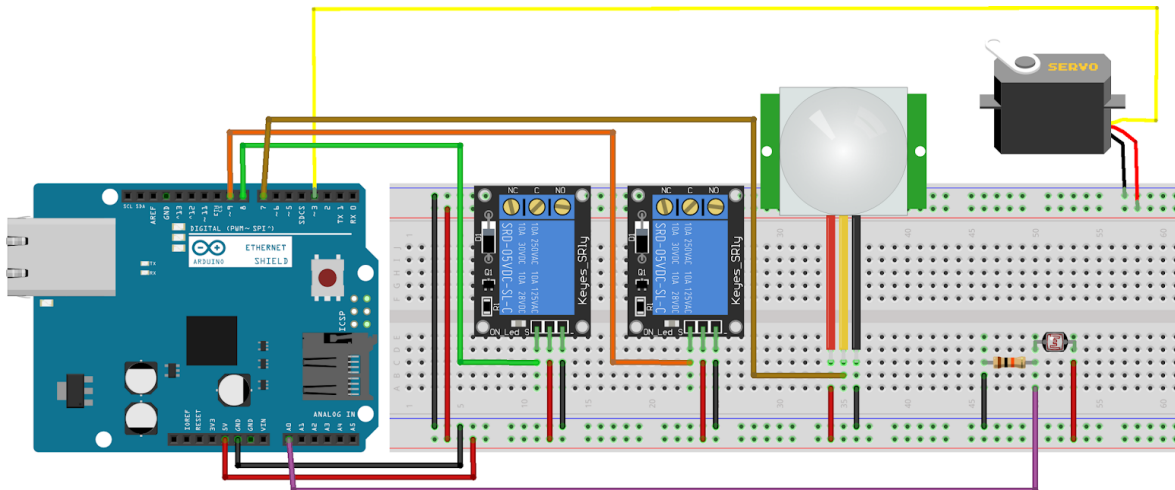
### Material necesario:

- 2 x Módulo relé
- 1 x Arduino Ethernet Shield
- 1 x Arduino UNO
- 1 x Módulo LDR
- 1 x Sensor PIR
- 1 x Servo
- Placa de prototipos y cables

### Conexiones:

- Servo = Pin 3
- Relé 1 = Pin 8
- Relé 2 = Pin 9
- Sensor PIR = Pin 7
- LDR = Pin A0

Esquema de conexión:



Programa ArduinoBlocks:

```

Inicializar
  Establecer luz1 = 0
  Establecer luz2 = 0
  Establecer persiana = 0
  Iniciar (EthernetShield)
  MAC 00:11:22:33:44:55
  Broker iot.eclipse.org
  Puerto 1883
  Cliente Id AB_p34
  Usuario
  Clave
  Suscribir Tema "AB/P35/luz1" > luz1
  Suscribir Tema "AB/P35/luz2" > luz2
  Suscribir Tema "AB/P35/persiana" > persiana

Bucle
  Ejecutar cada 250 ms
  actualizar estado
  Ejecutar cada 5000 ms
  enviar datos de los sensores

para enviar datos de los sensores
  Publicar Tema "AB/P35/presencia" Valor Detector de movimiento (PIR) Pin 7
  Publicar Tema "AB/P35/luz" Valor Nivel de luz % (LDR) Pin A0

para actualizar estado
  si luz1 = 1
  hacer Relé Pin 8 Estado ON
  sino Relé Pin 8 Estado OFF
  si luz2 = 1
  hacer Relé Pin 9 Estado ON
  sino Relé Pin 9 Estado OFF
  Establecer persiana = limitar persiana entre 0 y 180
  Servo Pin 3 Grados persiana Retardo (ms) 0
  
```



Ejemplo de control y monitorización desde aplicación MQTT Dashboard (Android):

**Publication** CREATE  
Type: SeekBar

Friendly name  
**persiana**

Topic  
**AB/P35/persiana**

QoS  Retained  
**0**

Min value  
**0**

Max value  
**180**

**MQTT Dashboard** ★  
Connected to iot.eclipse.org

**SUBSCRIBE** **PUBLISH**

**nivel de luz ambiente** **37 %**  
3 seconds

**presencia detectada** **1**  
42 seconds

**Publication** CREATE  
Type: Switch

Friendly name  
**luz 1**

Topic  
**AB/P35/luz1**

QoS  Retained  
**0**

Text (On)  
**On**

Text (Off)  
**Off**

Publish value (On)  
**1**

Publish value (Off)  
**0**

**MQTT Dashboard** ★  
Connected to iot.eclipse.org

**SUBSCRIBE** **PUBLISH**

**luz 1** **1**  
Off  On

**luz 2** **n/a**  
Off  On

**persiana** **128**



<https://play.google.com/store/apps/details?id=com.thn.iotmqttdashboard&hl=es>

## P36 - Robot con servos - Bluetooth

Vamos a realizar un pequeño vehículo que utiliza dos servos de rotación continua para el movimiento y un módulo Bluetooth HC-06 para comunicarse con una aplicación móvil y ser controlado de forma sencilla.

