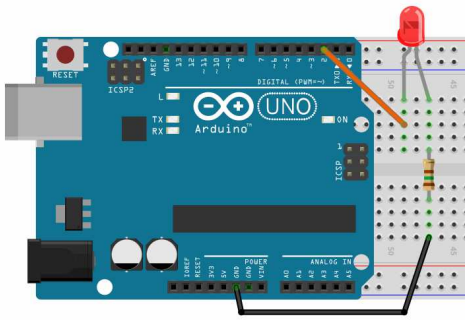
ACTIVIDADES ARDUINO

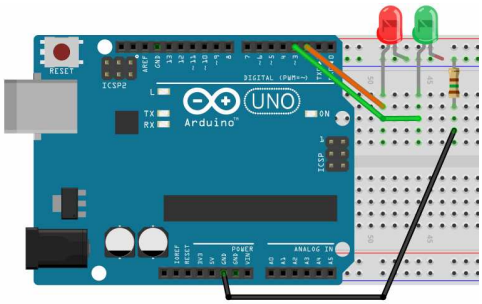
BLOQUE A: SIN ENTRADAS

A.0.- Ejemplo Resuelto. Programa para que un LED realice de forma indefinida una secuencia de 1 segundo encendido y 0,5 segundos apagado.

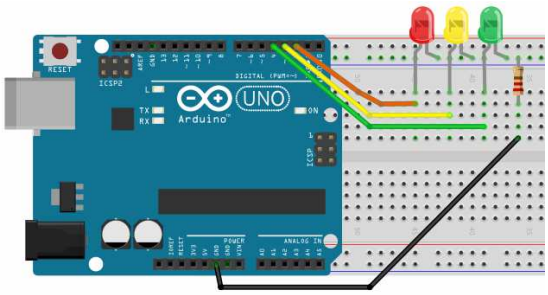
```
void setup() {
  pinMode(2,OUTPUT);
}

void loop() {
  digitalWrite(2,HIGH);
  delay(1000);
  digitalWrite(2,LOW);
  delay(500);
}
```



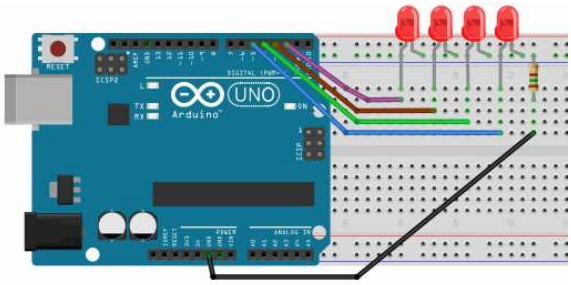
A.1.- Programa para que dos LED se enciendan de forma alterna (cuando el rojo está encendido el verde está apagado y viceversa) con una periodicidad de 1 segundo.



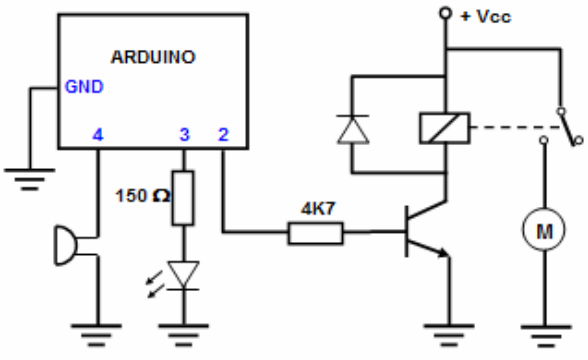
A.2.- Programa para que tres LED, rojo amarillo y verde, simulen el funcionamiento de un semáforo, de forma que el rojo esté encendido 5 segundos, el amarillo 1,5 segundos y el verde 4 segundos y así sucesivamente.

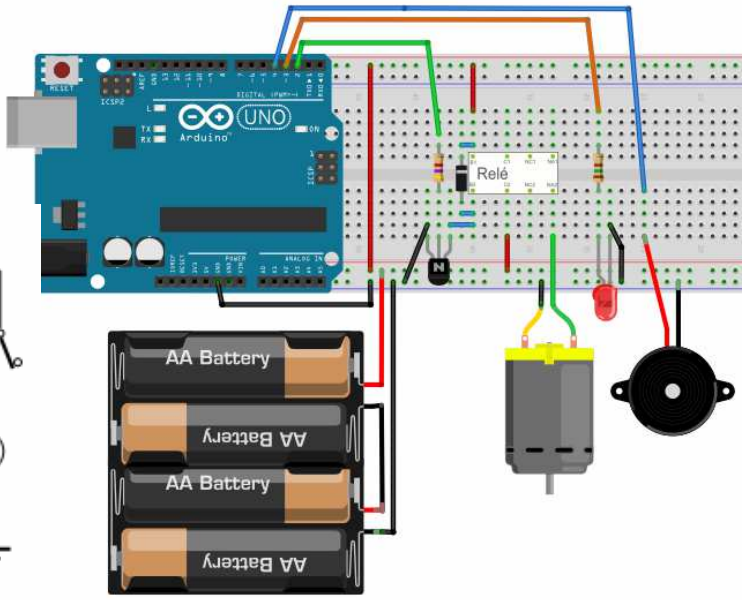


A.3.- Programa para que cuatro LEDs, se vayan encendiendo de forma sucesiva en un sentido y en sentido contrario (similar al “coche fantástico”) con una cadencia de 0,25 segundos aproximadamente.



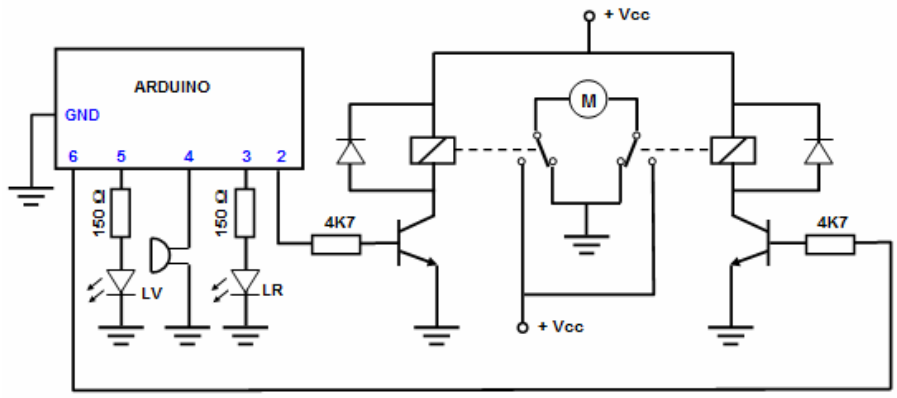
A.4.- Programa para que el sistema de la figura funcione del siguiente modo: el motor funcionará indefinidamente en ciclos de 4 s de marcha y 2 s de parada. Previamente al inicio de cada intervalo de marcha sonará durante 0,5 s el zumbador. El LED estará encendido durante el intervalo de marcha del motor.

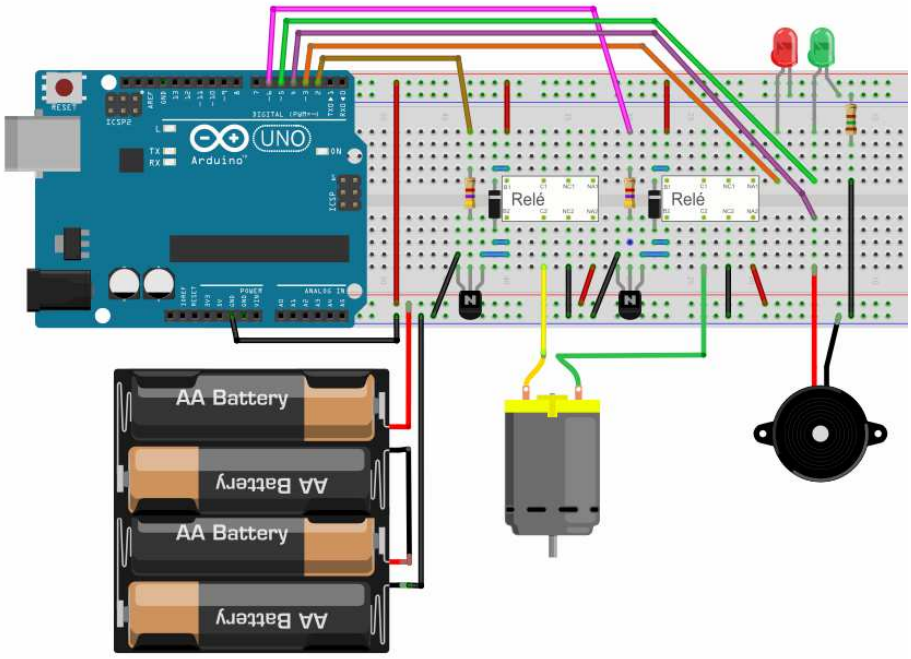




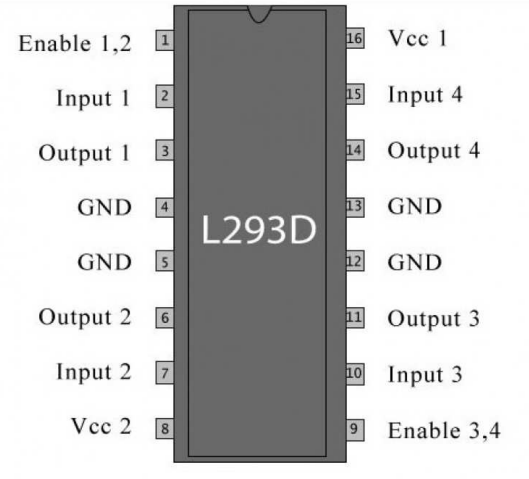
A.5.- Programa para que un motor funcione en ciclos de 4 s de marcha en un sentido, 2 s de parada, 4 s de marcha en sentido contrario, 2 s de parada y vuelta a empezar.

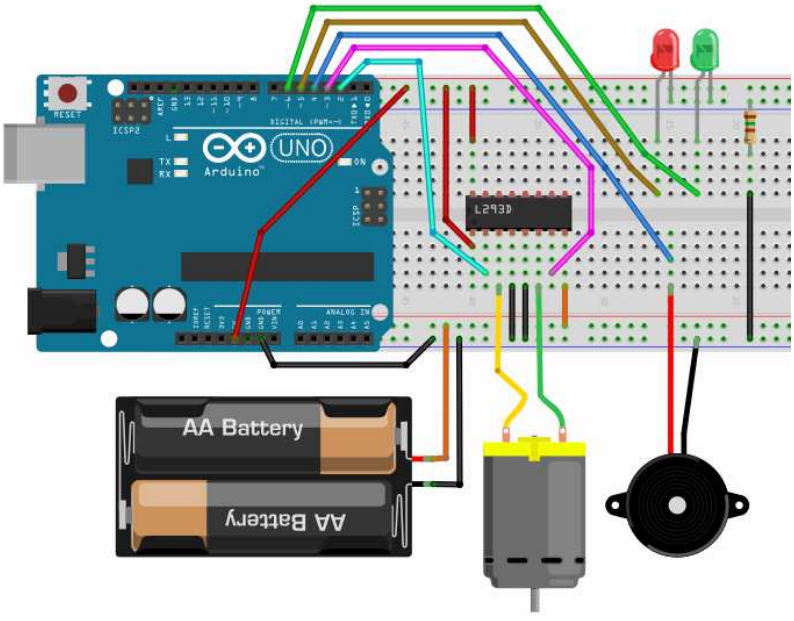
Previamente al inicio de cada intervalo de marcha sonará durante 0,5 s el zumbador. Cada LED estará encendido en un sentido de marcha.





Este mismo ejercicio puede realizarse con un integrado amplificador L293D. Ver figura:



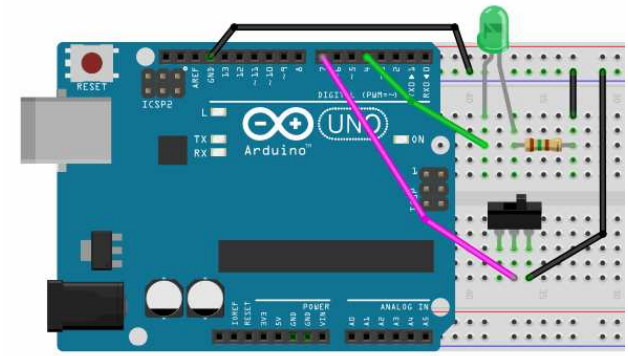


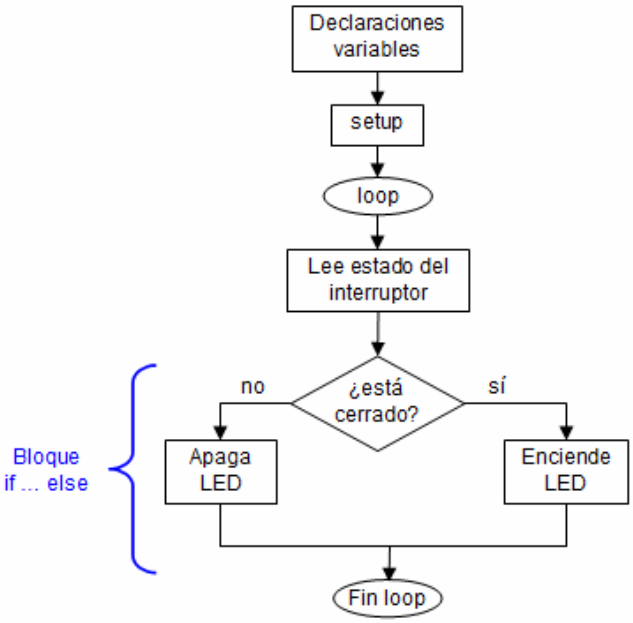
Vcc1 = 5 V (alimentación del integrado)

Vcc2 = Tensión aplicada en las salidas

BLOQUE B: DECISIONES if … else

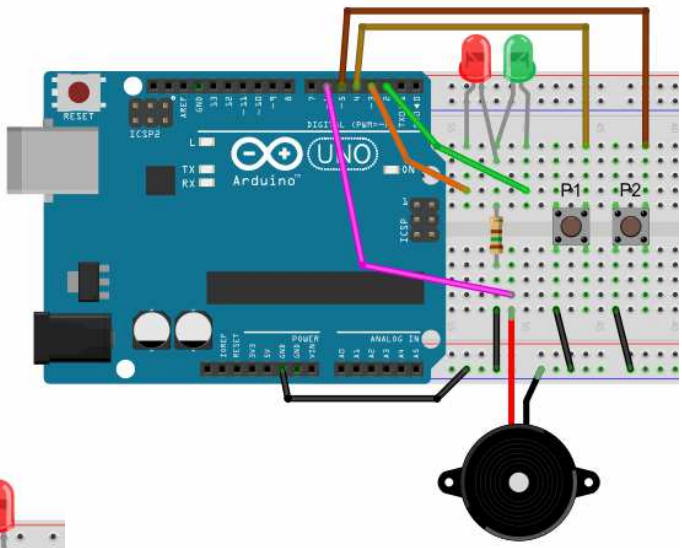
B.0.- Ejemplo resuelto que hace que un LED verde (LV) conectado en el pin 4 se encienda cuando un interruptor (E1) conectado al pin 7 está cerrado y el LED se apague cuando dicho interruptor está abierto. Edítalo, cárgalo en la tarjeta y comprueba que funciona.

