Desarrollo y diseño de interfaz para proyectos

@ZGZMakerSpace

Con la colaboración de:

GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN

FECYT FUNDACIÓN ESPAÑOLA PARA LA CIENCIA Y LA TECNOLOGÍA

Make It Special - Talleres

Programa de Talleres

Jueves – 18h – 19:30h

- 06/08/2020 - Cómo empezar a definir una idea?
- 13/08/2020 - Prototipado rápido, diseño modular y paramétrico.
- 20/08/2020 - Electrónica básica para proyectos.
- 27/08/2020 - **Desarrollo y diseño de interfaz de usuario.**
- 03/09/2020 - **Documentación final y publicación.**

Hackaton

- Inscripciones abiertas hasta el 18 de Septiembre
- Desarrollo del proyecto y documentación – durante Agosto hasta el 25 de Septiembre
- **Entrega de Documentación - 25 de Septiembre**
- Presentaciones **Hackaton – 3 de Octubre**

Bases del Hackaton

Idea

- Diseñar un menú
- Cómo programar un Menú

Electrónica

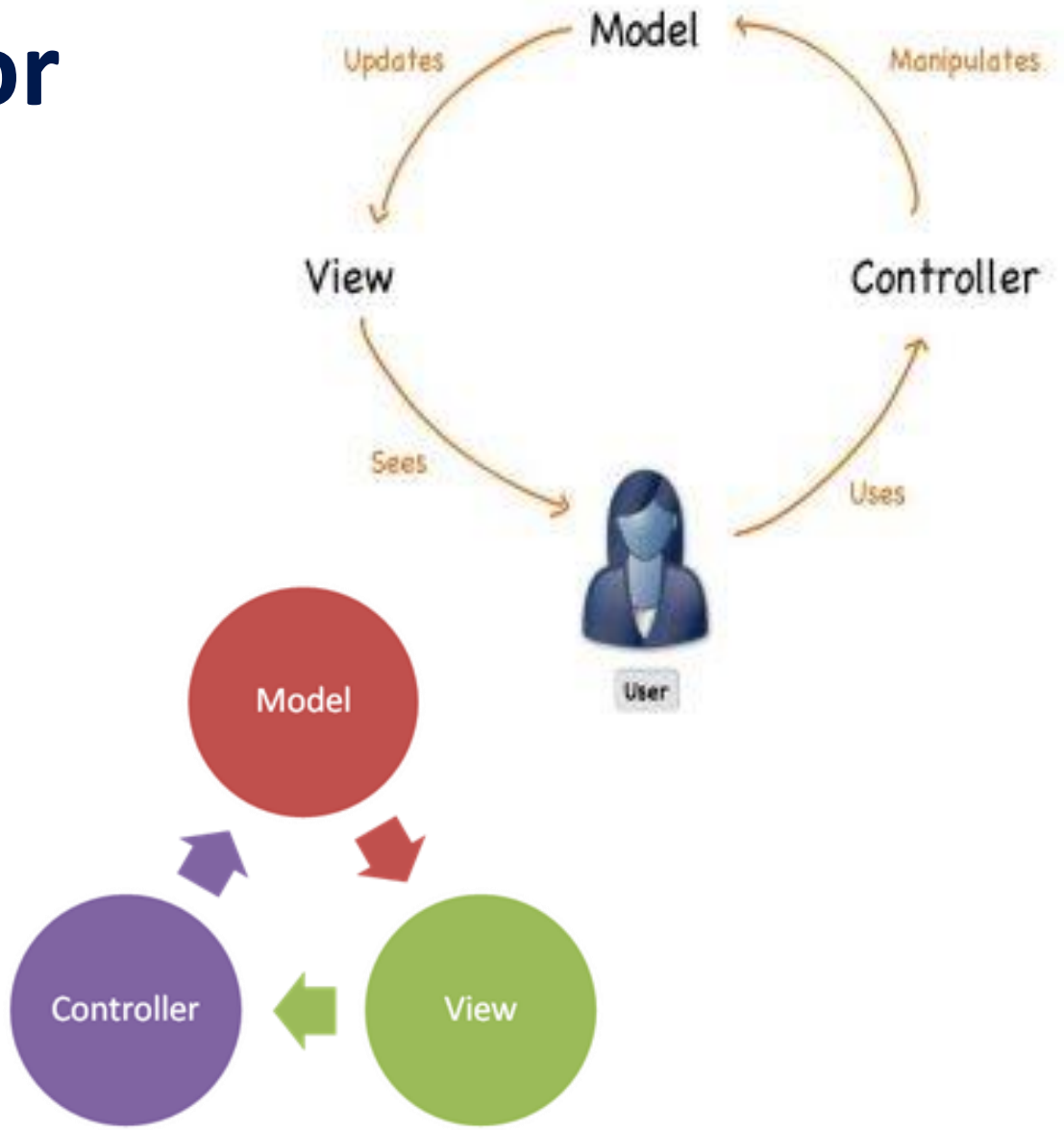
- Arduino + LCD + Encoder

Programación

- Código de programación.
- Estructura de menú

Modelo Visor Controlador

- **Vista** -> todos aquellos componentes que nos proporcionen información para conocer el estado de un sistema y elegir una acción a ejecutar.
- **Controladores** -> son los componentes con los que ejecutaremos la acción.
- **Modelo** -> es el programa que ofrece una respuesta o desarrolla el siguiente proceso entre la ejecución de una acción y la siguiente.