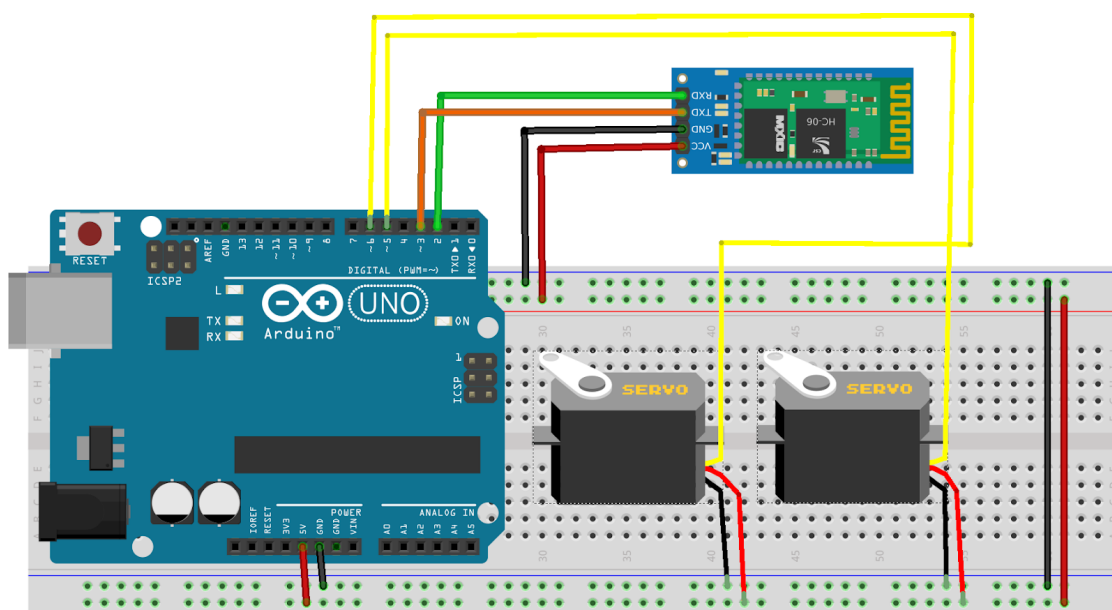
### Material necesario:

- 1 x Arduino UNO
- 1 x Sensor shield
- 1 x Módulo Bluetooth HC-06
- 2 x Servo rotación continua
- 1 x Batería 9v 1000mAh
- 1 x Cables

### Conexiones:

- Servo 1 = Pin 5
- Servo 2 = Pin 6
- HC-06 Rx=2
- HC-06 Tx=3

Esquema de conexión:

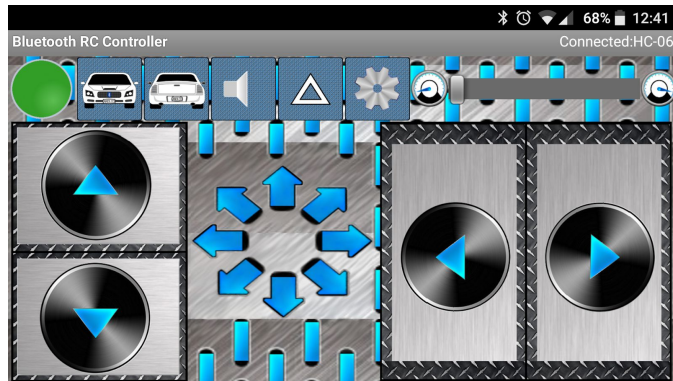


Plataforma utilizada y montaje final:



Para el control se ha utilizado la aplicación gratuita “Bluetooth RC Controller” (Android):

<https://play.google.com/store/apps/details?id=braulio.calle.bluetoothRCcontroller>



La aplicación enviará unos códigos a través de la conexión para cada acción:

<i>Avanzar</i>	F (ASCII: 70)
<i>Retroceder</i>	B (ASCII: 66)
<i>Izquierda</i>	L (ASCII: 76)
<i>Derecha</i>	R (ASCII: 82)
<i>Avanzar derecha</i>	G (ASCII: 71)
<i>Avanzar izquierda</i>	I (ASCII: 73)
<i>Parar</i>	S (ASCII: 83)

Control de los servos de rotación continua:



90°	parado
0°	gira en un sentido a máxima velocidad
180°	gira en sentido contrario a máxima velocidad