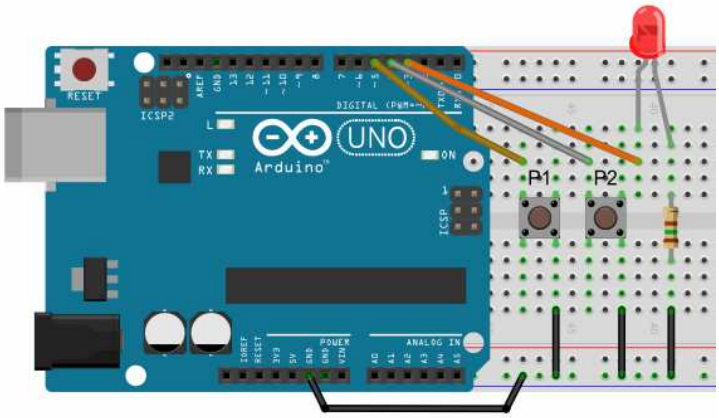





```
int pinLV=4;
int pinE1=7;
int estadointerruptor;

void setup() {
  pinMode(pinLV, OUTPUT);
  pinMode(pinE1, INPUT_PULLUP);
}
void loop() {
  estadointerruptor=digitalRead(pinE1);
  if(estadointerruptor==LOW){
    digitalWrite(pinLV, HIGH);
  }
  else{
    digitalWrite(pinLV, LOW);
  }
}
```

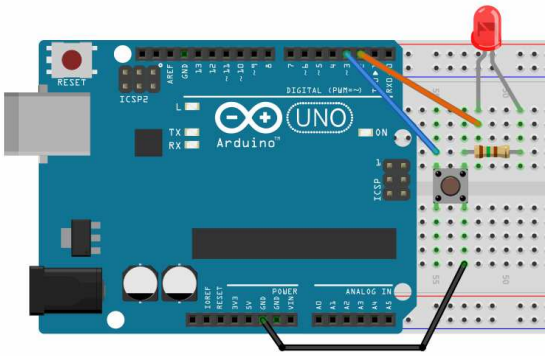
B.1.- Programa para que un LED verde permanezca encendido y un LED rojo permanezca apagado mientras se pulsa un pulsador P1 y cambien su estado cuando se deje de pulsar. Igualmente, un zumbador pitará mientras se pulse un pulsador P2 y callará cuando no esté pulsado. Las entradas correspondientes a los pulsadores serán del tipo INPUT_PULLUP.



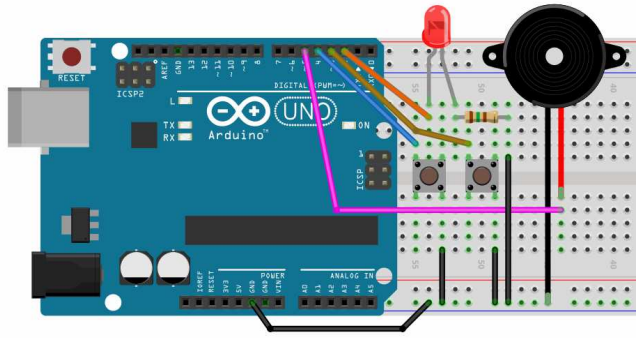


B.2.- Programa para que un LED se encienda cuando se pulse un pulsador P1 y se apague cuando se pulse otro pulsador P2. Si los dos pulsadores se pulsan simultáneamente el LED estará apagado. Las entradas correspondientes a los pulsadores serán del tipo INPUT_PULLUP.

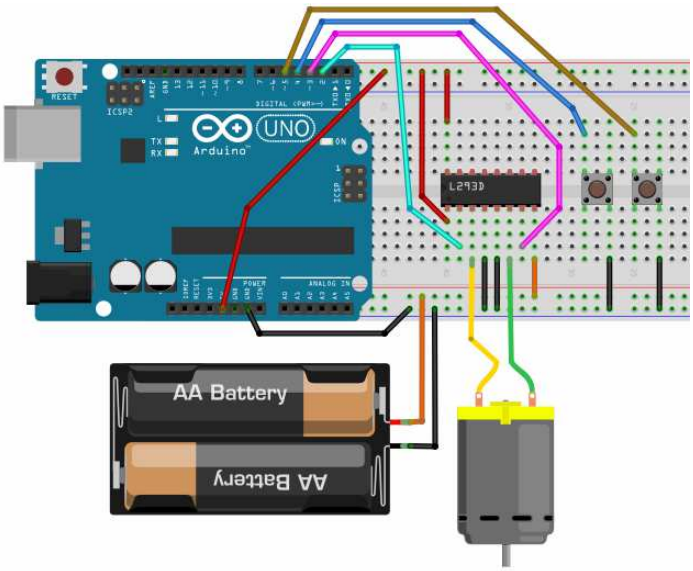
B.3.- Programa para que un LED se encienda al pulsar un pulsador y se apague cuando se vuelva a pulsar el mismo pulsador. Si se mantiene pulsado el pulsador de forma permanente, el LED se quedará estable en el último estado adquirido. La entrada correspondiente al pulsador será del tipo INPUT_PULLUP.

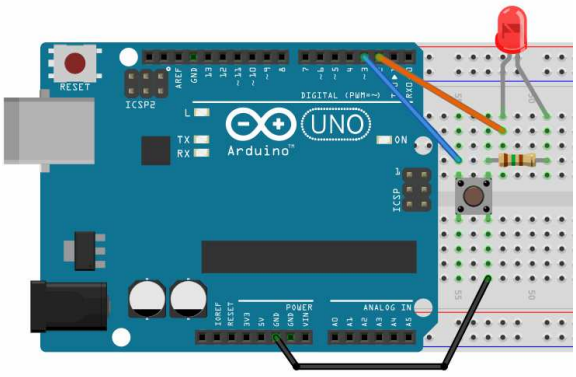


B.4.- Programa para que cada vez que se pulsa un pulsador P1 un LED cambia de estado y que cada vez que se pulsa otro pulsador P2 un zumbador cambie de estado. Las entradas correspondientes a los pulsadores serán del tipo INPUT_PULLUP.



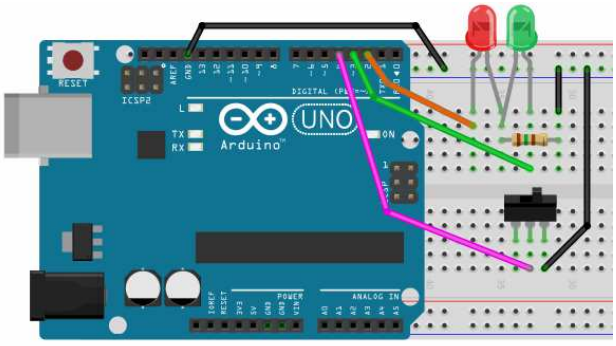
B.5.- Programa para controlar un motor con dos pulsadores. Con el pulsador P1 se controlará la marcha-paro del motor, y con el pulsador P2 el sentido de giro. Cada vez que se pulsa P1 el motor pasa de parado a funcionando o a la inversa. Cada vez que se pulsa P2 cambia de sentido de marcha si está funcionando. No será necesario mantener pulsados los pulsadores. Las entradas correspondientes a los pulsadores serán del tipo INPUT_PULLUP.





B.6.- Programa para que un LED se encienda cuando se pulse 3 veces un pulsador, y se apague cuando se pulse 2 veces el mismo pulsador, y así sucesivamente. La entrada correspondiente al pulsador será del tipo INPUT_PULLUP.

B.7.- Programa que controle dos LED que se encienden de forma alternativa con una cadencia de 2 segundos cuando un interruptor está abierto y que pasa a 0,5 segundos cuando está cerrado. La entrada correspondiente al interruptor será del tipo INPUT_PULLUP.

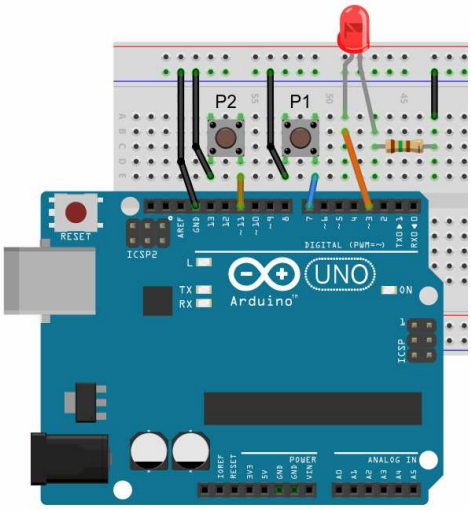


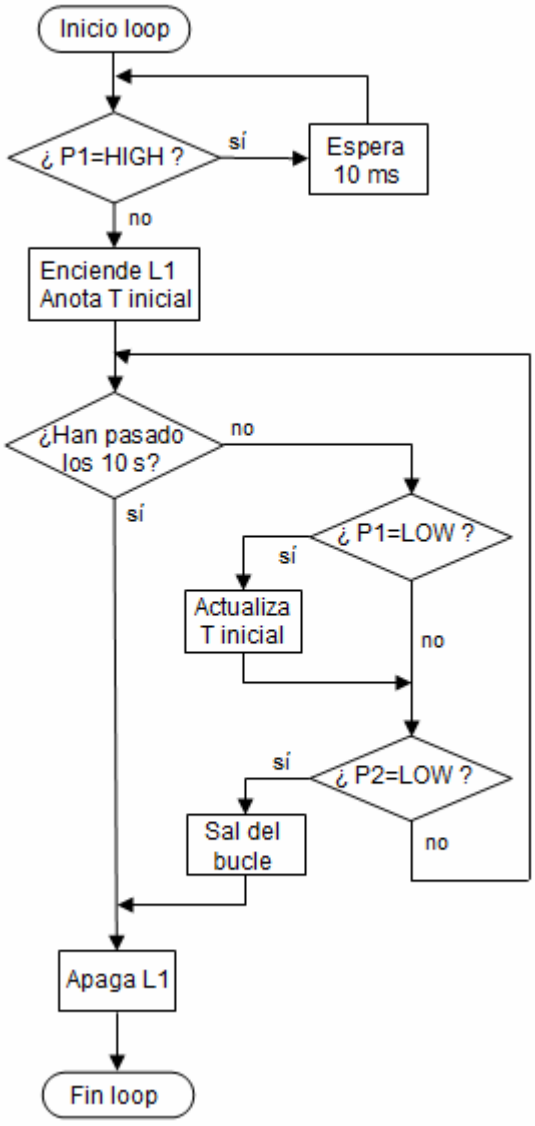
BLOQUE C: DECISIONES while(). EL USO DE break Y DE millis()

C.0.- Ejemplo resuelto para ilustrar el uso de las funciones while(), millis() y break, así como el uso de los diagramas de flujo: Tenemos un LED, L1, y dos pulsadores, P1 y P2. Al pulsar P1 el LED se enciende. Una vez encendido, al cabo de 10 s el LED se apaga solo, pero si antes de que transcurran los 10 segundos pulsamos P2 se apaga el LED. Si antes de pasar los 10 segundos se pulsa P1 vuelve a empezar a contar el tiempo. Edítalo, carga el programa en la tarjeta y comprueba que funciona. Asegúrate de comprender lo que hace el programa antes de seguir.

