MANUAL

CREADOR_PLANTILLAS

Versión 2.2.2

Por: the FDK

AVISO LEGAL

El autor no ofrece ninguna garantía en relación con el contenido del software, sus funciones específicas, su fiabilidad, su disponibilidad ni su capacidad para satisfacer sus necesidades. El software se ofrece «tal cual».
En los casos permitidos por la ley, el autor no será responsable de la pérdida de beneficios, de ingresos ni de datos, ni de pérdidas financieras ni de daños indirectos, especiales, derivados, ejemplares ni punitivos.

« Copyright 2016 the FDK» el programa se distribuye bajo los términos de la General Public License de GNU.

Este manual y el icono están bajo una:

INDICE pág.

INTRODUCCIÓN

Creador_Plantillas es un programa que genera plantillas de sketch de Arduino, para que trabaje siguiendo un grafcet.

El programa mediante consola pregunta al usuario la descripción del grafcet que se va a crear, este tendrá que introducir por teclado sus características.

Al finalizar, se guarda la configuración del grafcet en el archivo guardado.txt, para poder crear rápidamente otro sketch igual o modificarlo y hacer uno parecido sin volver a introducir todo de nuevo, puede finalizar el programa introduciendo: exit, sin generar la plantilla.

Se aconseja crear tablas donde se indique sus entradas, salidas, temporizadores y variables (Ver ejemplos).

Si el programa se ejecuta en la carpeta Arduino donde está el sketchbook o proyectos, se podrán cargar rápidamente las plantillas desde el programa de Arduino.

En las siguientes páginas se explica cómo introducir las características del grafcet.

NOMBRE

Al finalizar, la plantilla se crea una carpeta al lado del ejecutable, con nombre: según usuario o por defecto con el nombre “platilla”, dentro de la carpeta se encuentran 2 archivos uno .h y otro .ino, con el mismo nombre, para ejecutar con el programa Arduino.

La letra Ñ y acentos los detectara como símbolo y no serán válidos, los espacios en blanco serán remplazados por “_”.

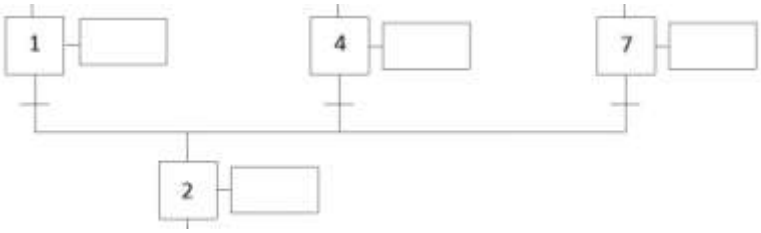
Si se crea una plantilla con nombre igual a otra solo sustituirá los archivos .h y .ino en la carpeta.

ESTRUCTURA

- Etapas anteriores (Entradas)

  Hay que indicar el número de etapas máximas que puedan activar la siguiente etapa.

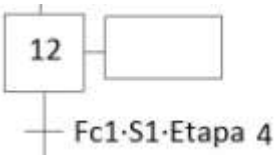
  En este caso son 3 las que activan la etapa 2.



- Condiciones

Indicar el número máximo de condiciones necesarias para la transición a la siguiente etapa, hay que tener en cuentan la etapa anterior.

Siendo 4, 3 + 1 de la etapa 12.

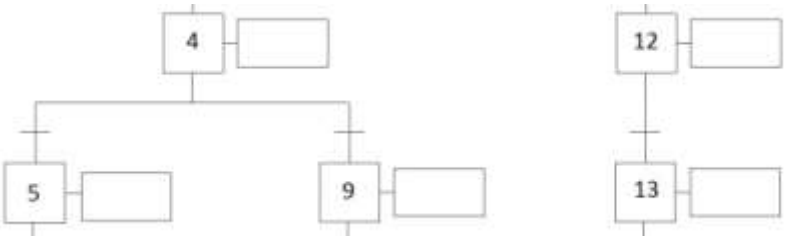


En el caso de que las condiciones fueran en OR (FC1+S1+Etapa) o de otra forma diferente, se deberá modificar la función para la transición o crear otra nueva.