Programa ArduinoBlocks:

```

Inicializar
  Bluetooth Iniciar Rx: 2 Tx: 3 Baudios: 9600
  Nombre: "AB-Car" "1234"
  parar

Bucle
  si ¿Datos recibidos?
  hacer
    Establecer comando = Recibir byte
    si comando = 83
    hacer parar
    si comando = 68
    hacer parar
    si comando = 70
    hacer avanzar
    si comando = 66
    hacer retroceder
    si comando = 76
    hacer girar izquierda
    si comando = 82
    hacer girar derecha
    si comando = 73
    hacer avanzar derecha
    si comando = 71
    hacer avanzar izquierda
  
```

```

para parar
  Servo Pin 5 Grados Ángulo 90º Retardo (ms) 0
  Servo Pin 6 Grados Ángulo 90º Retardo (ms) 0
  
```

```

para avanzar
  Servo Pin 5 Grados Ángulo 100º Retardo (ms) 0
  Servo Pin 6 Grados Ángulo 80º Retardo (ms) 0
  
```

```

para girar derecha
  Servo Pin 5 Grados Ángulo 100º Retardo (ms) 0
  Servo Pin 6 Grados Ángulo 100º Retardo (ms) 0
  
```

```

para avanzar derecha
  Servo Pin 5 Grados Ángulo 120º Retardo (ms) 0
  Servo Pin 6 Grados Ángulo 80º Retardo (ms) 0
  
```

```

para retroceder
  Servo Pin 5 Grados Ángulo 80º Retardo (ms) 0
  Servo Pin 6 Grados Ángulo 100º Retardo (ms) 0
  
```

```

para girar izquierda
  Servo Pin 5 Grados Ángulo 80º Retardo (ms) 0
  Servo Pin 6 Grados Ángulo 80º Retardo (ms) 0
  
```

```

para avanzar izquierda
  Servo Pin 5 Grados Ángulo 100º Retardo (ms) 0
  Servo Pin 6 Grados Ángulo 60º Retardo (ms) 0
  
```

## P37 - Robot con motores DC - Bluetooth

Los robots móviles con motores DC son los más habituales, el proyecto anterior con servos de rotación continua es muy sencillo pero es más habitual y fácil de conseguir robots con motores de corriente continua.

Para controlar este tipo de motores como se ha descrito anteriormente necesitamos un driver o controlador de puente en H que nos permite controlar la dirección de giro y la velocidad.

De igual forma que en el proyecto anterior se utiliza un módulo HC-06 para control vía Bluetooth a través de la aplicación "Bluetooth RC Controller" (Android).

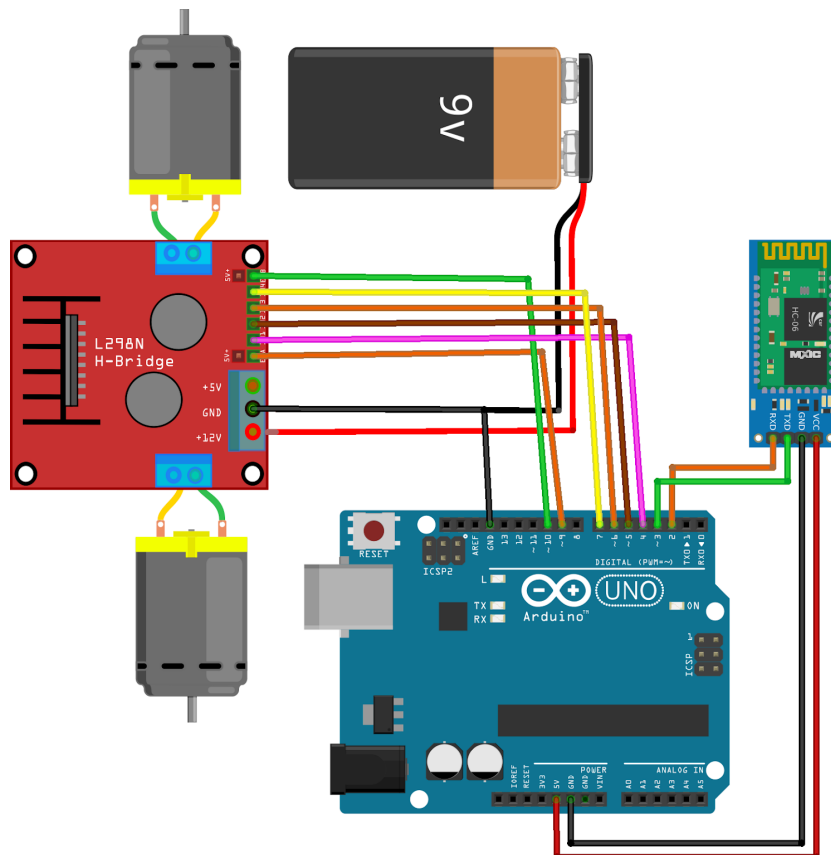
### Material necesario:

- 1 x Arduino UNO
- 1 x Módulo puente en H
- 1 x Módulo Bluetooth HC-06
- 1 x Cables
- 2 x Motores DC + ruedas
- 1 x Estructura vehículo

### Conexiones:

- Puente-H EN1 = Pin 9
- Puente-H M1 = Pines 4,5
- Puente-H EN2 = Pin 10
- Puente-H M2 = Pines 6,7
- Bluetooth RX = Pin 2
- Bluetooth TX = Pin 3

Esquema de conexión:





Programa ArduinoBlocks:

The program is structured as follows:

- Inicializar:**
  - Bluetooth: Iniciar Rx 2, Tx 3, Baudios 9600
  - Bluetooth: Nombre "AB-Car", "1234"
  - parar
- Bucle:**
  - si ¿Datos recibidos?
    - hacer Establecer comando = Recibir byte
      - si comando = 83
        - hacer parar
      - si comando = 68
        - hacer parar
      - si comando = 70
        - hacer avanzar
      - si comando = 66
        - hacer retroceder
      - si comando = 76
        - hacer girar izquierda
      - si comando = 82
        - hacer girar derecha
      - si comando = 73
        - hacer avanzar derecha
      - si comando = 71
        - hacer avanzar izquierda
- para parar:**
  - Escribir analógica (PWM) Pin 9 Valor 0
  - Escribir digital Pin 4 OFF
  - Escribir digital Pin 5 OFF
  - Escribir analógica (PWM) Pin 10 Valor 0
  - Escribir digital Pin 6 OFF
  - Escribir digital Pin 7 OFF
- para avanzar:**
  - Escribir analógica (PWM) Pin 9 Valor 255
  - Escribir digital Pin 4 OFF
  - Escribir digital Pin 5 ON
  - Escribir analógica (PWM) Pin 10 Valor 255
  - Escribir digital Pin 6 ON
  - Escribir digital Pin 7 OFF
- para girar derecha:**
  - Escribir analógica (PWM) Pin 9 Valor 255
  - Escribir digital Pin 4 OFF
  - Escribir digital Pin 5 ON
  - Escribir analógica (PWM) Pin 10 Valor 255
  - Escribir digital Pin 6 OFF
  - Escribir digital Pin 7 ON
- para girar izquierda:**
  - Escribir analógica (PWM) Pin 9 Valor 255
  - Escribir digital Pin 4 ON
  - Escribir digital Pin 5 OFF
  - Escribir analógica (PWM) Pin 10 Valor 255
  - Escribir digital Pin 6 ON
  - Escribir digital Pin 7 OFF

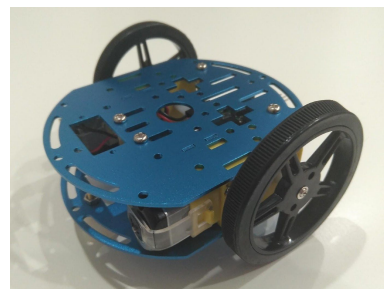
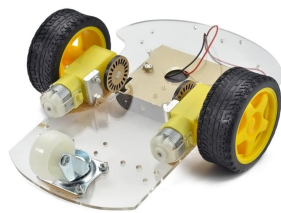


```
para retroceder
  Escribir analógica (PWM) Pin 9 Valor 255
  Escribir digital Pin 4 ON
  Escribir digital Pin 5 OFF
  Escribir analógica (PWM) Pin 10 Valor 255
  Escribir digital Pin 6 OFF
  Escribir digital Pin 7 ON
```

```
para avanzar derecha
  Escribir analógica (PWM) Pin 9 Valor 255
  Escribir digital Pin 4 OFF
  Escribir digital Pin 5 ON
  Escribir analógica (PWM) Pin 10 Valor 200
  Escribir digital Pin 6 ON
  Escribir digital Pin 7 OFF
```

```
para avanzar izquierda
  Escribir analógica (PWM) Pin 9 Valor 200
  Escribir digital Pin 4 OFF
  Escribir digital Pin 5 ON
  Escribir analógica (PWM) Pin 10 Valor 255
  Escribir digital Pin 6 ON
  Escribir digital Pin 7 OFF
```

Ejemplo de estructuras para robots de 2 ruedas:



## P38 - Robot con motores DC - Evita obstáculos

Un robot evita obstáculos es un tipo de robot autónomo que automáticamente detecta obstáculos delante de él y los intenta esquivar. El robot se mueve continuamente girando al detectar un obstáculo.