Conexiones: L1 en pin 3, P1 en pin 7 y P2 en pin 11.

```
int pinL1=3;
int pinP1=7;
int pinP2=11;
unsigned long tini; //variable para anotar
                    // el tiempo de inicio
                    // de la temporización

void setup() {
   pinMode(pinL1, OUTPUT);
   pinMode(pinP1, INPUT_PULLUP);
   pinMode(pinP2, INPUT_PULLUP);
   digitalWrite(pinL1,LOW); //Empezamos con
                            //el LED apagado
}
void loop() {
  while(digitalRead(pinP1)==HIGH) delay(10);
  digitalWrite(pinL1,HIGH);
  tini=millis();
  while(millis()-tini<10000){
    if(digitalRead(pinP1)==LOW) tini=millis();
    if(digitalRead(pinP2)==LOW) break;
  }
  digitalWrite(pinL1,LOW);
}
```





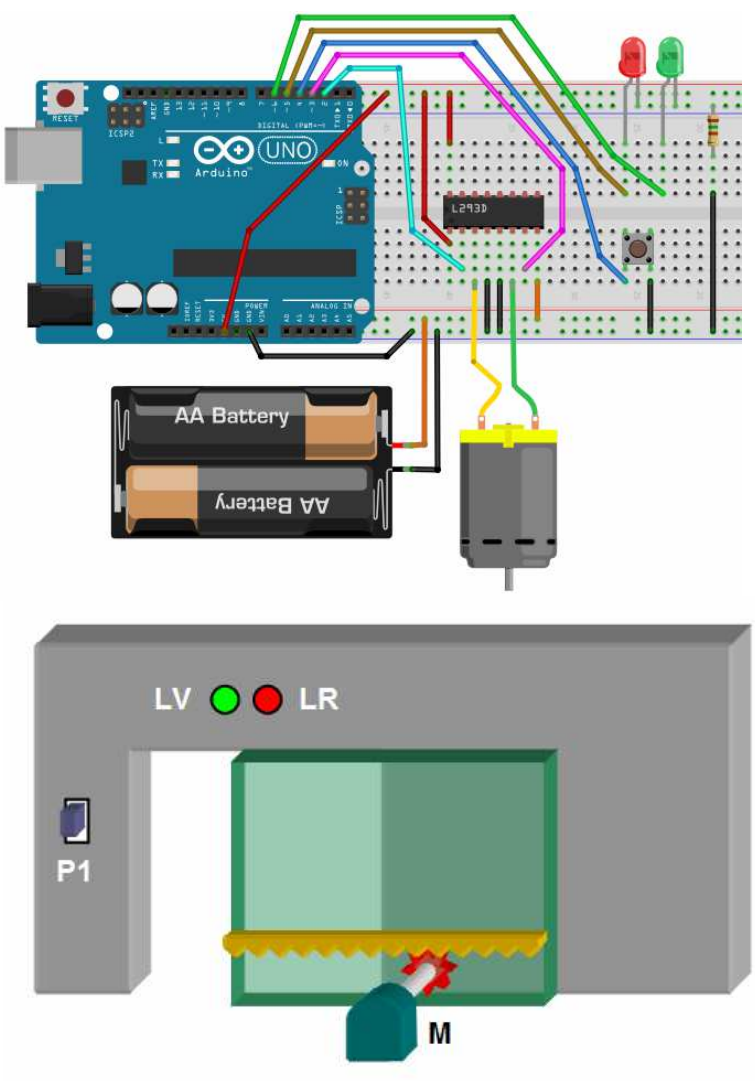
C.1.- Programa que controla un LED que se enciende con una cadencia inicial de 2 segundos. Cada vez que se pulse un pulsador P1 la cadencia se reduce en 0,25 segundos y así hasta llegar a un mínimo de 0,25 segundos. Cada vez que se pulse otro pulsador P2 la cadencia aumenta en 0,25 segundos y así hasta llegar a un máximo de 4 segundos.

Nota: en este problema no se podrán usar delay() para controlar los intervalos de encendido y apagado pues si la pulsación se produce durante la ejecución de delay() siendo más breve que la duración de dicha instrucción, el microprocesador no se entera de la pulsación.

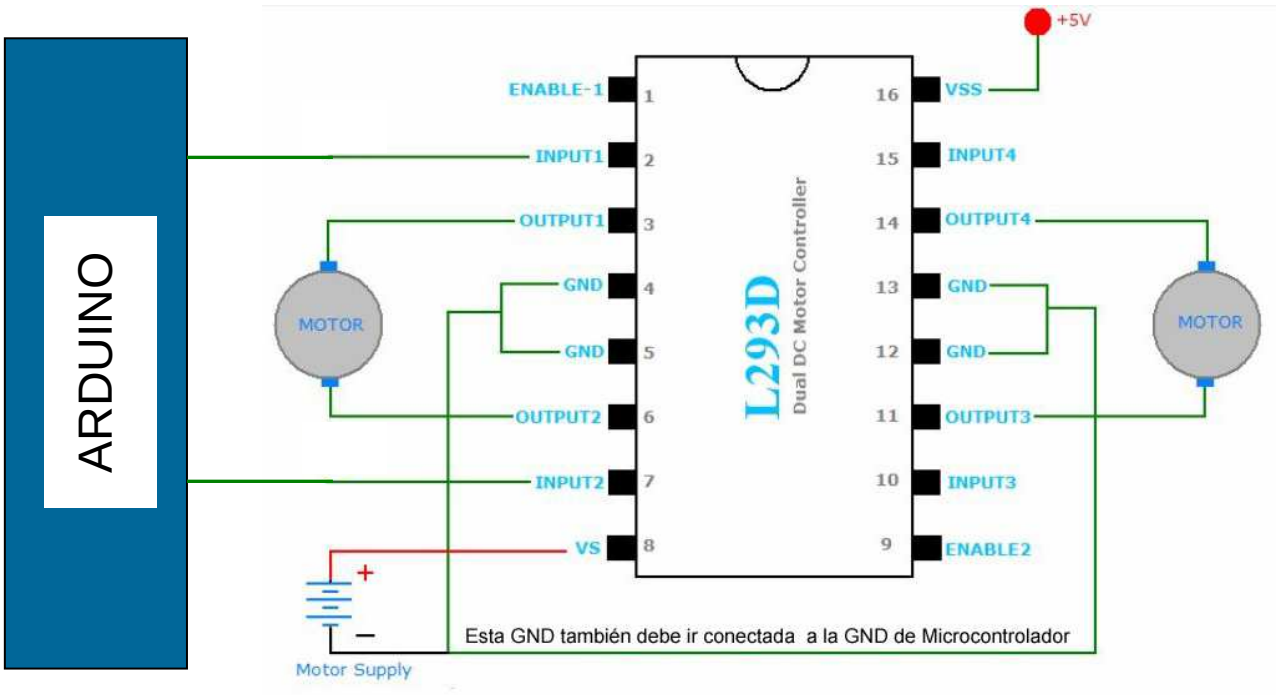
C.2.- Realiza el programa y el diagrama de flujo para resolver el siguiente problema:

Tenemos un motor conectado en los pines 2 y 6, un LED verde (LV) conectado en el pin 5, un LED rojo (LR) conectado al pin 3, un pulsador P1 conectado en el pin 4.

El motor mueve una puerta de garaje motorizada.



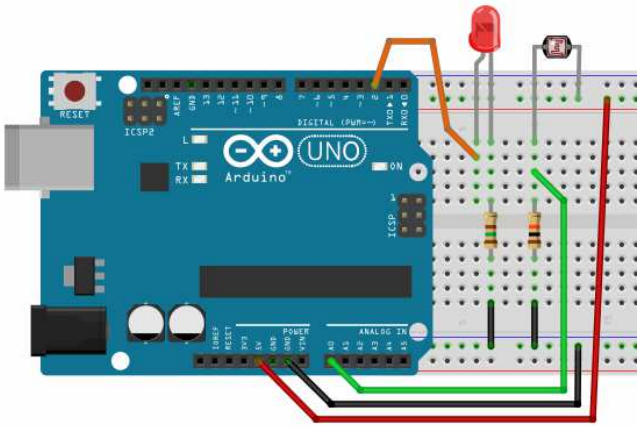
Inicialmente el motor estará parado y los LEDs estarán apagados. Al pulsar P1 (pulsador para ordenar la apertura de la puerta) se pondrá en marcha el motor en el sentido de las agujas del reloj (apertura de puerta). A los 5 segundos (tiempo necesario para abrir la puerta) se parará el motor (puerta abierta) y se encenderá LV (coche puede pasar). Al cabo de 6 segundos (tiempo de espera para pasar el coche) el motor se pondrá a girar en sentido contrario a las agujas del reloj otros 5 segundos (tiempo para cerrar la puerta), durante los cuales se encenderá LR (peligro, puerta cerrándose). Al cabo de dichos 5 segundos el motor se parará (puerta cerrada) y LR se apagará y el sistema queda a la espera de una nueva pulsación de P1.



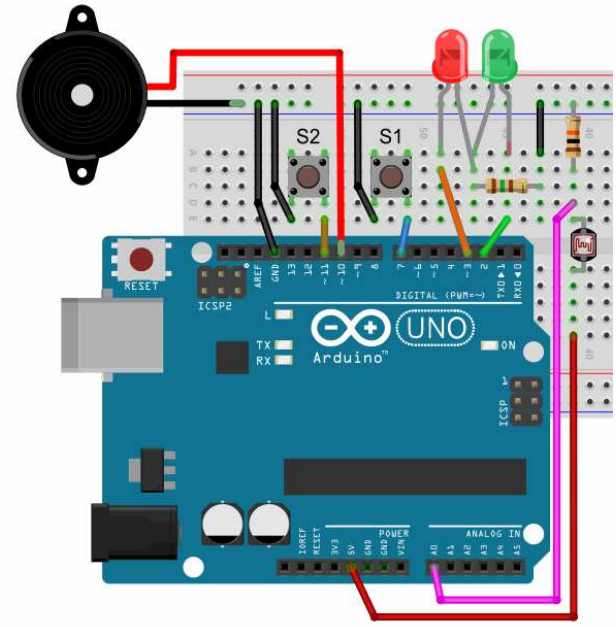
BLOQUE D: LAS ENTRADAS ANALÓGICAS Y LAS SALIDAS PSEUDOANALÓGICAS

D.0.1.- Ejemplo resuelto. Disponemos de una LDR de modo que cuando incide suficiente luz sobre ella un LED estará apagado. Cuando el nivel de luz incidente disminuye por debajo de un cierto valor umbral el LED se encenderá. La resistencia que forma el divisor de tensión con la LDR será de 10 K. Cuanto más luz incida sobre la LDR menos resistencia tendrá y el valor leído en la entrada A0 será mayor.

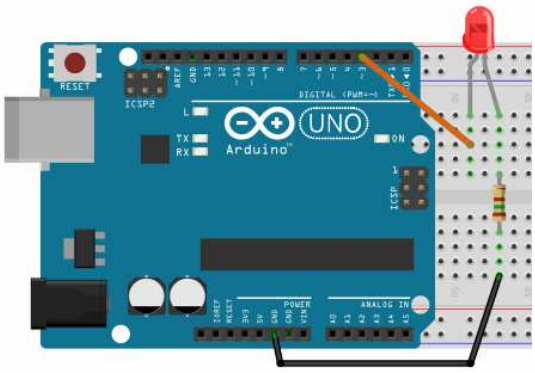
```
int LDR=A0;
int L1=2;
int umbral=500;

void setup() {
  pinMode(LDR, INPUT);//Aunque no es necesario
  pinMode(L1, OUTPUT);
}

void loop() {
  if(analogRead(LDR)>umbral)digitalWrite(L1,LOW);
  else digitalWrite(L1, HIGH);
}
```



D.1.- Programa que controla un sistema de alarma que dispone de una LDR, dos pulsadores (S1 y S2) en montaje pull-up, dos LED (LR y LV) y un zumbador. La alarma pasa del estado de desconectada a conectada pulsando S1. Cuando está desconectada está encendido LV y cuando está conectada lo está LR. Si estando conectada se oscurece la LDR la alarma salta haciendo sonar el zumbador, el cual no se calla aunque se vuelva a iluminar la LDR. Para desconectar la alarma o para desactivarla una vez ha saltado, se pulsará S2, con lo cual la alarma pasa a estado de desconectada.



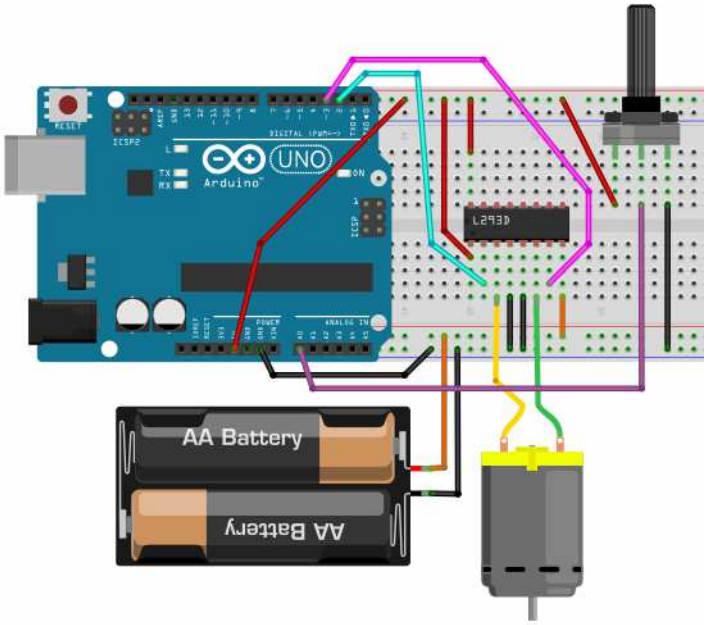
D.0.2.- Ejemplo resuelto. Variación gradual de la intensidad de un LED conectado a una salida PWM (~). Nota, las salidas 3, 5, 6, 9, 10, y 11 son de tipo PWM en la tarjeta Arduino 1.