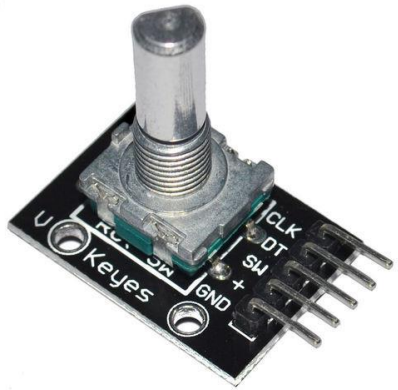
MVC por partes

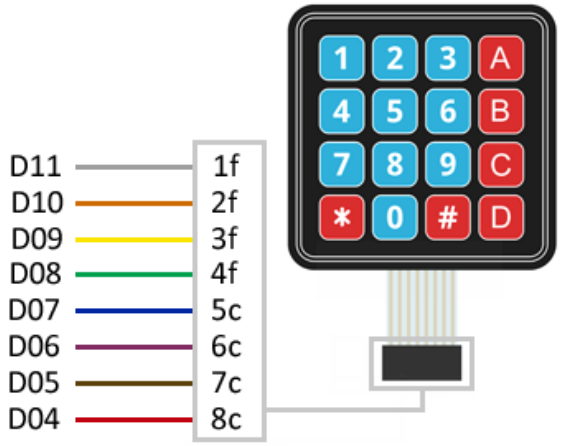
Visor

- Pantalla LCD
- Pantalla TFT
- Puerto Serie

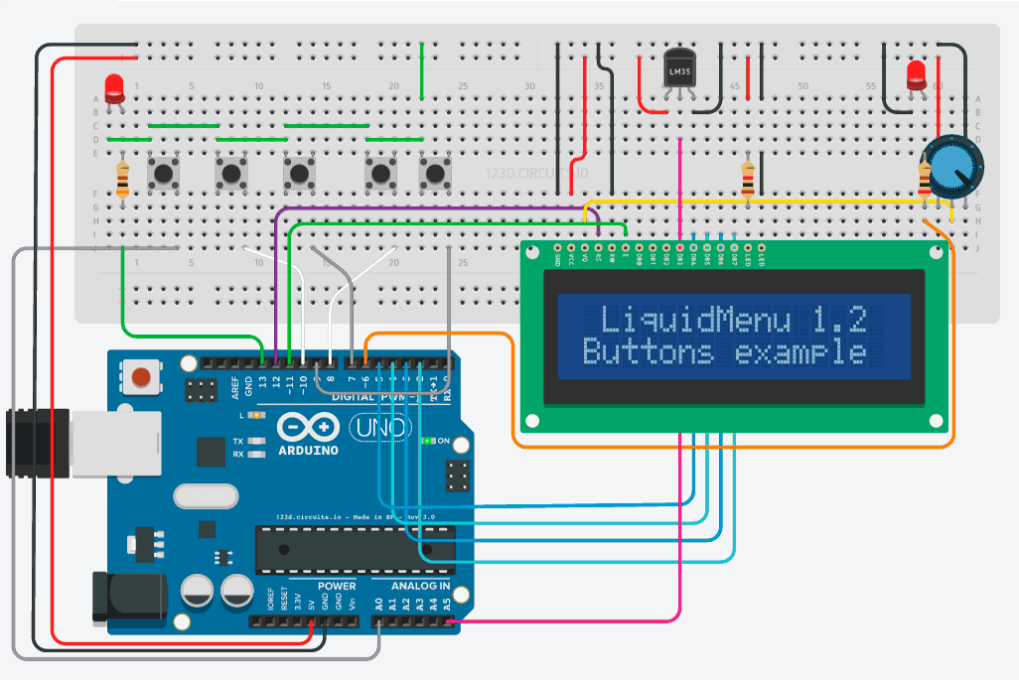
Controlador

- Botonera