Para la detección de obstáculos se utiliza un sensor HC-SR04. Al detectar un obstáculo el robot gira de forma aleatoria a la izquierda o a la derecha para cambiar de dirección.

El control de los motores DC se realizará con un modulo de puente en H como en los proyectos anteriores.

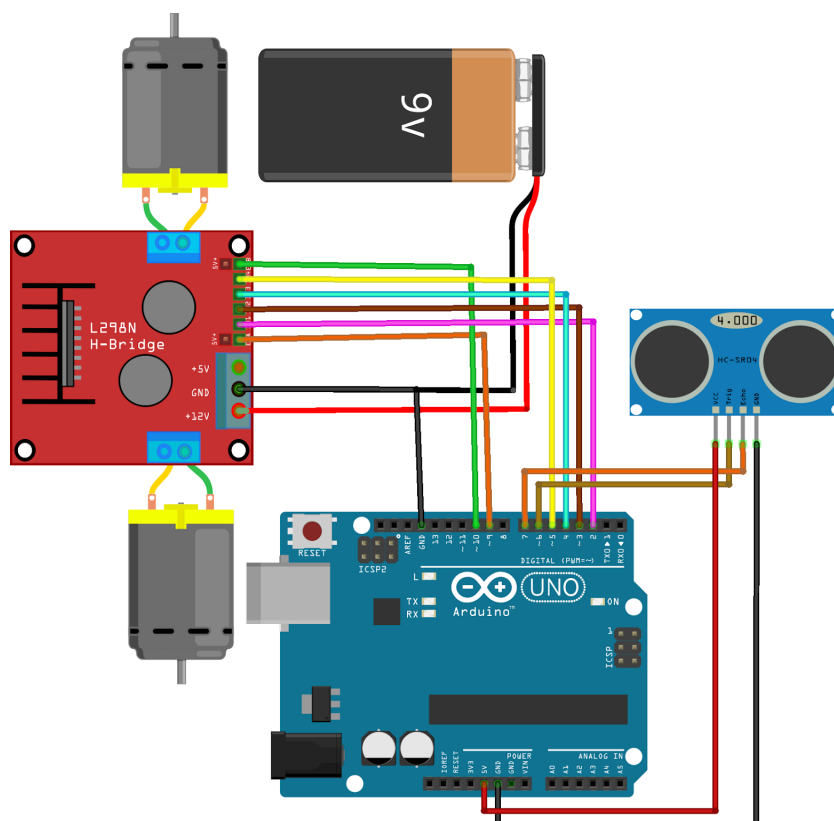
### Material necesario:

- 1 x Arduino UNO
- 1 x Módulo puente en H
- 1 x Módulo HC-SR04
- 1 x Cables
- 2 x Motores DC + ruedas
- 1 x Estructura vehículo

### Conexiones:

- Puente-H EN1 = Pin 9
- Puente-H M1 = Pines 2,3
- Puente-H EN2 = Pin 10
- Puente-H M2 = Pines 4,5
- HC-SR04 Trigger = Pin 6
- HC-SR04 Echo = Pin 7

Esquema de conexión:



Programa ArduinoBlocks:

```

Inicializar
  Establecer velocidad = 255
  Establecer tiempo giro = 350
  Establecer dist min = 15
  Establecer direccion = 0
  parar
  Esperar 2000 milisegundos

Bucle
  Establecer distancia = Distancia (cm) [Trigger] 6 [Echo] 7
  si distancia > 0 y distancia < dist min
    hacer
      parar
      Establecer direccion = entero aleatorio de 1 a 10
      si direccion < 5
        hacer girar izquierda
      sino girar derecha
      Esperar 1000 milisegundos
    sino avanzar

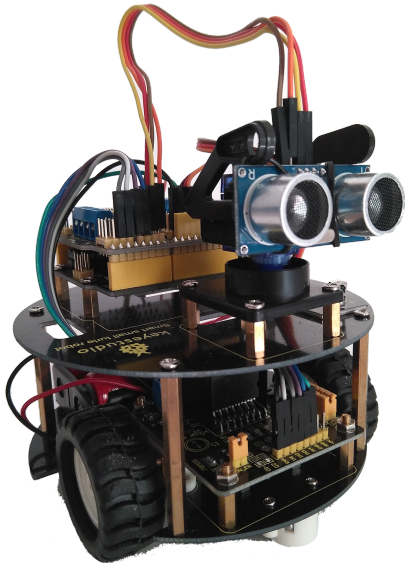
para avanzar
  Escribir analógica (PWM) Pin 9 Valor velocidad
  Escribir digital Pin 2 OFF
  Escribir digital Pin 3 ON
  Escribir analógica (PWM) Pin 10 Valor velocidad
  Escribir digital Pin 4 ON
  Escribir digital Pin 5 OFF

para parar
  Escribir analógica (PWM) Pin 9 Valor 0
  Escribir digital Pin 2 OFF
  Escribir digital Pin 3 OFF
  Escribir analógica (PWM) Pin 10 Valor 0
  Escribir digital Pin 4 OFF
  Escribir digital Pin 5 OFF
  Esperar 500 milisegundos
  