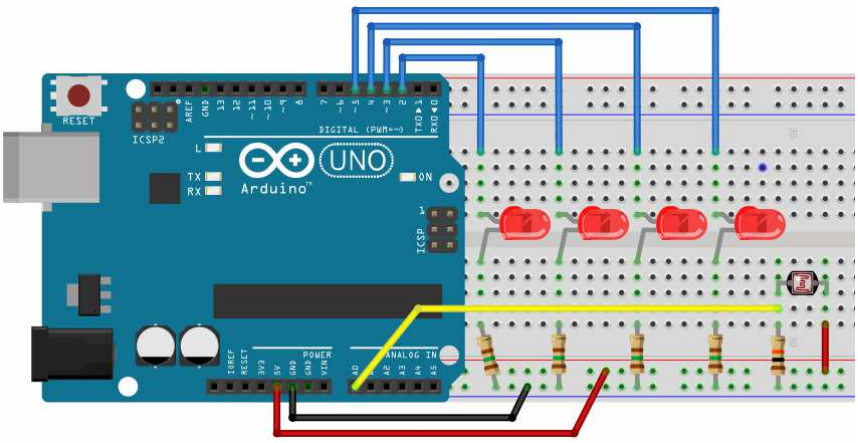
```
const int L1=3; //pin tipo PWM
void setup() {
  pinMode(L1, OUTPUT);
}
void loop() {
  for(int i=0;i<=255;i++){
    analogWrite(L1,i);
    delay(10);
  }
  for(int i=255;i>=0;i--){
    analogWrite(L1,i);
    delay(10);
  }
}
```

D.0.3.- Ejemplo resuelto. Variación de la velocidad de un motor mediante un potenciómetro. Lo haremos aplicando una señal PWM a la patilla de Enable del integrado L293D (patilla 1). El valor sacado por la salida PWM dependerá de la lectura tomada en la entrada analógica conectada al potenciómetro.

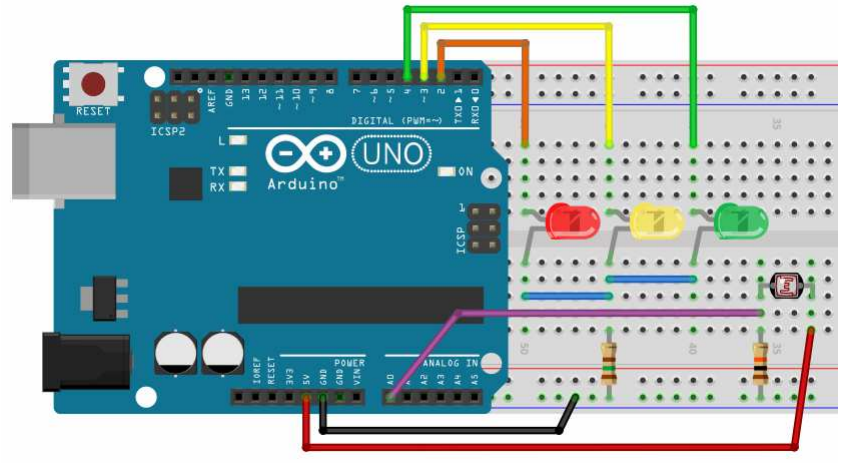
```
const int M=2;
const int pinpoten=A0; //pin analógico
const int Enable=3; //pin tipo PWM
int valorpoten;
int pwm1;

void setup() {
  pinMode(M, OUTPUT);
  pinMode(Enable, OUTPUT);
  digitalWrite(M, HIGH);
}
void loop() {
  valorpoten=analogRead(pinpoten);
  pwm1=map(valorpoten, 0, 1023, 0, 255);
  analogWrite(Enable, pwm1);
}
```



D.2.- Elabora un programa para que en función del nivel de iluminación que incida sobre una LDR se ilumine un mayor o menor número de LEDs. Se dispondrá de 4 LED; con un nivel alto de iluminación no se encenderá ninguno. Con una oscuridad total se encenderán los cuatro. Para niveles intermedios se encenderán uno, dos o tres. Los niveles de luz los establecerás tú.

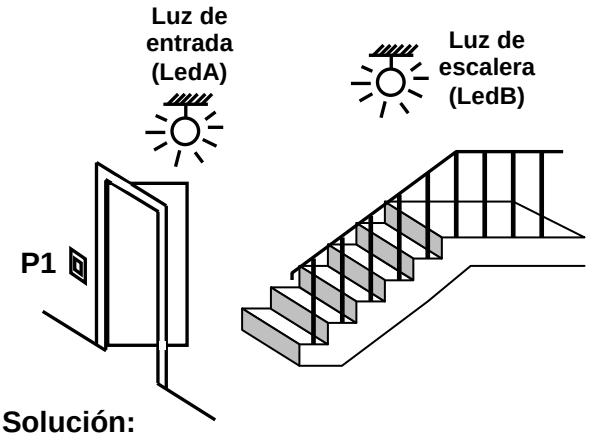


D.3.- Elabora un programa para que tres LEDs se enciendan sucesivamente (rojo, amarillo, verde,...) con una cadencia de 1 segundo cada uno cuando la LDR está totalmente destapada. Conforme se va aumentando la sombra sobre la LDR, la cadencia se hará más rápida (los intervalos de encendido y apagado de los LEDs serán más pequeños).



BLOQUE E: EL MÉTODO DE LA DEFINICIÓN DE ESTADOS Y USO DE SWITCH…CASE

E.0.1.- **Ejemplo resuelto** para ilustrar el método de la definición de estados**:** Programa de control del apagado automático de las luces de escalera de un bloque de viviendas. Al pulsar un pulsador P1, dispuesto en montaje pull-up, se encenderán dos LEDs (simulan a las luces). El LED A (que se supone que simula a la luz de la entrada del bloque) se apagará a los 5 s y el LED B (que se supone que simula a las luces de las escaleras) se apagará a los 15 s. Si antes de transcurrir estos tiempos se volviera a pulsar el pulsador el cómputo de tiempo se reanudaría.



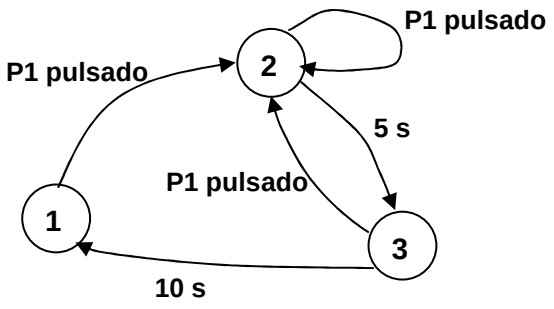
Solución:

Asignación de estados:

- Estado 1: LedA = LedB = LOW; al pulsar P1 pasa a estado 2
- Estado 2: LedA = LedB = HIGH; a los 5 s pasa a estado 3. Si antes se pulsa P1 se empieza a contar de nuevo.
- Estado 3; LedA = LOW, LedB = HIGH; a los 10 s pasa a estado 1. Si antes se pulsa P1 pasa al estado 2.

Asignación de estado inicial:

Si P1 = HIGH, asigno estado 1, si P1 = LOW estado 2.




```
const int LA=2;
const int LB=3;
const int P1=4;
int estadoescalera;
unsigned long tini;

void setup() {
  pinMode(LA, OUTPUT);
  pinMode(LB, OUTPUT);
  pinMode(P1, INPUT_PULLUP);
  if(digitalRead(P1)==HIGH)estadoescalera=1;
  else estadoescalera=2;
}

void loop() {
  controlaescalera();
}

void controlaescalera(){
  switch (estadoescalera){
    case 1:
      digitalWrite(LA,LOW);
      digitalWrite(LB,LOW);
      if(digitalRead(P1)==LOW){
          estadoescalera=2;
          tini=millis();
      }
      break;
    case 2:
      digitalWrite(LA,HIGH);
      digitalWrite(LB, HIGH);
      if(millis()-tini>5000){
          estadoescalera=3;
          tini=millis();
      }
      if(digitalRead(P1)==LOW){
          tini=millis();
      }
      break;
    case 3:
      digitalWrite(LA,LOW);
      digitalWrite(LB, HIGH);
      if(millis()-tini>10000){
          estadoescalera=1;
      }
      if(digitalRead(P1)==LOW){
          estadoescalera=2;
          tini=millis();
      }
      break;
  }
}
```

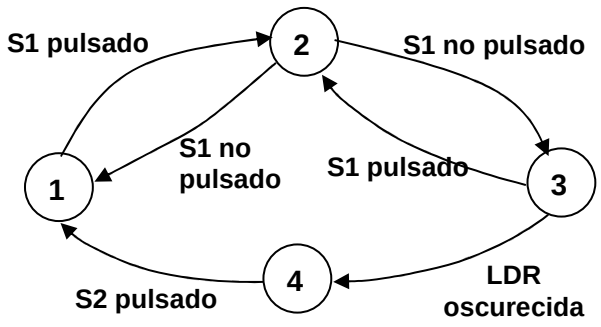
E.0.2.- Ejemplo resuelto por el método de la definición de estados de un sistema de alarma formado por una LDR, dos LED (LV y LR), dos pulsadores (S1 y S2) en montaje pull-up y un zumbador. La alarma pasa de conectada a desconectada y viceversa pulsando S1 (tener en cuenta que por breve que sea la pulsación, el programa tiene tiempo de ejecutarse innumerables veces). Estando conectada está encendido LR y cuando está desconectada lo está LV. Si estando conectada se oscurece la LDR la alarma salta haciendo sonar el zumbador, el cual no se calla aunque se vuelva a iluminar la LDR. Para acallar el zumbador se pulsará S2, con lo cual se calla el zumbador y la alarma pasa a estado de desconectada (o sea, una vez ha saltado la alarma no la desconecta S1 sino S2).

Solución E.0.2 : primera variante

Asignación de estados:

- Estado 1 alarma desconectada: LV= HIGH, LR= LOW; al pulsar S1 pasa a estado 2
- Estado 2: estado transitorio mientras pulsador S1 permanece pulsado; al dejar de pulsar S1 pasa a estado 3 si el estado anterior era el estado 1 y pasa al estado 1 si el estado anterior era el 3.
- Estado 3 alarma conectada; LV = LOW, LR = HIGH; si se pulsa S1 vuelve al estado 2 y si se oscurece la LDR pasa al estado 4.
- Estado 4 alarma activada; LV = LOW, LR = HIGH, Zumbador pitando. Si se pulsa S2 pasa al estado 1

Asignación de estado inicial: estado 1




```
const int LV=2;
const int LR=3;
const int S1=7;
const int S2=11;
const int LDR=A0;
const int Zumb=10;
int estadoalarma;
int estadoanterior=1;

void setup() {
  pinMode(LV, OUTPUT);
  pinMode(LR, OUTPUT);
  pinMode(Zumb, OUTPUT);
  pinMode(S1, INPUT_PULLUP);
  pinMode(S2, INPUT_PULLUP);
  estadoalarma=1;
}

void loop() {
  controla_alarma();
  delay(10);
}

void controla_alarma(){
  switch (estadoalarma){
    case 1:
      digitalWrite(LV,HIGH);
      digitalWrite(LR,LOW);
      digitalWrite(Zumb, LOW);
      if(digitalRead(S1)==LOW){
          estadoalarma=2;
          estadoanterior=1;
      }
      break;
    case 2:
      if(digitalRead(S1)==HIGH && estadoanterior==1)
        estadoalarma=3;
      if(digitalRead(S1)==HIGH && estadoanterior==3)
        estadoalarma=1;
      break;
    case 3:
      digitalWrite(LV,LOW);
      digitalWrite(LR, HIGH);
      if(digitalRead(S1)==LOW){
          estadoalarma=2;
          estadoanterior=3;
      }
      if(analogRead(LDR)<500)estadoalarma=4;
      break;
    case 4:
      digitalWrite(Zumb, HIGH);
      if(digitalRead(S2)==LOW)estadoalarma=1;
      break;
  }
}
```

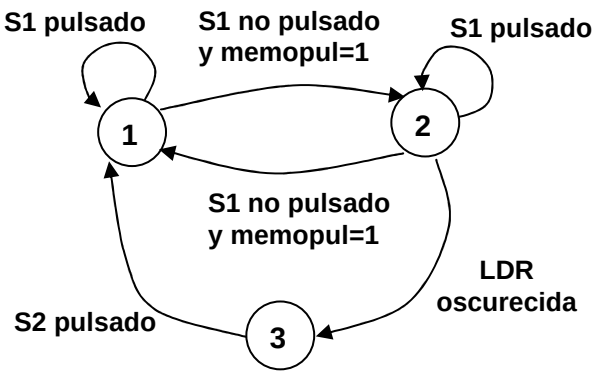
Solución E.0.2: segunda variante

En este caso, en vez de definir un estado intermedio en el que se mantiene el sistema mientras está P1 pulsado, lo que hacemos es utilizar una variable que se pone a 1 cuando se pulsa P1 de modo que el cambio efectivo de estado no se produce hasta que P1 deja de estar pulsado y dicha variable está en estado 1. O sea, esta variable digamos que memoriza que se ha pulsado P1. La llamaremos “memopul”

Asignación de estados:

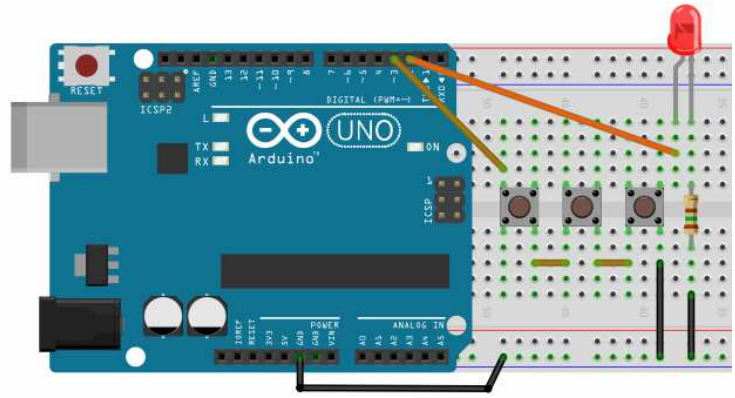
- Estado 1 alarma desconectada: LV= HIGH, LR= LOW; al pulsar S1 la variable “memopul” pasa a valer 1. Cuando P1 deja de estar pulsado pasa a estado 2 y “memopul” vuelve a 0.
- Estado 2 alarma conectada; LV = LOW, LR = HIGH; la variable “memopul” pasa a valer 1. Cuando P1 deja de estar pulsado pasa a estado 1 y “memopul” vuelve a 0. Si se oscurece la LDR pasa al estado 3.
- Estado 3 alarma activada; LV = LOW, LR = HIGH, Zumbador pitando. Si se pulsa S2 pasa al estado 1.