- Encoder Rotatorio
- Teclado numérico
- Puerto Serie

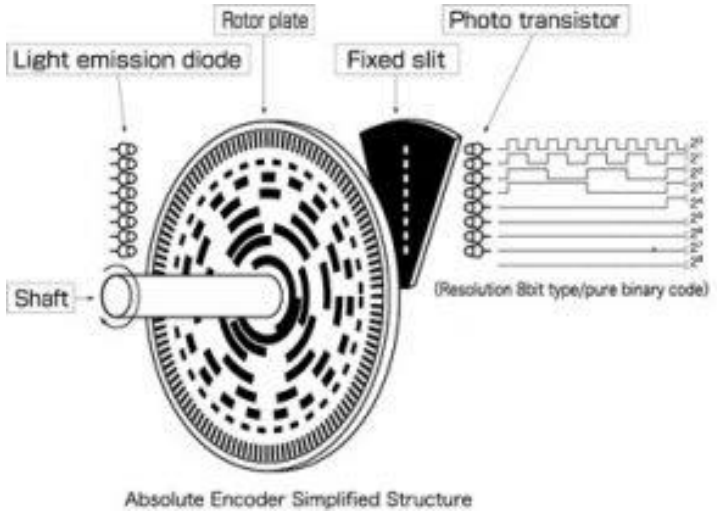




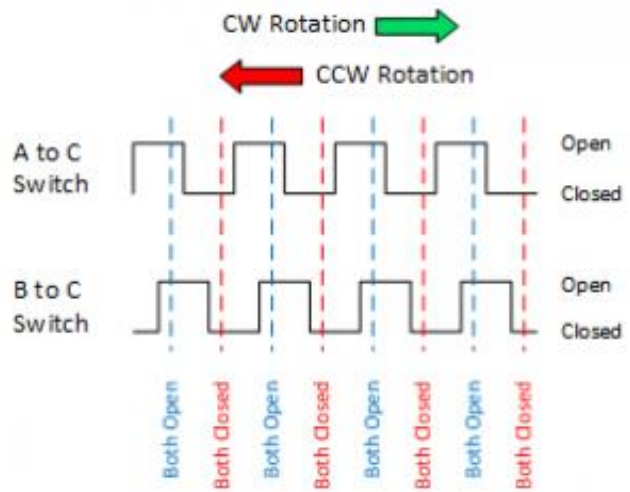
Modelo visor controlador por partes



Introducción al Encoder Rotatorio