- Etapas Posteriores (Salidas)

  Indicar el número de etapas máximas que puede activar una etapa.

  Son 2 las que activa la etapa 4, las otras 2 no suman.



- Secuencias simultáneas

Indicar si varias etapas pueden estar funcionando a la vez por activación simultánea y finalizando en la misma etapa (**Ejemplo direccional y simultáneo**)

En el sketch, en la parte de transiciones, la etapa en que converjan las otras se tendrá que indicar:

Inicial: E[f] = etapa ([E[f],….

Modificado: E[f] = etapand ([E[f],….

- Secuencias lineales

En el archivo .h viene la función etapalin con su transición, para sustituir aquellas transiciones en las que se siga una secuencia lineal para así simplificar.

Inicial: E[f] = etapa ([E[f],….

Modificado: E[f] = etapalin ([E[f],….

TEMPORIZADORES

Indicar el número de temporizaciones que tienen que hacer acciones retardadas y limitadas a un tiempo.

Si se tuviera que retardar una acción 2 segundos y otra limitarla durante 5 segundos, se indicaría que hay 2 temporizadores, una de 2000 (2 segundos) y otra de 5000 (5 segundos).

Para el Arduino constarían como T[0] y T[1].

Se implementaría en el sketch, en la parte de acciones, esta función:

```
//Función para temporizar:
if(fe[f]==1) pret[0] = pretemp (tempst[0]);
T[0] = temp (pret[0]);
```

Modificando el [0] a [1] en las etapas (case X) que se active el T1. La función se encuentra en el archivo .h.

Temporización máxima con: unsigned int 65’535 segundos.

Temporización máxima con: long 2.147.483’647 segundos, 24d 20h 31m 23'647s.

Temporización máxima con: unsigned long 4.294.967’295 segundos, 49d 17h 2m 47'295s, el tiempo actual del microprocesador más el tiempo de temporización no debe superar esa cifra, para así evitar desbordamientos de datos.

- Parpadeo en la misma etapa

Función parpadeo: La salida se mantiene alternando en alto y bajo en la misma etapa y el mismo periodo de tiempo o diferente si tempst[x] es diferente en las dos funciones. En la función pretemp se indicara el tiempo que la salida este en alto y en parpadeo el tiempo que este en bajo.

Se puede eliminar: “If(fe[f]==1) M[x]=3; ” de la función si: el vector pret esta a 0 inicialmente y el tiempo en el microprocesador es superior a tempst[x] a la hora de actuar.

En el **Ejemplo direccional y simultáneo** se ve cómo hacer parpadear.

ETAPAS

Indicar el número total de etapas teniendo en cuenta las etapas iniciales.

- Etapas iniciales

Indicar el número de las etapas iniciales, la etapa 0 está por defecto incluida, si no se quiere indicar más etapas iniciales, pulse intro sin introducir ninguna cifra y así continuar en la creación de la plantilla.

- En el sketch.

En la parte de **transiciones**: E[f] significa etapa actual

E[f+1], E[f+2]… Etapas siguientes, si E[f] fuera la etapa 9 (case 9), las otras 2 serian la 10 y 11.

E[f-1], E[f-2]… Etapas anteriores si E[f] fuera la etapa 9, las otras 2 serian la 8 y 7.

En la parte de **acciones** el "case 0" corresponderá a la etapa 0, "case 1" etapa 1 y así sucesivamente.

Evitar los E[f-x] cuando sea "case 0", ya que la etapa -1 no existe, igual que si el máximo de etapas es 6 evitar en el "case 5" el E[f+x] por el mismo motivo de que la etapa 6 no existe, teniendo en cuenta la etapa 0.

La condición "1", siempre activa, se utiliza para rellenar una parte de la transición si no llega al máximo de condiciones o si la transición es verdadera siempre, en este último caso el programa puede saltarse la etapa correspondiente, se recomiendo usar una variable, por ejemplo "bool c[1]", ejemplo en la página siguiente.