Asignación de estado inicial: estado 1

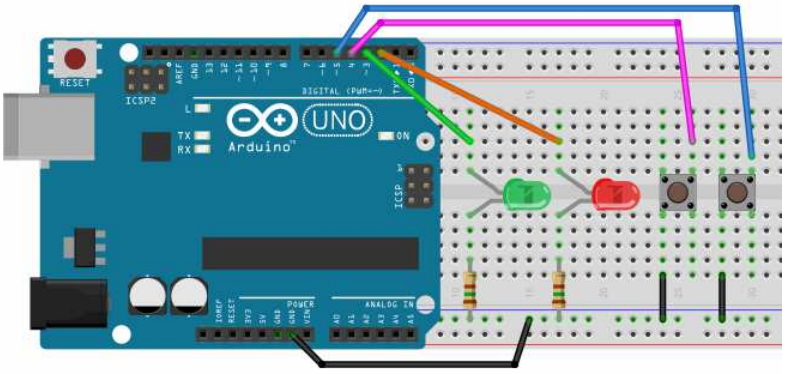



```
const int LV=2;
const int LR=3;
const int S1=7;
const int S2=11;
const int LDR=A0;
const int Zumb=10;
int estadoalarma;
int memopul=0;

void setup() {
  pinMode(LV, OUTPUT);
  pinMode(LR, OUTPUT);
  pinMode(Zumb, OUTPUT);
  pinMode(S1, INPUT_PULLUP);
  pinMode(S2, INPUT_PULLUP);
  estadoalarma=1;
}

void loop() {
  controla_alarma();
  delay(10);
}
void controla_alarma(){
  switch (estadoalarma){
    case 1:
      digitalWrite(LV,HIGH);
      digitalWrite(LR,LOW);
      digitalWrite(Zumb, LOW);
      if(digitalRead(S1)==LOW)memopul=1;
      else if((digitalRead(S1)==HIGH)&&memopul==1){
        estadoalarma=2;
        memopul=0;
      }
      break;
    case 2:
      digitalWrite(LV,LOW);
      digitalWrite(LR, HIGH);
      if(digitalRead(S1)==LOW)memopul=1;
      else if((digitalRead(S1)==HIGH)&&memopul==1){
        estadoalarma=1;
        memopul=0;
      }
      if(analogRead(LDR)<500)estadoalarma=3;
      break;
    case 3:
      digitalWrite(Zumb, HIGH);
      if(digitalRead(S2)==LOW)estadoalarma=1;
      break;
  }
}
```

E.1.- Utilizando el método de la definición de estados, elabora un programa para el control del encendido y apagado automático del alumbrado de escalera de un bloque de viviendas que funcionará del siguiente modo: cada puerta de vivienda, mientras está cerrada, mantiene pulsado y cerrado un pulsador. Al abrir cualquiera de las puertas se deja de pulsar y se abre su pulsador correspondiente, haciendo que se encienda un LED que simula el alumbrado de la escalera. El LED se mantendrá encendido en tanto que alguna puerta esté abierta. Una vez cerradas todas las puertas, el LED se mantendrá encendido durante 10 segundos más. Si antes de terminar el tiempo de espera de 10 segundos se vuelve a abrir alguna puerta se reinicia de nuevo el ciclo. En nuestro caso, supondremos que sólo son tres viviendas.

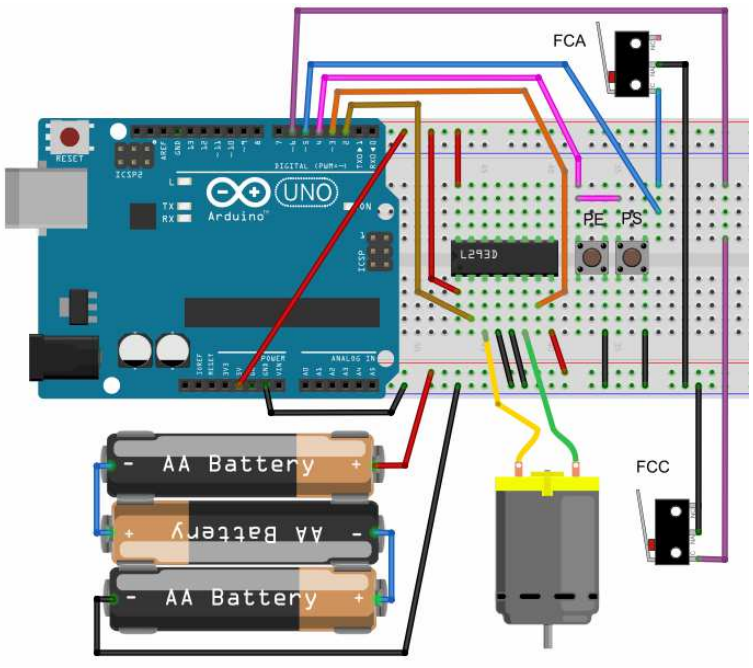


E.2.- a) Elaborar el programa de control de un sistema formado por dos LEDs, LR y LV, y dos pulsadores, P1 y P2. Al pulsar P1 se enciende LR, al pulsar P2 se enciende LV y se apagan ambos al cabo de 6 segundos de la segunda pulsación. Si antes de pulsar el P1 se pulsa el P2 no ocurre nada. Para simplificar, lo haremos de forma que si se vuelve a pulsar el P1 antes de que se hayan apagado los LEDs no se le hace caso.



b) Resolver el mismo caso anterior pero teniendo en cuenta que si antes de terminar el tiempo de espera de 6 s, se vuelve a pulsar el pulsador P1, se inicia de nuevo el ciclo, es decir, se enciende sólo LR y queda a la espera de que se pulse P2 para encender LV.

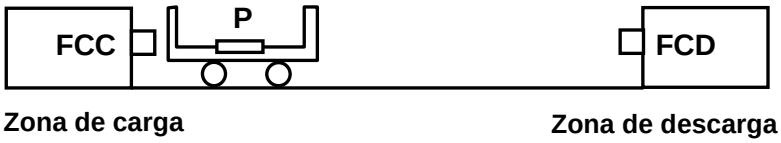
E.3.- Elabora, utilizando el método de la definición de estados, el programa de control de una puerta de garaje por medio de un motor que gira en ambos sentidos. La puerta es de entrada y salida. Tendrá un pulsador a la entrada, PE, y otro a la salida, PS. Al pulsar cualquiera de los pulsadores la puerta se abrirá hasta pisar el final de carrera de “puerta abierta”, FCA y al cabo de 10 s abierta se cerrará automáticamente hasta pisar el final de carrera de “puerta cerrada”, FCC.

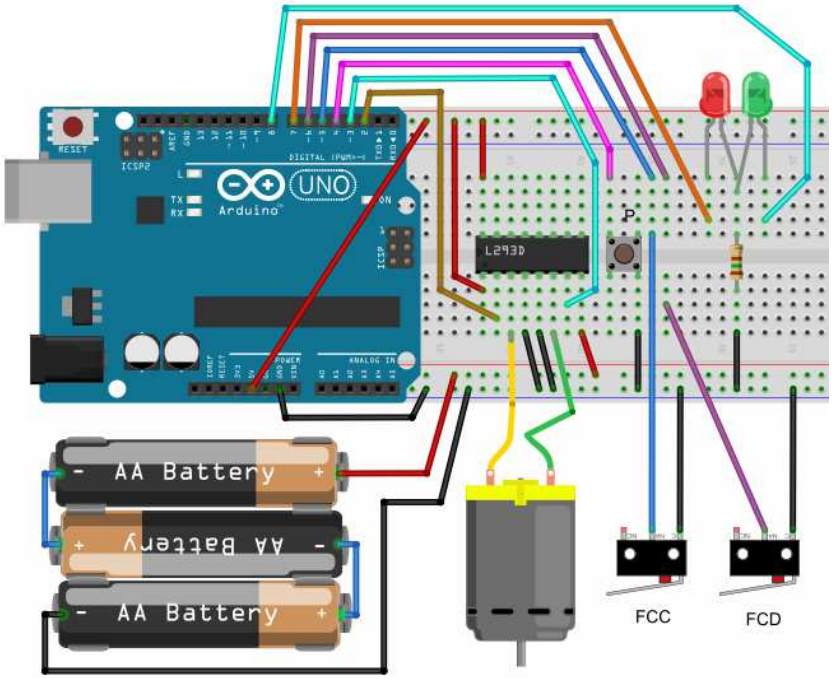


E.4.- Realizar una variante con respecto al problema anterior añadiendo una LDR que sea tapada por el coche mientras está pasando justo por la puerta. En este caso la puerta no debe empezar a cerrarse aunque hayan pasado los 10 s (para evitar golpear un coche que, por ejemplo, se haya calado justo al cruzar). También, en caso de que se tape la LDR una vez la puerta ha comenzado a cerrar, debe ponerse a abrir (lo mismo que si se hubiese pulsado PE o PS) y permanecer abierta hasta que se destape la LDR. La LDR se conectará en el pin analógico 0 (A0).

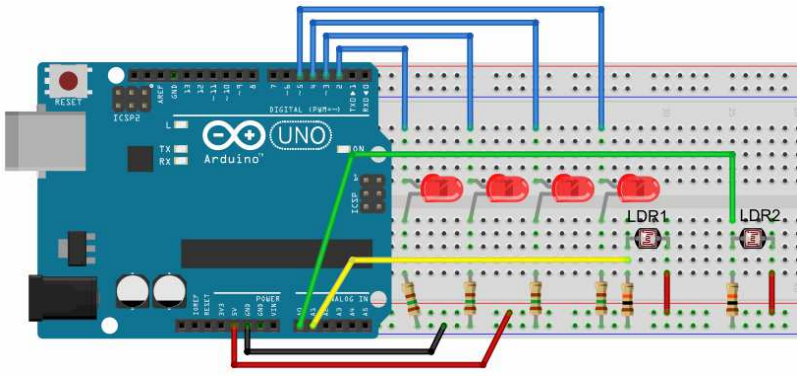
E.5.- Elaborar el programa de control de una carretilla automática que lleva mercancías de un lugar a otro en una fábrica. El funcionamiento será como sigue:

La carretilla inicia su recorrido en la zona de carga (posición que es detectada por el final de carrera FCC). La carretilla dispone de un sensor (P) que se activa al ponerle peso encima (lo simularemos por un pulsador). Cuando se pone peso encima espera 5 seg. (para dar tiempo al operario que deposita la carga a retirarse de la carretilla para no ser arrollado) y emprende el camino hacia la zona de descarga (posición que es detectada por el final de carrera FCD) donde se para al llegar. Cuando un operario le retira la carga, vuelve a esperar 5 seg. y emprende el camino de regreso hacia la zona de carga. Como medida de seguridad, cuando la carretilla esté en movimiento se encenderá un juego de pequeñas luces rojas (LR) a lo largo del recorrido y cuando esté parada un juego de luces verdes (LV).

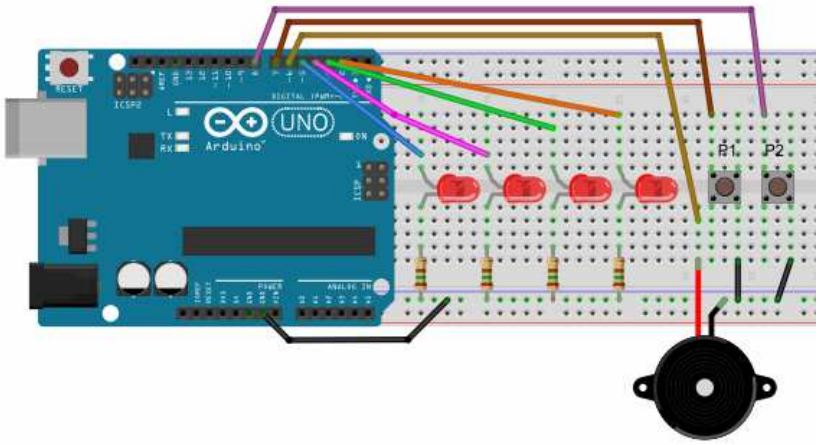




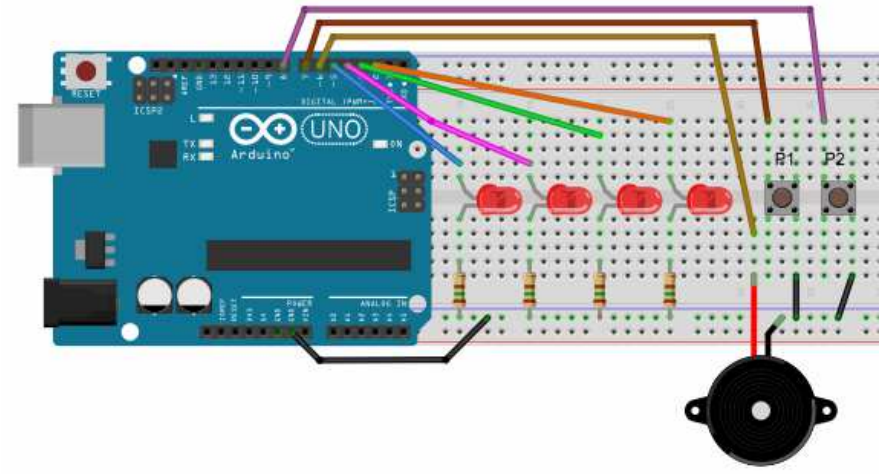
E.6.- Elabora un programa para el control del siguiente sistema: disponemos de dos LDR para contabilizar el número de personas que pasan por una puerta. Si se oscurece la LDR1 y a continuación las LDR2 la persona entra y si el orden de los oscurecimientos es el inverso, la persona sale. Cada vez que entra una persona se enciende un LED adicional, con un máximo de 4. Cada vez que sale una persona se apaga un LED, hasta un mínimo de 0. Se debe tener en cuenta que si se oscurece una LDR y tras su iluminación la siguiente en oscurecerse es ella misma, no se contará.