```

```
para girar izquierda
  Escribir analógica (PWM) Pin 9 Valor velocidad
  Escribir digital Pin 2 ON
  Escribir digital Pin 3 OFF
  Escribir analógica (PWM) Pin 10 Valor velocidad
  Escribir digital Pin 4 ON
  Escribir digital Pin 5 OFF
  Esperar tiempo giro milisegundos

para girar derecha
  Escribir analógica (PWM) Pin 9 Valor velocidad
  Escribir digital Pin 2 OFF
  Escribir digital Pin 3 ON
  Escribir analógica (PWM) Pin 10 Valor velocidad
  Escribir digital Pin 4 OFF
  Escribir digital Pin 5 ON
  Esperar tiempo giro milisegundos
```

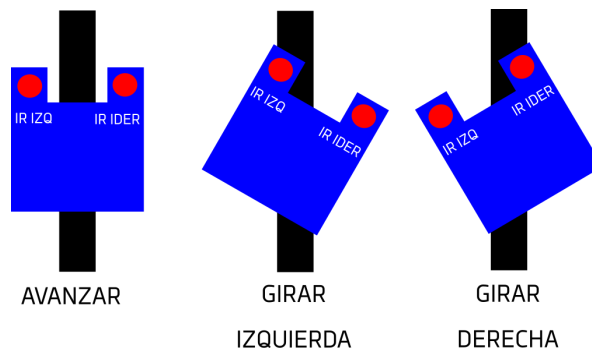
Ejemplo de robot evita obstáculos:



## P39 - Robot con motores DC - Sigue líneas

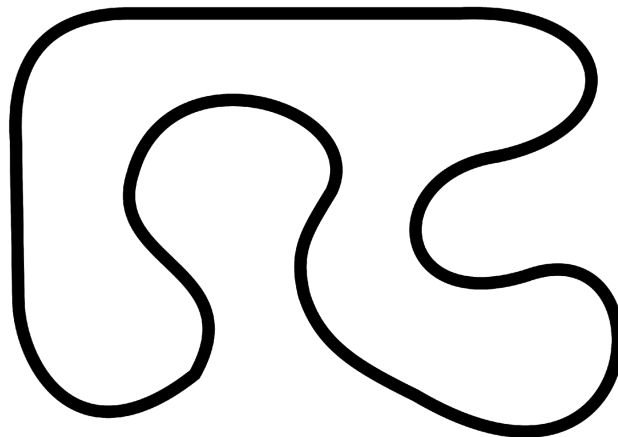
Un robot sigue líneas consiste en un robot motorizado que seguirá el recorrido marcado por una línea negra sobre una superficie blanca.

El seguimiento de la línea se realiza gracias a unos sensor IR que detectan si la superficie sobre la que están es blanca o negra (por la reflexión de la luz IR).



El tiempo de giro y la velocidad se debe ajustar en función de los motores utilizados, la distancia entre los sensores, el ancho de las líneas, etc.

Ejemplo de circuito para robot sigue líneas:



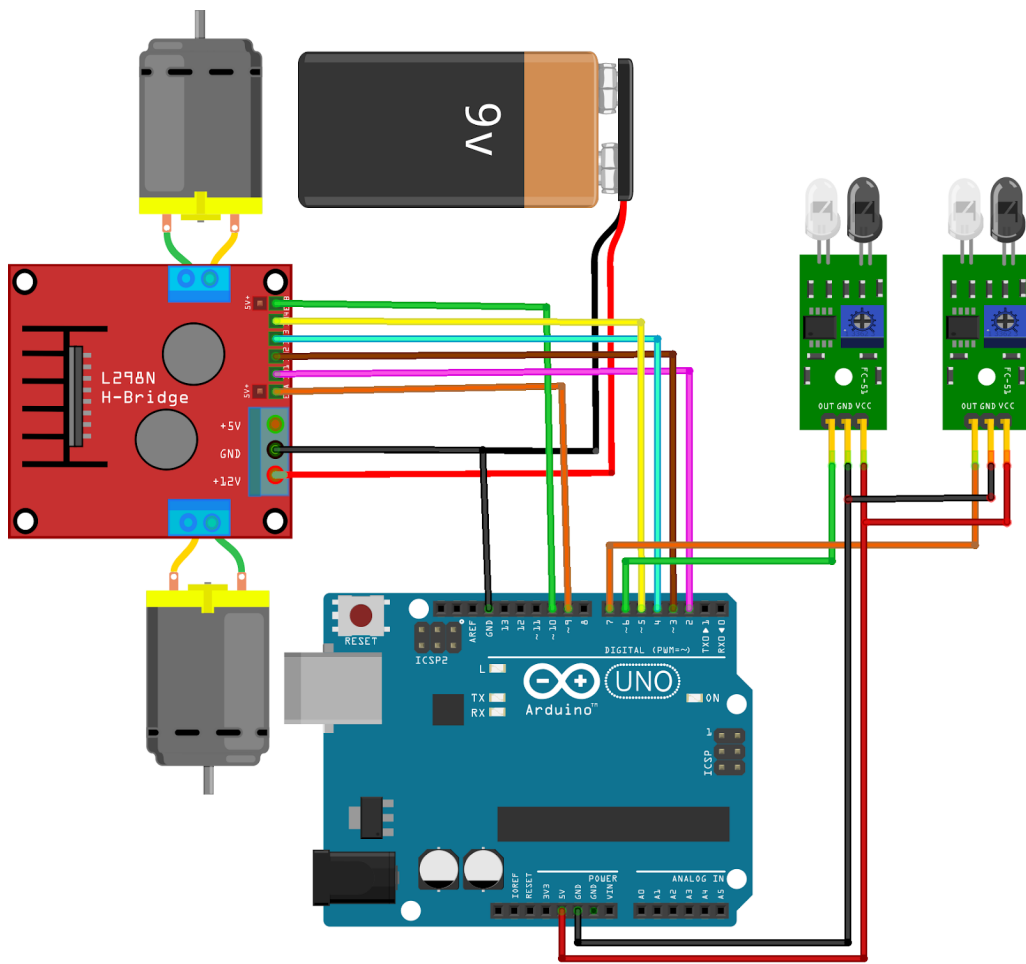
### Material necesario:

- 1 x Arduino UNO
- 1 x Módulo puente en H
- 2 x Sensor IR
- 1 x Cables
- 2 x Motores DC + ruedas
- 1 x Estructura vehículo

### Conexiones:

Puente-H EN1 = Pin 9  
Puente-H M1 = Pines 2,3  
Puente-H EN2 = Pin 10  
Puente-H M2 = Pines 4,5  
Sensor IR Izq. = Pin 6  
Sensor IR der. = Pin 7

Esquema de conexión:





Programa ArduinoBlocks:

```

Inicializar
  Establecer velocidad = 150
  parar
  
```

```

Bucle
  Establecer negro en izquierda = no Leer digital Pin 7
  Establecer negro en derecha = no Leer digital Pin 8
  si negro en izquierda
  hacer girar izquierda
  sino si negro en derecha
  hacer girar derecha
  sino avanzar
  
```

```

para avanzar
  Escribir analógica (PWM) Pin 9 Valor velocidad
  Escribir digital Pin 2 OFF
  Escribir digital Pin 3 ON
  Escribir analógica (PWM) Pin 10 Valor velocidad
  Escribir digital Pin 4 ON
  Escribir digital Pin 5 OFF
  
```

```

para girar izquierda
  Escribir analógica (PWM) Pin 9 Valor velocidad
  Escribir digital Pin 2 OFF
  Escribir digital Pin 3 ON
  Escribir analógica (PWM) Pin 10 Valor 0
  Escribir digital Pin 4 ON
  Escribir digital Pin 5 OFF
  
```

```

para parar
  Escribir analógica (PWM) Pin 9 Valor 0
  Escribir digital Pin 2 OFF
  Escribir digital Pin 3 OFF
  Escribir analógica (PWM) Pin 10 Valor 0
  Escribir digital Pin 4 OFF
  Escribir digital Pin 5 OFF
  Esperar 500 milisegundos
  
```

```

para girar derecha
  Escribir analógica (PWM) Pin 9 Valor 0
  Escribir digital Pin 2 OFF
  Escribir digital Pin 3 ON
  Escribir analógica (PWM) Pin 10 Valor velocidad
  Escribir digital Pin 4 ON
  Escribir digital Pin 5 OFF
  
```

## P40 - Brazo robótico controlado desde PC (consola)

Un brazo robótico consiste en una serie de articulaciones mecánicas accionadas por motores que controlan el movimiento. Para la realización del proyecto se han utilizado 4 servos para el control de los movimientos de las articulaciones y 1 servo en el extremo del brazo para mover una pinza que abre y cierra con el objetivo de poder coger y soltar objetos.