Y en "0" nunca se cumpliría esa parte de transición a la etapa y se quedaría anulada.

En la parte de transiciones, "default:" se encarga de volver al esta inicial el grafcet en caso de algun fallo en las transiciones.

- Para evitar saltos en las transiciones (Acción impulsional):

Ejemplo para la transición entre la etapa 1 y 2, modificar en la parte de:

//Etapas, flancos, entradas, salidas y variables:

```
bool c [1];
```

//Transiciones:

```
case 2 :
E[f] = etapalin (E[f], E[f - 1], c[0],  E[f + 1]);
break;
```

//Acciones:

```
case 1 :
c[0] = 1;
break;
```

//Reset de temporizadores más contador.

```
for(f=0;f<1;f++) T[f]=0;
for(f=0;f<1;f++) c[f]=0;
```

ENTRADAS – SALIDAS

Indicar el número máximo de pines que se van utilizar como entradas y salidas.

Y luego su respectivo numero de pin.

Quedando en: I [0], I [1]… las entradas.

En el sketch, en las transiciones las entradas serán i[0], i[1]…

Y en: O [0], O [1]… las salidas.

Si hay alguna duplicidad en el número de pin, el programa avisa de ello.

Si la frase final: (*** Plantilla finalizada *** Pulse intro para salir...) no aparece en verde, es señal de que hay algún error en las entradas, salidas de pin o ambas.

- Acciones mantenidas/momentáneas.

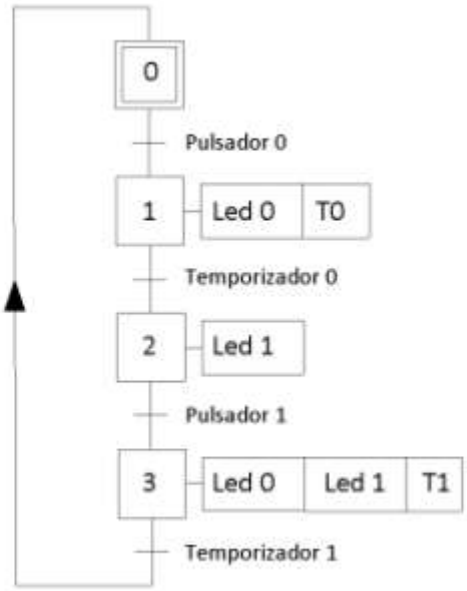
  Para realizar acciones que ha de prolongarse durante dos o más etapas consecutivas:
  En la primera etapa que se active, escriba la digitalWrite (O[x], HIGH);
  Y en la ultima digitalWrite (O[x], LOW);

  Para realizar la acción solo en una etapa:
  Ponga a 1 el M[x] correspondiente al O[x] que desee activar, automáticamente cuando se deje de poner a 1 se desactivara la salida.

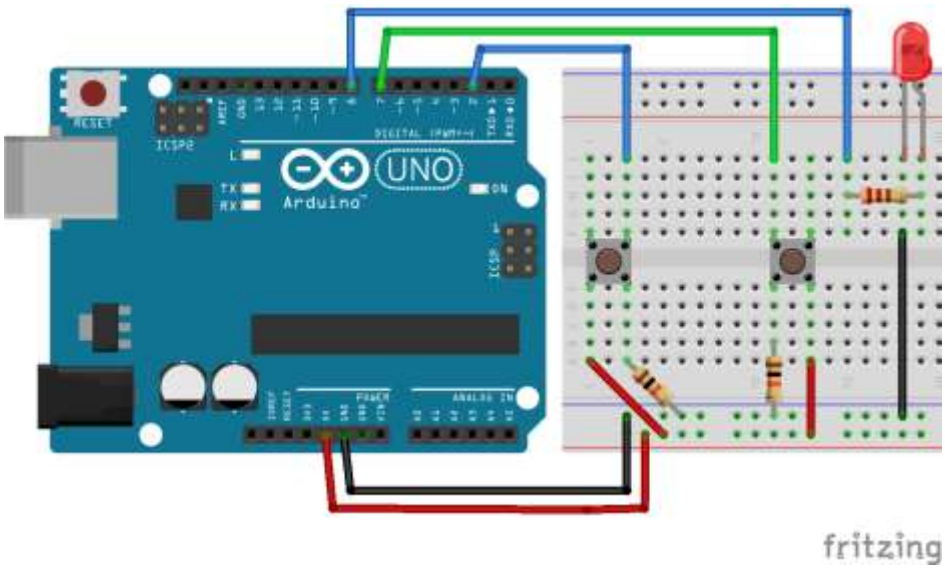
  Véase **ejemplo direccional y simultáneo**.