E.7.- Elaborar el programa de control de un sistema con 4 LEDs, dos pulsadores (P1 y P2) y un zumbador (Z). Empiezan los 4 leds apagados. Pulso un número de veces el pulsador P1; cuando pulso P2 se encienden tantos LEDs como pulsaciones haya dado antes en el pulsador P1 (a partir de 4 pulsaciones se encienden sólo los 4 LEDs). Al pulsar de nuevo el pulsador P2 se apagan los LEDs, suena brevemente el zumbador (por ejemplo 0,5 segundos) y se empieza de nuevo.

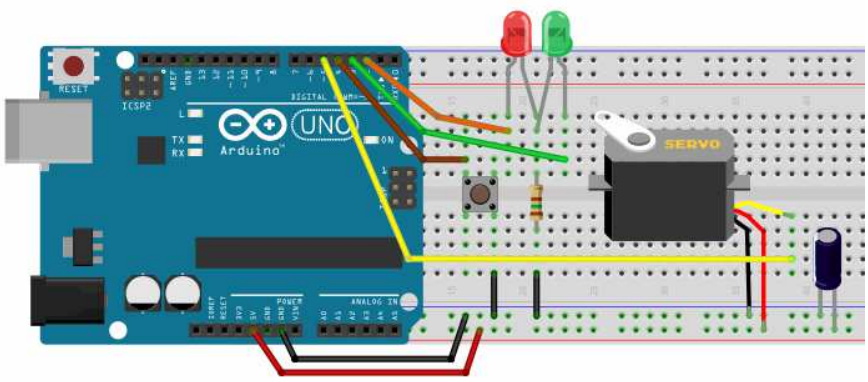


E.8.- Elaborar el programa de control de un sistema con 4 LEDs, dos pulsadores (P1 y P2) y un zumbador (Z) (mismos elementos que en la actividad anterior). Empiezan 2 LEDs encendidos. Por cada pulsación de P1 se enciende un nuevo LED y por cada pulsación del P2 se apaga un LED. Como máximo puede haber los 4 LEDs encendidos y como mínimo todos apagados. Si se pulsa el pulsador P1 cuando ya están los 4 LEDs encendidos sonará el zumbador; igualmente, si se pulsa el pulsador P2 cuando ya están todos los LEDs apagados sonará el zumbador. Tener en cuenta que si se mantiene un pulsador pulsado no debe contarlo más que como una pulsación, es decir, hay que dejar de pulsar entre pulsación y pulsación.



BLOQUE F: SERVOMOTORES

F.0.1.- Ejemplo resuelto. Elaborar programa para controlar una barrera de aparcamiento movida por un servomotor. Con la barrera bajada o en camino de subida o bajada estará el LED rojo encendido, con la barrera abierta estará encendido el LED verde. Al pulsar un pulsador, la barrera subirá, se mantendrá subida durante 6 segundos y luego bajará. La subida será de forma súbita, mientras que la bajada será suave. Si durante la bajada se pulsa el pulsador la barrera subirá de nuevo. Si durante el tiempo de espera subida, se pulsa de nuevo el pulsador, el cómputo del tiempo de espera se reiniciará.

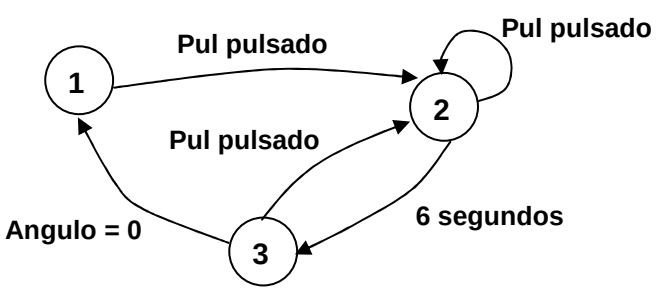


Solución:

Asignación de estados:

- Estado 1 barrera bajada: LR= HIGH, LV= LOW; al pulsar Pul el servo sube la barrera y pasa a estado 2.
- Estado 2 barrera subida; LR = LOW, LV = HIGH; a los 6 segundos pasa a estado 3, salvo que se pulse Pul con lo que reinicia el cómputo de tiempo.
- Estado 3 barrera bajando; LR = HIGH, LV = LOW, si se pulsa Pul sube la barrera y vuelve a estado 2. Si no se pulsa, cuando el ángulo llega a 0 pasa a estado 1.

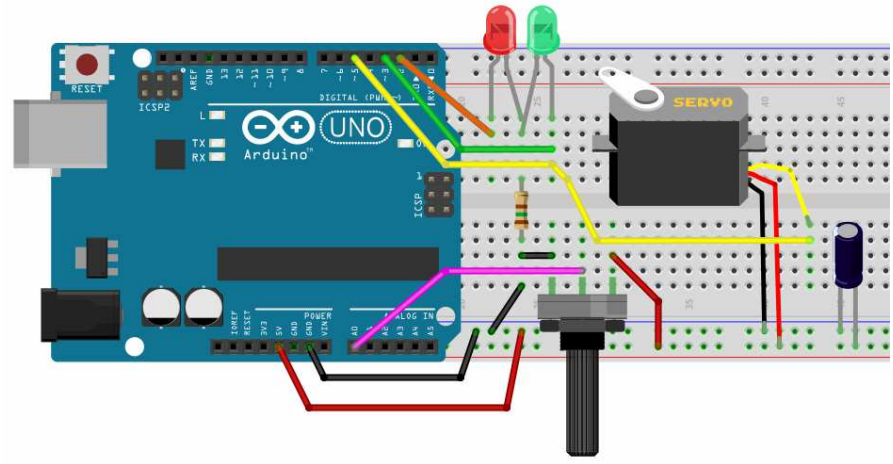
Asignación de estado inicial: estado 1



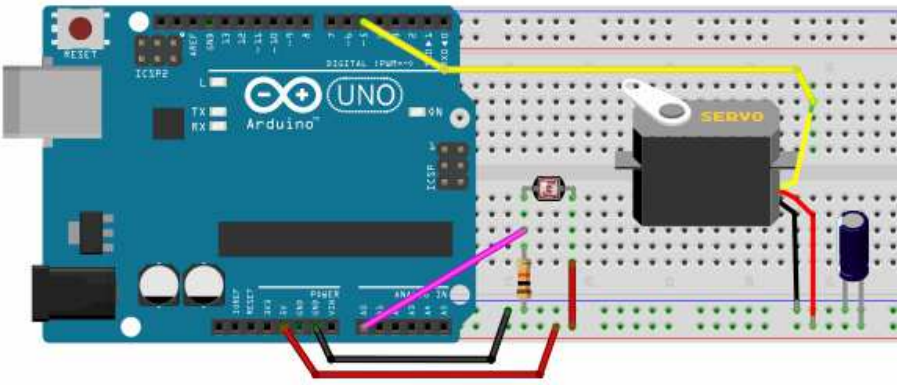

```
#include <Servo.h>
Servo servoA;
const int LR=2;
const int LV=3;
const int Pul=4;
int estadobarrera=1;
unsigned long tini;
int angulo=0;

void setup() {
  servoA.attach(5);
  pinMode(LV, OUTPUT);
  pinMode(LR, OUTPUT);
  pinMode(Pul, INPUT_PULLUP)
  servoA.write(0);
}

void loop() {
  controla_barrera();
}
```

```
void controla_barrera(){
  switch (estadobarrera){
    case 1:
      digitalWrite(LR, HIGH);
      digitalWrite(LV, LOW);
      if(digitalRead(Pul)==LOW){
         servoA.write(90);
         delay(20);
         estadobarrera=2;
         tini=millis();
      }
      break;
    case 2:
      digitalWrite(LR, LOW);
      digitalWrite(LV, HIGH);
      if(digitalRead(Pul)==LOW)
        tini=millis();
      if(millis()-tini>6000){
        estadobarrera=3;
        angulo=90;
      }
      break;
    case 3:
      digitalWrite(LR, HIGH);
      digitalWrite(LV, LOW);
      if(digitalRead(Pul)==LOW){
         servoA.write(90);
         delay(20);
         estadobarrera=2;
         tini=millis();
      }
      else {
        angulo=angulo-1;
        if(angulo>=0){
            servoA.write(angulo);
            delay(20);
        }
        else estadobarrera=1;
      }
      break;
  }
}
```

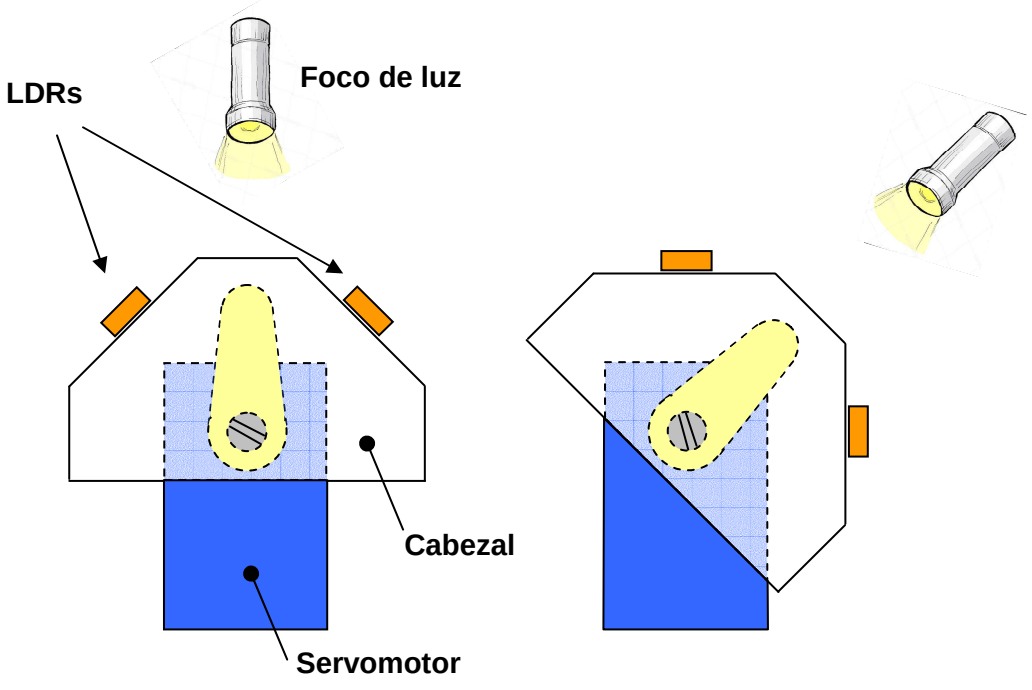
F.1.- Elaborar el programa para posicionar un servomotor entre 0º y 180º con un potenciómetro. Cuando el servo llegue a 0º se encenderá un LED rojo y cuando llegue a 180º se encenderá un LED verde.



F.2.- Elaborar el programa para posicionar un servomotor entre 0º y 180º variando la luz incidente sobre una LDR.



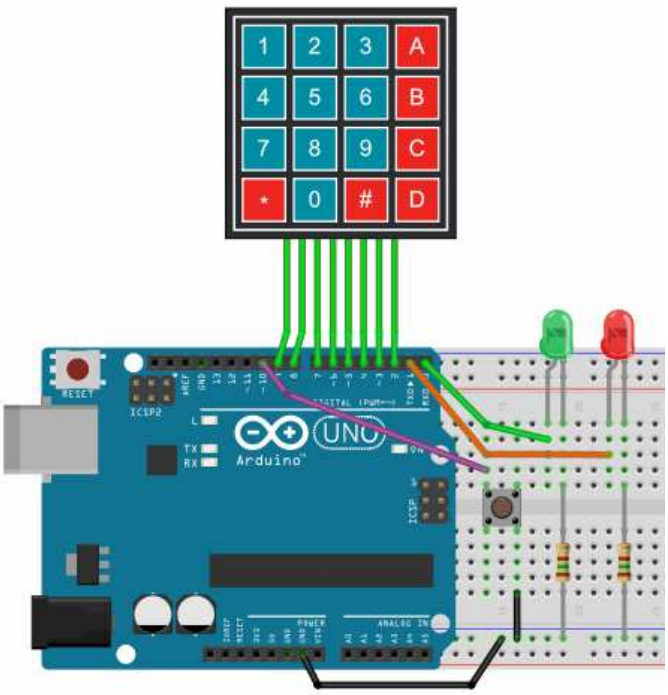
F.3.- Elaborar el programa para posicionar un servomotor en la dirección de una fuente de luz. El rango de barrido estará entre 0º y 180º. El servomecanismo llevará acoplado un cabezal giratorio formado por dos LDR en posiciones simétricas como se indica en la figura. El programa debe posicionar el servo de modo que el cabezal apunte hacia el foco, lo que ocurrirá cuando las dos LDR reciban la misma luz. Al hacer los experimentos habrá que asegurarse de que la luz exterior no incide más sobre una LDR que sobre la otra.