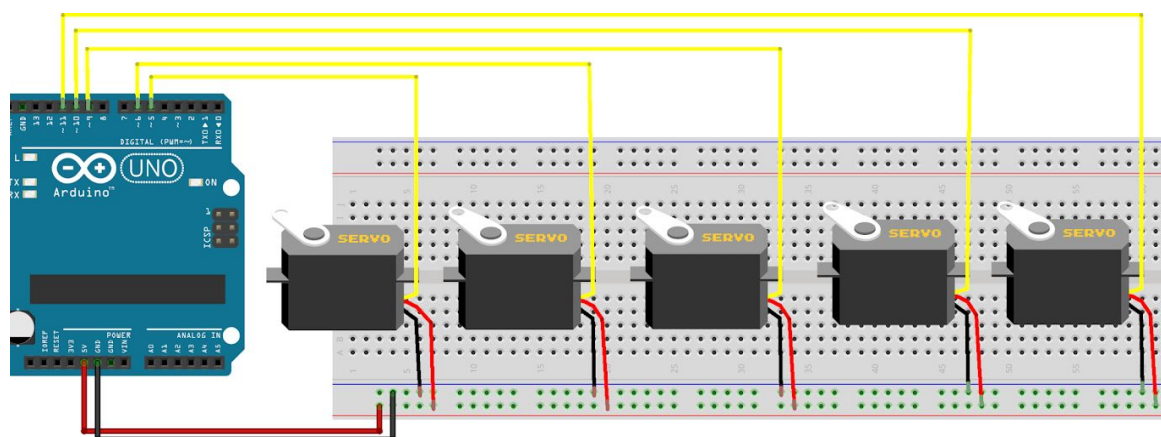
### Material necesario:

- 1 x Arduino UNO
- 1 x Sensor shield
- 5 x Servos
- 1 x Cables

### Conexiones:

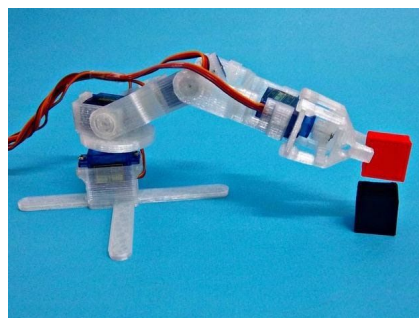
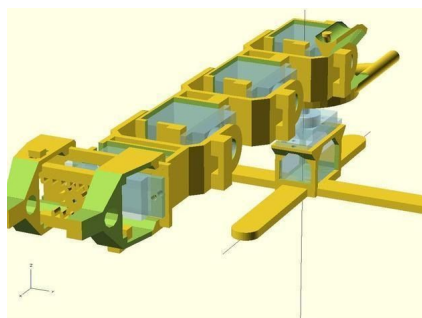
- Servo-1 = Pin 5
- Servo-2 = Pin 6
- Servo-3 = Pin 9
- Servo-4 = Pin 10
- Servo-Pinza= Pin 11

Esquema de conexiones:



El brazo robótico para probar este proyecto ha sido impreso en 3D. El modelo puedes encontrarlo compartido en el enlace:

<https://www.thingiverse.com/thing:65081>

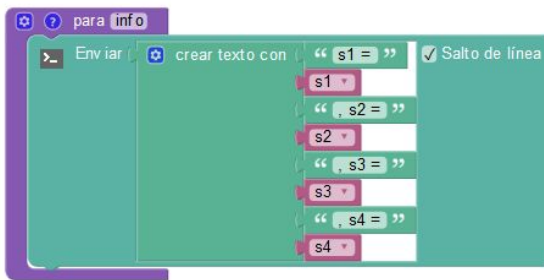


Programa ArduinoBlocks:

```

Inicializar
  Iniciar Baudios 9600
  Establecer rx = 0
  Establecer motorsel = 0
  reset
  info inicio

Bucle
  si ¿Datos recibidos?
  hacer
    Establecer rx = Recibir como número Hasta salto de línea
    si rx ≥ 201 y rx ≤ 204
    hacer
      Establecer motorsel = rx - 200
      Enviar crear texto con "motor = " Salto de línea motorsel
    sino si rx ≤ 180
    hacer
      si motorsel = 1
      hacer Establecer s1 = rx
      si motorsel = 2
      hacer Establecer s2 = rx
      si motorsel = 3
      hacer Establecer s3 = rx
      si motorsel = 4
      hacer Establecer s4 = rx
      info
    sino si rx = 301
    hacer abrir pinza
    sino si rx = 302
    hacer cerrar pinza
    sino si rx = 400
    hacer reset
  actualizar posicion
  
```



El control se realizará desde la consola serie desde un PC. Se puede utilizar directamente la consola de ArduinoBlocks.