EJEMPLO LINEAL

Vamos a hacer que nuestro Arduino trabaje siguiendo este grafcet:



Circuito:

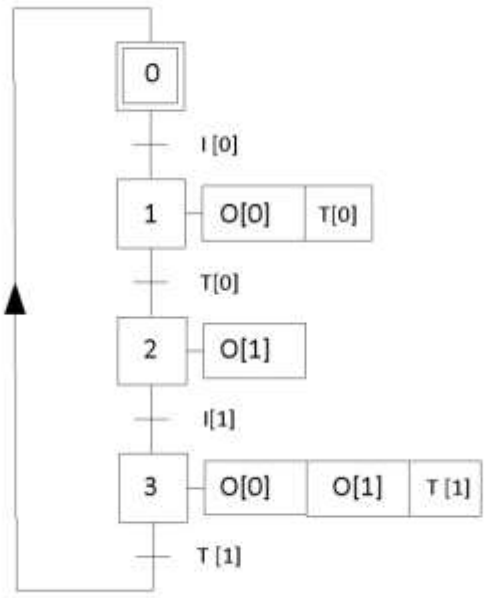


Esta imagen fue creada con Fritzing

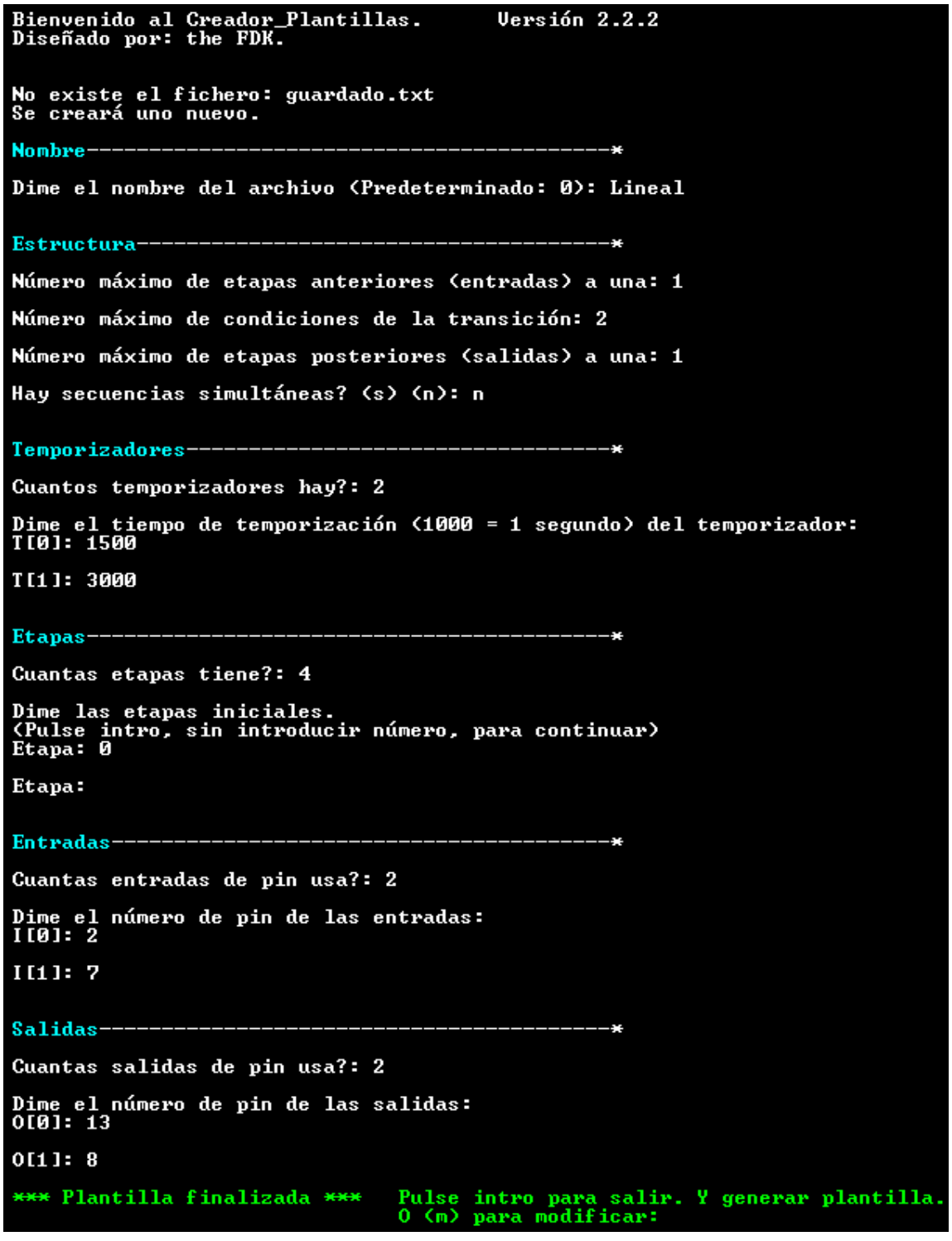
Si es necesario creamos una tabla para saber que pin es el correspondiente a cada uno y evitar cualquier equivocación.