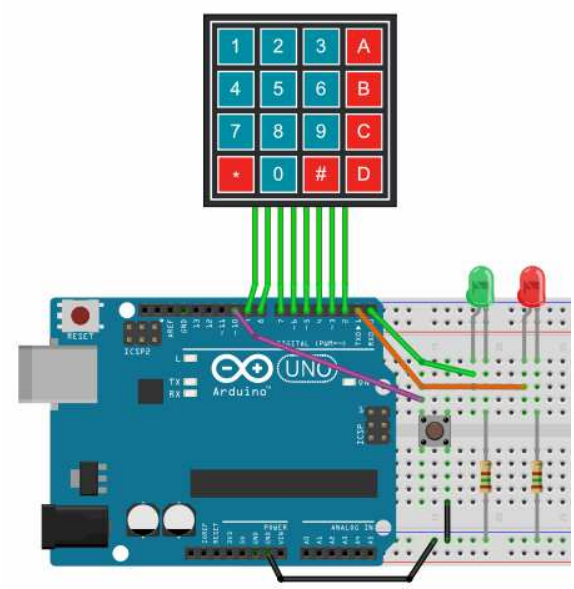
BLOQUE G: TECLADO (KEYPAD)

G.0.1.- Ejemplo resuelto. Programa para que al pulsar la tecla A del teclado, se encienda el LED verde, al pulsar la tecla B, se encienda el LED rojo, al pulsar la tecla 0 se apague el LED verde y al pulsar el pulsador se apague el LED rojo.

```
#include <Keypad.h>

const byte Fil = 4;
const byte Col = 4;
char teclas[Fil][Col] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte pinFil[Fil] = {9, 8, 7, 6};  //en orden F1, F2, F3 y F4
byte pinCol[Col] = {5, 4, 3, 2};  //en orden C1, C2, C3 y C4
Keypad Teclado = Keypad(makeKeymap(teclas), pinFil, pinCol, Fil, Col);

const int LV=0;
const int LR=1;
const int Pul=10;

void setup(){
  pinMode(LV, OUTPUT);
  pinMode(LR, OUTPUT);
  pinMode(Pul, INPUT_PULLUP);
}

void loop(){
  revisa_teclado();
  revisa_pulsador();
}

void revisa_teclado(){
  char tecla = Teclado.getKey();
  if(tecla != NO_KEY){
    switch(tecla){
      case 'A':
        digitalWrite(LV, HIGH);
        break;
      case 'B':
        digitalWrite(LR, HIGH);
        break;
      case '0':
        digitalWrite(LV, LOW);
        break;
    }
  }
}

void revisa_pulsador(){
  if(digitalRead(Pul)==LOW)digitalWrite(LR, LOW);
}
```



G.0.2.- Ejemplo resuelto. Programa aplicado al mismo sistema anterior de modo que el LED verde se enciende pulsado la combinación seguida de teclas A10, se apaga pulsado la combinación seguida de teclas B0. El LED rojo se enciende pulsando la tecla D y se apaga pulsando el pulsador.



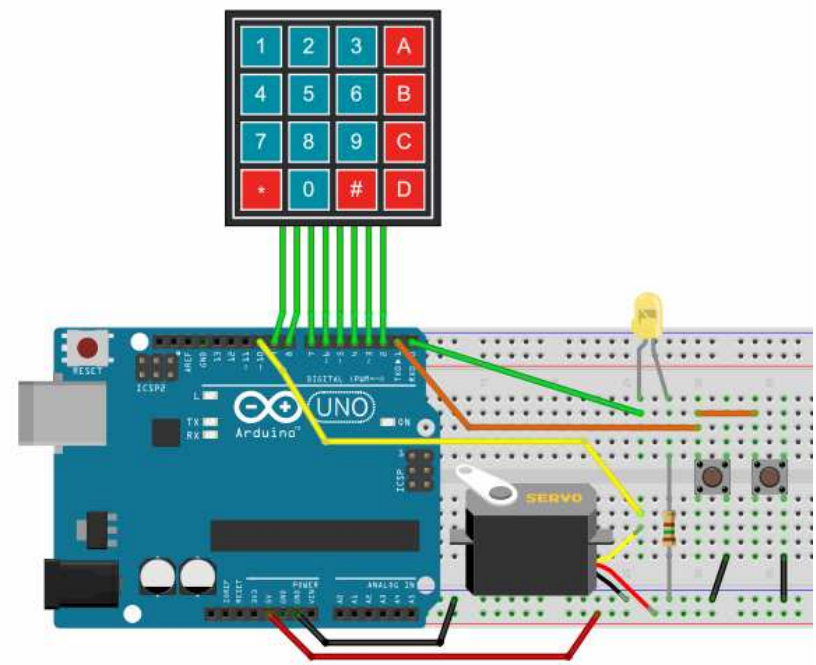
El programa debe tener en cuenta, que el encendido o el apagado del LED rojo se pueda hacer de forma independiente del LED verde. Es decir, si, por ejemplo, habiendo introducido la secuencia A1 (falta el 0) porque estoy encendiendo el LED verde, introduzco la D, se encenderá el LED rojo, y si, a continuación, introduzco el 0 que faltaba, se encenderá el LED verde. Sin embargo, si la tecla que hubiera introducido no fuera la D, entonces la A1 que había introducido se pierde y habría que empezar de nuevo.

Solución:

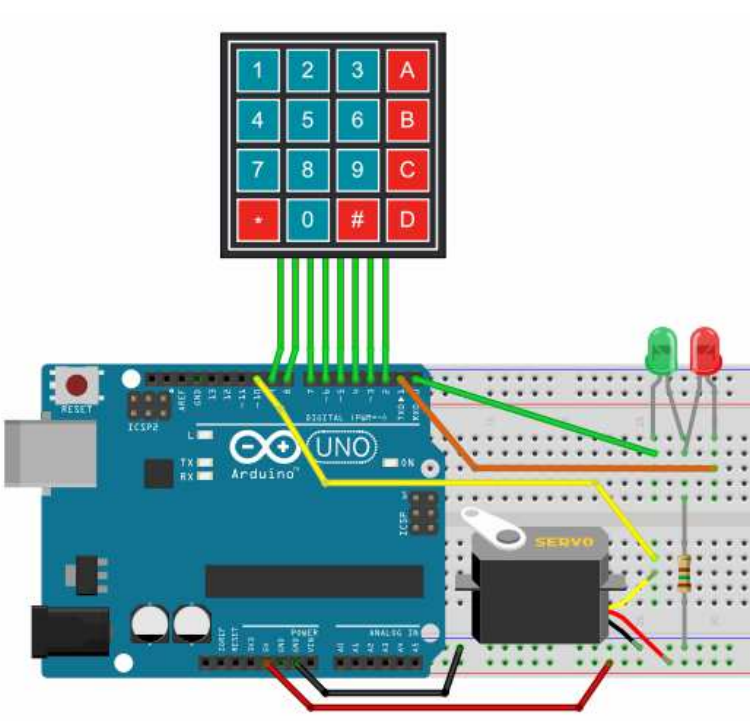
```
#include <Keypad.h>

const byte Fil = 4;
const byte Col = 4;
char teclas[Fil][Col] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte pinFil[Fil] = {9, 8, 7, 6};
    //en orden F1, F2, F3 y F4
byte pinCol[Col] = {5, 4, 3, 2};
    //en orden C1, C2, C3 y C4
Keypad Teclado = Keypad(makeKeymap(teclas),
           pinFil, pinCol, Fil, Col);

const int LV=0;
const int LR=1;
const int Pul=10;
int estadoteclado=0;

void setup(){
  pinMode(LV, OUTPUT);
  pinMode(LR, OUTPUT);
  pinMode(Pul, INPUT_PULLUP);
}

void loop(){
  revisa_teclado();
  revisa_pulsador();
}
```

```
void revisa_teclado(){
  char tecla = Teclado.getKey();
  if(tecla != NO_KEY){
    switch(tecla){
      case 'A':
        estadoteclado=10;
        break;
      case 'B':
        estadoteclado=20;
        break;
      case 'D':
        digitalWrite(LR, HIGH);
        break;
      default:
        switch(estadoteclado){
          case 10:
            if(tecla=='1')estadoteclado=11;
            else estadoteclado=0;
            break;
          case 11:
            if(tecla=='0')digitalWrite(LV,HIGH);
            estadoteclado=0;
            break;
          case 20:
            if(tecla=='0')digitalWrite(LV,LOW);
            estadoteclado=0;
            break;
        }
    }
  }
}

void revisa_pulsador(){
  if(digitalRead(Pul)==LOW)digitalWrite(LR, LOW);
}
```

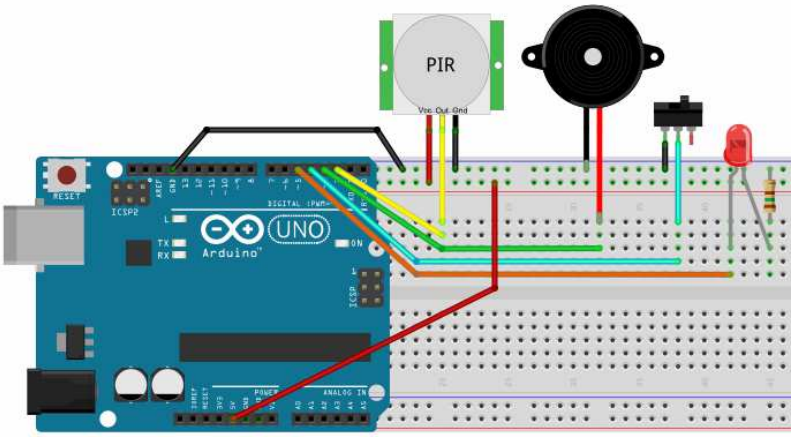
G.1.- Elaborar programa para controlar la apertura y cierre de una barrera de aparcamiento movida por un servomotor. A pulsar la tecla A de un teclado, la barrera subirá y un LED que simula a las luces del aparcamiento se encenderá. A los 5 segundos, la barrera bajará sola, salvo que antes se pulse la tecla C, con lo que la barrera bajará sin esperar a los 5 segundos. El LED que simula las luces se apagará solo a los 10 segundos de haberse encendido, sin que le afecte la pulsación de C. En caso de que pasen los 10 segundos y el LED se apague, habrá dos pulsadores (se supone que en realidad habrá más distribuidos por el aparcamiento), de modo que cada vez que pulsamos cualquiera de ellos, el LED se enciende otros 10 segundos. Si la pulsación de algún pulsador se produce antes de que hayan pasado los 10 segundos, el plazo de tiempo volverá a iniciarse.



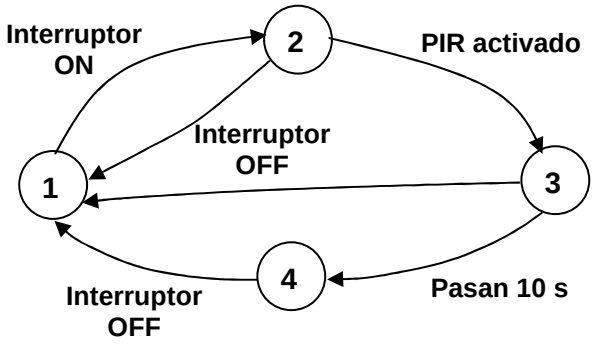
G.2.- Elaborar un programa para controlar la apertura y cierre de una barrera de aparcamiento movida por un servomotor. Disponemos del servo, un teclado y dos LEDs (LR y LV). Si tecleamos la secuencia A27 la barrera subirá y bajará automáticamente a los 5 segundos. Si lo que tecleamos es B27, la barrera subirá y bajará automáticamente a los 10 segundos. Si lo que pulsamos es C27, la barrera sube y permanece en dicho estado hasta que pulsamos C0, con lo cual bajará la barrera. Si estando la barrea subida en los dos primeros casos, en espera de que pase el tiempo, se teclea C0, la barrerá bajará sin esperar a que se cumpla el plazo de espera. Por simplificar, tanto las subidas como las bajadas de la barrera serán de forma súbita. En cuanto a los LEDs, funcionarán como en otros casos, estará encendido el LED verde con la barrera totalmente levantada y el LR durante los movimientos de subida o bajada o cuando esté bajada.



BLOQUE H: SONIDO Y DETECCIÓN DE PRESENCIA PIR



H.0.1.- Ejemplo resuelto. Elaborar un programa para controlar un sistema de alarma. Por simplificar, la alarma se conecta y desconecta mediante un interruptor. Cuando la alarma está conectada o activada, está encendido el LED. El sonido de la alarma lo produce un zumbador. La activación de la alarma es producida por un sensor de movimiento PIR. Estando conectada la alarma, si se activa el sensor PIR suena un sonido breve (para recordar al propietario que tiene que desconectar la alarma y se da un margen de 10 segundos para desconectar la alarma antes de hacer sonar el zumbador (esto es para dar al propietario tiempo de desconectar la alarma). Observa cómo está construida la función toca_alarma().