Entradas	Para arduino	Nº Pin
Pulsador 0 Pulsador 1	I[0] I[1]	2 7
Salidas		
Led 0 Led 1	O[0] O[1]	13 8
Temporizadores		Tiempo
Temporizador 0 Temporizador 1	T[0] T[1]	1'5segundos 3 segundos

Nuestro Arduino trabajara así:



Se ejecuta el programa para crear la plantilla.



Una vez finalizada se habrán creado los archivos en la misma ubicación donde se hallé el programa, en el sketch habrá que completar 2 partes:

1)

```
//Transiciones.
for(g=0;g<2;g++){
for(f=3;f>=0;f--){
if(g==0||(g&&E[f])==1){
switch(f){
case 0 :
E[f] = etapa (E[f],E[3],1,E[f+1]);
break;
case 1 :
E[f] = etapa (E[f],E[f-1],1,E[f+1]);
break;
case 2 :
E[f] = etapa (E[f],E[f-1],1,E[f+1]);
break;
case 3 :
E[f] = etapa (E[f],E[f-1],1,E[0]);
break;}
}
}
}
```

En la condición pondremos la entradas de pin i[x], temporizadores T[x] o etapa E[x] que corresponda para que se active la etapa.

Con Ctrl + t, auto formato, se ordenara mejor el sketch.

2)

```
//Acciones.
for(f=0;f<4;f++){
if (E[f]==1){
switch(f){
case 0 :

break;
case 1 :

break;
case 2 :

break;
case 3 :

break;
}
}
}
```

← En los espacios en blanco se rellenara con la acción de encendido, temporización o ambos, correspondiente a esa etapa. La función de temporizar se encuentra al final del archivo **.h**.

Una vez completados quedaran así.