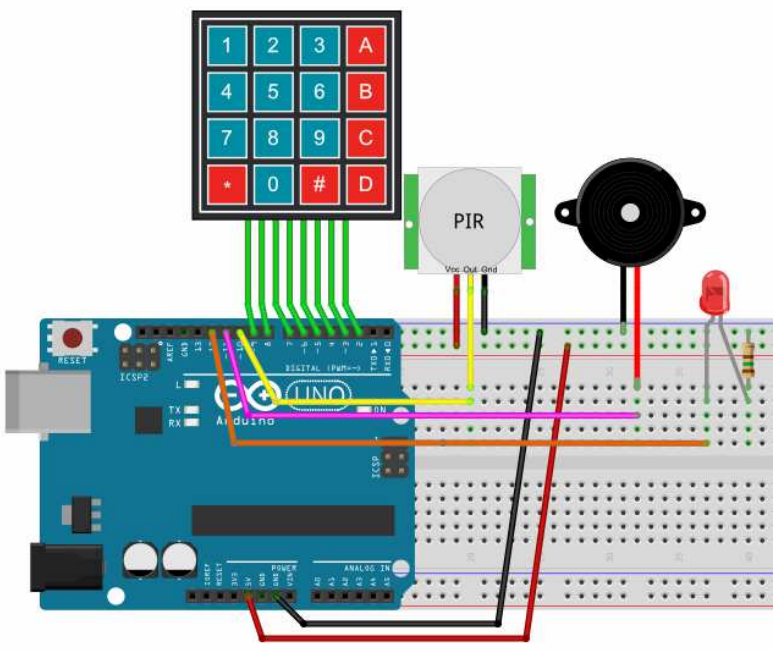



```
const int pinPIR=2; //detector presencia PIR
const int pinZumb=3; //zumbador
const int pinInt=4;  //interruptor
const int pinLED=5;
int estadoalarma=1;
unsigned long tini;
const int Fmin=2000; //frec. mínima sonido
const int Fmax=4000; //frec. máxima sonido
int tono;
int ascendente=1; //sentido variación tono

void setup() {
  pinMode(pinPIR,INPUT);
  pinMode(pinZumb,OUTPUT);
  pinMode(pinInt,INPUT_PULLUP);
  pinMode(pinLED,OUTPUT);

}

void loop() {
  revisa_alarma();
}
```

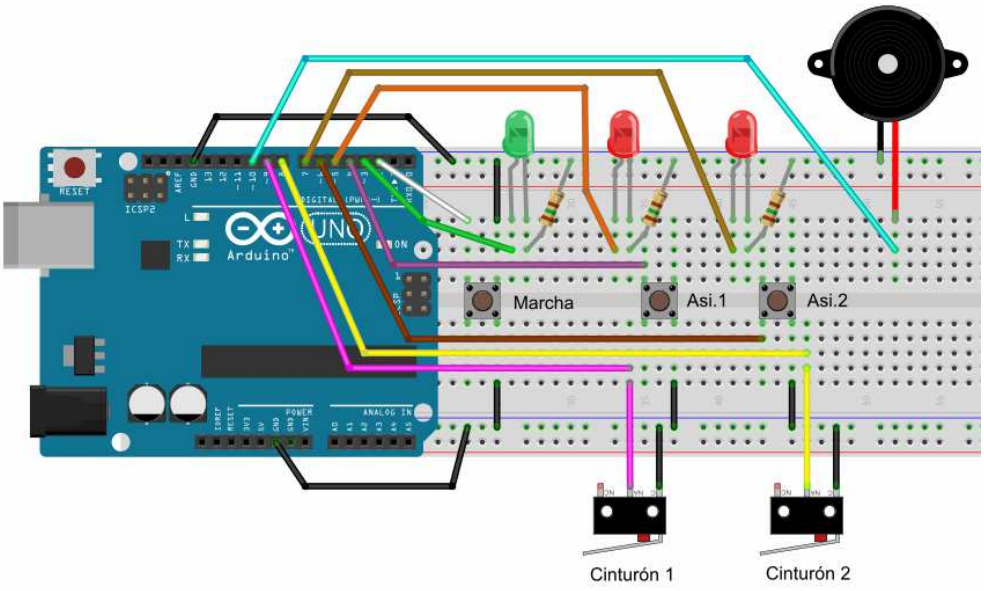
```
void revisa_alarma(){
  switch (estadoalarma){
    case 1:  //alarma desconectada
      noTone(pinZumb);
      tono=Fmin-1;
      ascendente=1;
      digitalWrite(pinLED,LOW);
      if(digitalRead(pinInt)==LOW)estadoalarma=2;
      break;
    case 2:  //alarma conectada
      digitalWrite(pinLED,HIGH);
      if(digitalRead(pinInt)==HIGH)estadoalarma=1;
      if(digitalRead(pinPIR)==HIGH){
        estadoalarma=3;
        tone(pinZumb,440,200); //pitido de aviso
        tini=millis();
      }
      break;
    case 3:  //en estado de espera desconexión
      if((millis()-tini)>10000)estadoalarma=4;
      if(digitalRead(pinInt)==LOW)estadoalarma=1;
      break;
    case 4:  //alarma activada
      toca_alarma();
      if(digitalRead(pinInt)==LOW)estadoalarma=1;
      break;
  }
}

void toca_alarma(){
  if(ascendente==1){
    tono++;
    if(tono>=Fmax)ascendente=0;
  }
  else {
    tono--;
    if(tono<=Fmin)ascendente=1;
  }
  tone(pinZumb,tono);
}
```

H.1.- Mejorar el programa anterior de la alarma, haciendo que desde que el propietario conecta la alarma con el interruptor, haya un margen de 7 segundos para que el propietario pueda salir de la vivienda sin que se active la alarma. Esto es lo lógico pues la zona donde está el interruptor de conexión/desconexión debe estar en la zona cubierta por el detector de presencia PIR.

H.2.- Mejorar el sistema de alarma anterior haciendo que la conexión/desconexión de la alarma sea a través de un teclado. Para conectar será suficiente con pulsar la tecla C, pero para desconectar habrá que teclear una clave de cuatro dígitos (por ejemplo D123).

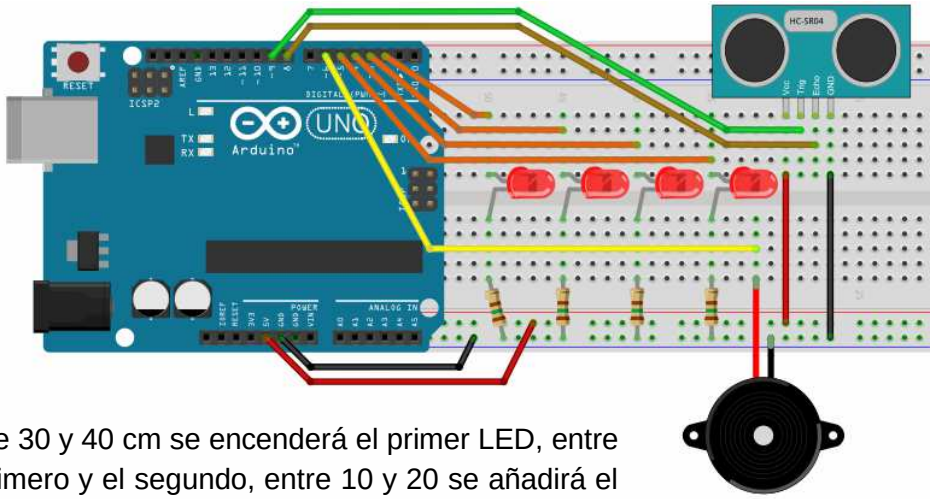


H.3.- Queremos simular un sistema de seguridad para el automóvil, de modo que si tratamos de poner en marcha un vehículo sin que el conductor y/o el copiloto (si es que lo hay) lleven puesto el cinturón, no nos lo permitirá. Además nos avisará con una señal luminosa y una acústica. El giro de la llave de contacto será simulado por un pulsador. La puesta en marcha del vehículo será simulada por el encendido de un LED verde. Si tenemos puesto los cinturones, lo cual será simulado por unos finales de carrera pisados, al pulsar el pulsador de contacto se encenderá el LED verde. Si los asientos están ocupados se detectará por la presión sobre sendos pulsadores (irían bajo los asientos). Si estando alguien sentado sobre el asiento éste no lleva el cinturón puesto, se encenderá el LED rojo correspondiente al asiento y el zumbador emitirá un pitido intermitente y el vehículo no se pondrá en marcha. Si el conductor no está sentado en su asiento, tampoco se podrá poner en marcha el vehículo. Evidentemente, si no va nadie en el asiento del copiloto no tendrá que ir puesto el cinturón de dicho asiento. Por último, si ya está el vehículo en marcha y el conductor o el copiloto se quitan el cinturón, se encenderá el LED rojo correspondiente y el zumbador pitará, pero no se desconectará el vehículo.



BLOQUE I: SENSOR DE DISTANCIA ULTRASÓNICO

I.0.1.- Ejemplo resuelto. Programa para visualizar el acercamiento de un objeto al sensor ultrasónico. El sistema constará del sensor, cuatro LEDs y un zumbador. Cuando el objeto se encuentre a una distancia superior a 40 cm del sensor no se encenderá ningún LED, entre 30 y 40 cm se encenderá el primer LED, entre 20 y 30 se encenderán el primero y el segundo, entre 10 y 20 se añadirá el tercero y el zumbador emitirá pitidos intermitentes y a menos de 10 cm se encenderán los cuatro LEDs y el zumbador emitirá un sonido continuo.



Solución:

```
#include <Ultrasonic.h>

Ultrasonic sensor(9,8); //(Trig, Eco)
const int L1=2;
const int L2=3;
const int L3=42;
const int L4=5;
const int Zumb=6;
int distancia;
unsigned long tini=0;

void setup() {
  pinMode(L1,OUTPUT);
  pinMode(L2,OUTPUT);
  pinMode(L3,OUTPUT);
  pinMode(L4,OUTPUT);
  pinMode(Zumb,OUTPUT);
}

void loop() {
  controla_sensor();
}
```

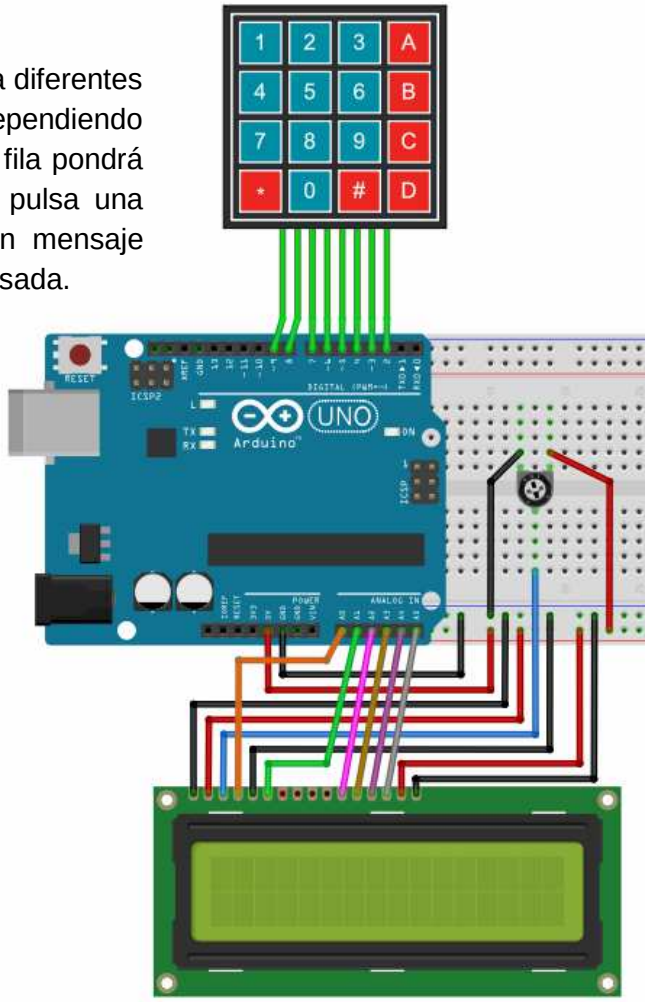
```
void controla_sensor(){
  distancia=sensor.Ranging(CM);
  if(distancia<40)digitalWrite(L1,HIGH);
  else digitalWrite(L1,LOW);
  if(distancia<30)digitalWrite(L2,HIGH);
  else digitalWrite(L2,LOW);
  if(distancia<20){
    digitalWrite(L3,HIGH);
    toca_pitido();
  }
  else {
    digitalWrite(L3,LOW);
    noTone(Zumb);
  }
  if(distancia<10){
    digitalWrite(L4,HIGH);
    tone(Zumb,440);
  }
  else {
    digitalWrite(L4,LOW);
    noTone(Zumb);
  }
}

void toca_pitido(){
  if((millis()-tini)>500){
    tone(Zumb,440,200);
    tini=millis();
  }
}
```

I.1.- Mejorar el programa anterior haciendo que el zumbador emita pitidos intermitentes con una frecuencia mayor conforme se va acercando el objeto al sensor, de modo que cuando se encienda el cuarto LED el pitido llegue a ser continuo.

I.2.- Controlar el posicionamiento del eje de un servomotor en función de la distancia de un objeto al sensor ultrasónico. Por ejemplo, a una distancia de 10 cm o menor, el eje debe estar en 0º, a 20 cm en 90º y a 30 cm o más en 180º.

BLOQUE J: DISPLAY LCD

J.0.1.- Ejemplo resuelto. Programa que presenta diferentes mensajes en la primera fila de la pantalla LCD dependiendo de la tecla pulsada en el teclado. En la segunda fila pondrá de forma permanente IES BELLAVISTA. Si se pulsa una tecla que no sea 1, 2, 3, 4 ó 5 aparecerá un mensaje diciendo “No hay mensaje x”, siendo x la tecla pulsada.



Solución:

```
#include <LiquidCrystal.h>
#include <Keypad.h>

LiquidCrystal miLCD(A0,A1,A2,A3,A4,A5);

const byte Fil = 4;
const byte Col = 4;
char teclas[Fil][Col] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte pinFil[Fil] = {9, 8, 7, 6};
    //en orden F1, F2, F3 y F4
byte pinCol[Col] = {5, 4, 3, 2};
    //en orden C1, C2, C3 y C4
Keypad Teclado = Keypad(makeKeymap(teclas),
           pinFil, pinCol, Fil, Col);

String mensaje[]={"                ",
                  "Abro puerta",
                  "Alarma conectada",
                  "Alarma de Fuego",
                  "Enciendo lampara",
                  "Detecto intruso"
};

void setup() {
  miLCD.begin(16,2);
  miLCD.clear();
  miLCD.setCursor(0,1);
  miLCD.print(" IES BELLAVISTA ");
}

void loop() {
  revisa_teclado();
}

void muestra_mensaje_LCD(int i){
  miLCD.setCursor(0,0);
  miLCD.print(mensaje[0]);
  miLCD.print(mensaje[i]);
}
```

```
a);

void revisa_teclado(){
  char tecla = Teclado.getKey();
  if(tecla != NO_KEY){
    switch(tecla){
      case '1':
        muestra_mensaje_LCD(1);
        break;
      case '2':
        muestra_mensaje_LCD(2);
        break;
      case '3':
        muestra_mensaje_LCD(3);
        break;
      case '4':
        muestra_mensaje_LCD(4);
        break;
      case '5':
        muestra_mensaje_LCD(5);
        break;
      default:
        miLCD.setCursor(0,0);
        miLCD.print(mensaje[0]);
        miLCD.print("No hay mensaje "+tecla);
    }
  }
}
```