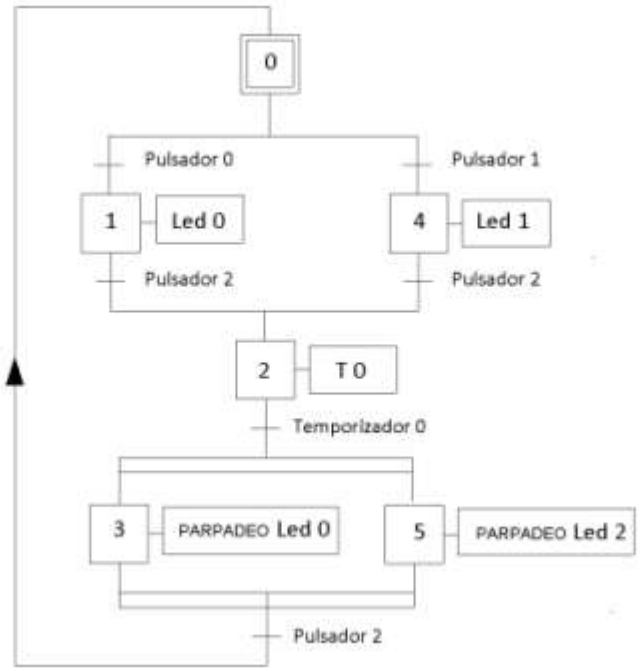
```
//Transiciones.
for(g=0;g<2;g++){
for(f=3;f>=0;f--){
if(g==0||(g&&E[f])==1){
switch(f){
case 0 :
E[f] = etapa (E[f],E[3],T[1],E[f+1]);
break;
case 1 :
E[f] = etapa (E[f],E[f-1],i[0],E[f+1]);
break;
case 2 :
E[f] = etapa (E[f],E[f-1],T[0],E[f+1]);
break;
case 3 :
E[f] = etapa (E[f],E[f-1],i[1],E[0]);
break;}
}
}
}

//Acciones.
for(f=0;f<4;f++){
if (E[f]==1){
switch(f){
case 0 :
digitalWrite(O[0],LOW);
digitalWrite(O[1],LOW);
break;
case 1 :
digitalWrite(O[0],HIGH);
if(fe[f]==1) pret[0] = pretemp (tempst[0]);
T[0] = temp (pret[0]);
break;
case 2 :
digitalWrite(O[0],LOW);
digitalWrite(O[1],HIGH);
break;
case 3 :
digitalWrite(O[0],HIGH);
if(fe[f]==1) pret[1] = pretemp (tempst[1]);
T[1] = temp (pret[1]);
break;
}
}
}
```

← También se pueden usar las acciones momentáneas, M[0] = 1, como en el ejemplo siguiente.

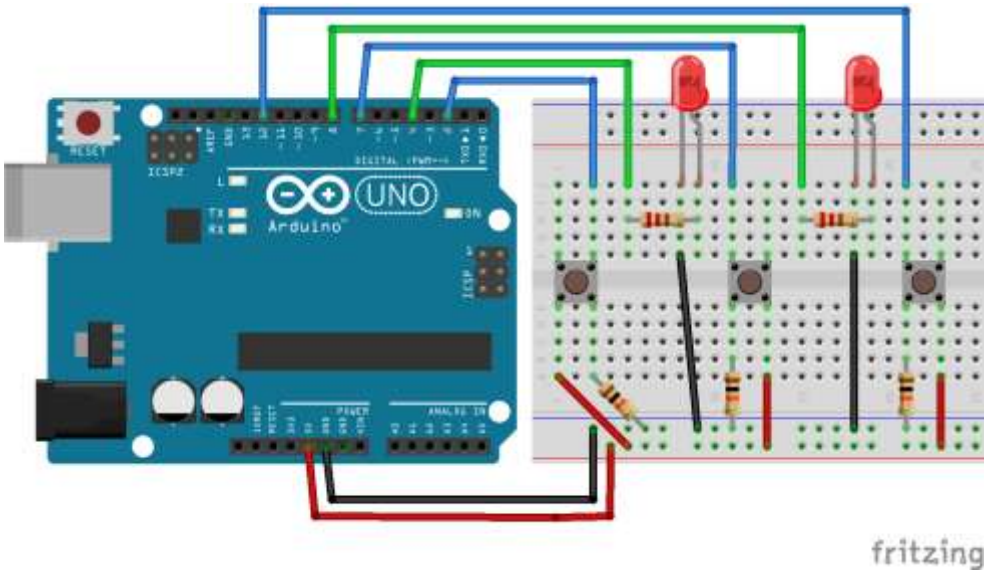
Y ya estará listo para funcionar.

EJEMPLO DIRECCIONAL Y SIMULTÁNEO

Vamos a hacer que nuestro Arduino trabaje siguiendo este grafcet:



Circuito:

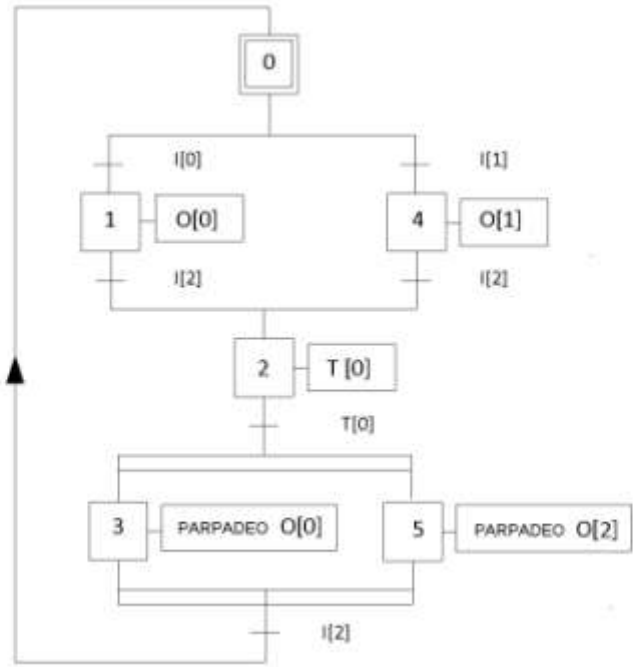


Esta imagen fue creada con Fritzing

Si es necesario creamos una tabla para saber que pin es el correspondiente a cada uno y evitar cualquier equivocación.

Entradas	Para arduino	Nº Pin
Pulsador 0	I[0]	2
Pulsador 1	I[1]	7
Pulsador 2	I[2]	12
Salidas		
Led 0	O[0]	13
Led 1	O[1]	8
Led 2	O[2]	4
Temporizadores		Tiempo
Temporizador 0	T[0]	2 segundos

Nuestro Arduino trabajara así:



Se ejecuta el programa para crear la plantilla.

```
Bienvenido al Creador_Plantillas.      Versión 2.2.2
Diseñado por: the FDK.


Utilizar configuración anterior?
" Lineal "
(s) (n): n


Nombre----------------------------------------*

Dime el nombre del archivo (Predeterminado: 0): Direccional y Simultaneo


Estructura------------------------------------*

Número máximo de etapas anteriores (entradas) a una: 2

Número máximo de condiciones de la transición: 2

Número máximo de etapas posteriores (salidas) a una: 2

Hay secuencias simultáneas? (s) (n): s


Temporizadores--------------------------------*

Cuantos temporizadores hay?: 1

Dime el tiempo de temporización (1000 = 1 segundo) del temporizador:
T[0]: 2000


Etapas----------------------------------------*

Cuantas etapas tiene?: 6

Dime las etapas iniciales.
(Pulse intro, sin introducir número, para continuar)
Etapa: 0

Etapa:


Entradas--------------------------------------*

Cuantas entradas de pin usa?: 3

Dime el número de pin de las entradas:
I[0]: 2

I[1]: 7

I[2]: 12


Salidas---------------------------------------*

Cuantas salidas de pin usa?: 3

Dime el número de pin de las salidas:
O[0]: 13

O[1]: 8

O[2]: 4

*** Plantilla finalizada ***   Pulse intro para salir. Y generar plantilla.
                               O (m) para modificar:
```

Una vez finalizada se habrán creado los archivos en la misma ubicación donde se hallé el programa, en el sketch habrá que completar 2 partes:

1)

```
//Transiciones.
for(g=0;g<2;g++){
for(f=5;f>=0;f--){
if(g==0||(g&&E[f])==1){
switch(f){
case 0 :
E[f] = etapa (E[f],E[5],1,E[4],0,E[f+1],E[f+2]);
break;
case 1 :
E[f] = etapa (E[f],E[f-1],1,E[f-2],0,E[f+1],E[f+2]);
break;
case 2 :
E[f] = etapa (E[f],E[f-1],1,E[f-2],0,E[f+1],E[f+2]);
break;
case 3 :
E[f] = etapa (E[f],E[f-1],1,E[f-2],0,E[f+1],E[f+2]);
break;
case 4 :
E[f] = etapa (E[f],E[f-1],1,E[f-2],0,E[f+1],E[f+2]);
break;
case 5 :
E[f] = etapa (E[f],E[f-1],1,E[f-2],0,E[0],E[1]);
break;}
}
}
}
```

En la condición pondremos la entradas de pin i[x], temporizadores T[x] o etapa E[x] que corresponda para que se active la etapa.

Con Ctrl + t, auto formato, se ordenara mejor el sketch.

2)

```
//Acciones.
for(f=0;f<6;f++){
if (E[f]==1){
switch(f){
case 0 :

break;
case 1 :

break;
case 2 :

break;
case 3 :

break;
case 4 :

break;
case 5 :

break;
}
}
}
```

← En los espacios en blanco se rellenara con la acción de encendido, temporización o ambos, correspondiente a esa etapa. La función de temporizar se encuentra al final del archivo **.h**.

Queremos que en la etapa 3 y 5 los led parpadeen sin cambiar de etapa.

Para eso necesitaremos modificar las variables, escribiéndolo así en el archivo **.h**:

Inicial

```
//Temporizadores, comparadores, tiempo.
bool T[1];
unsigned long pret[1]={0};
unsigned int tempst[1]={2000};
```

→

Modificado

```
//Temporizadores, comparadores, tiempo.
bool T[1];
unsigned long pret[3]={0,0,0};
unsigned int tempst[3]={2000,500,1500};
```

Sumando 2 al pret para cada función y tempst, añadiendo el tiempo de parpadeo en este caso 0'5 segundos y 1'5 segundos.

Aplicar la función en la parte de acciones, en la etapa que corresponda para que parpadear, la función se encuentra en el archivo **.h** al final.

Completar las partes:

```
//Transiciones.
for(g=0;g<2;g++){
for(f=5;f>=0;f--){
if(g==0||(g&&E[f])==1){
switch(f){
case 0 :
E[f] = etapand (E[f],E[5],i[2],E[3],i[2],E[f+1],E[f+2]);
break;
case 1 :
E[f] = etapalin (E[f],E[f-1],i[0],E[f+1]);
break;
case 2 :
E[f] = etapa (E[f],E[f-1],i[2],E[4],i[2],E[f+1],E[5]);
break;
case 3 :
E[f] = etapalin (E[f],E[f-1],T[0],E[0]);
break;
case 4 :
E[f] = etapalin (E[f],E[0],i[1],E[2]);
break;
case 5 :
E[f] = etapalin (E[f],E[2],T[0],E[0]);
break;}
}
}
}
```

←Como van a convergir 2 etapas a la vez hay que indicárselo.

Se puede usar la función etapalin para simplificar.

```
//Acciones.
for(f=0;f<6;f++){
if (E[f]==1){
switch(f){
case 0 :

break;
case 1 :
M[0]=1;
break;
case 2 :
if(fe[f]==1) pret[0] = pretemp (tempst[0]);
T[0] = temp (pret[0]);
break;
case 3 :
if(fe[f]==1) M[0]=3;
if(M[0]==3) pret[1] = pretemp (tempst[1]);
M[0] = parpadeo(pret[1], tempst[1]);
break;
case 4 :
M[1]=1;
break;
case 5 :
if(fe[f]==1) M[2]=3;
if(M[2]==3) pret[2] = pretemp (tempst[1]);
M[2] = parpadeo(pret[2], tempst[2]);
break;}
}
}
```

←Aplicando la función en el case 3 y 5, haremos que parpadee.

← En este caso el led 2 estará 0’5 (s) en alto y 1’5 (s) en bajo.

Y ya estará listo para funcionar.