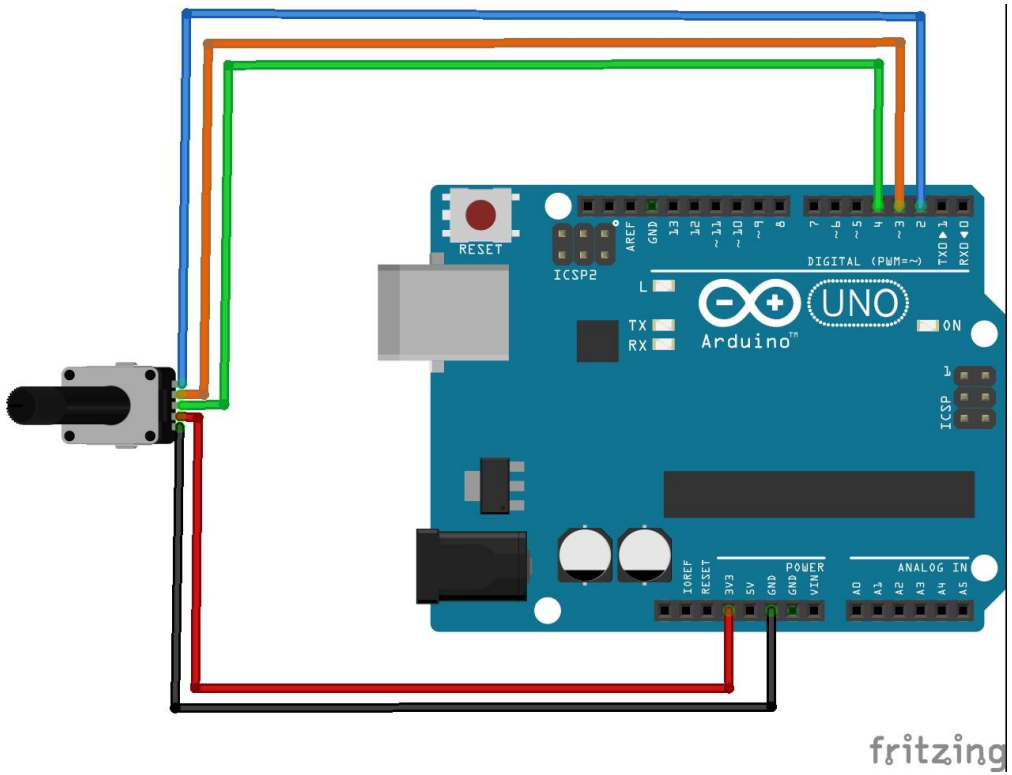


Absolute Encoder Simplified Structure



CUIDADO: NO CONFUNDIR POTENCIÓMETRO CON ENCODER

Conexión Arduino + Encoder

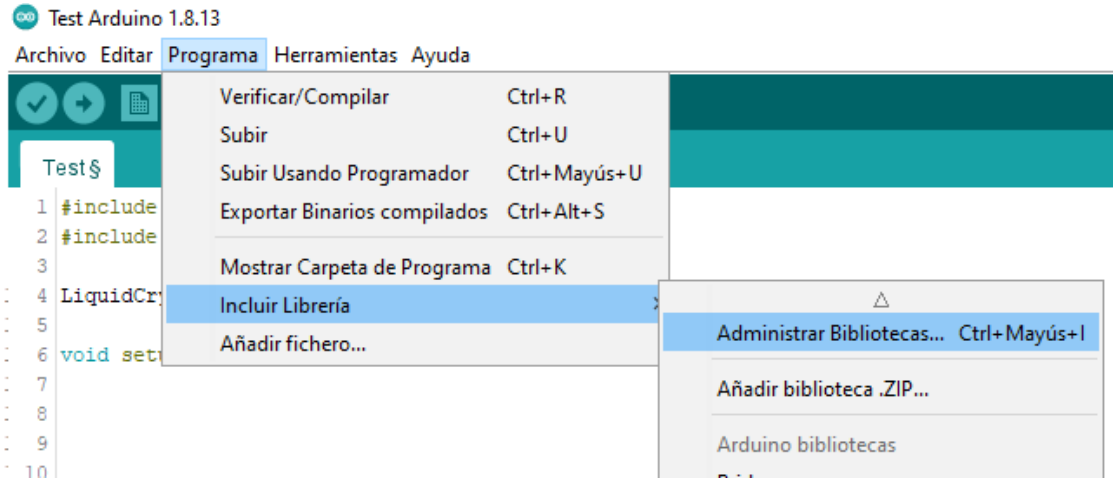


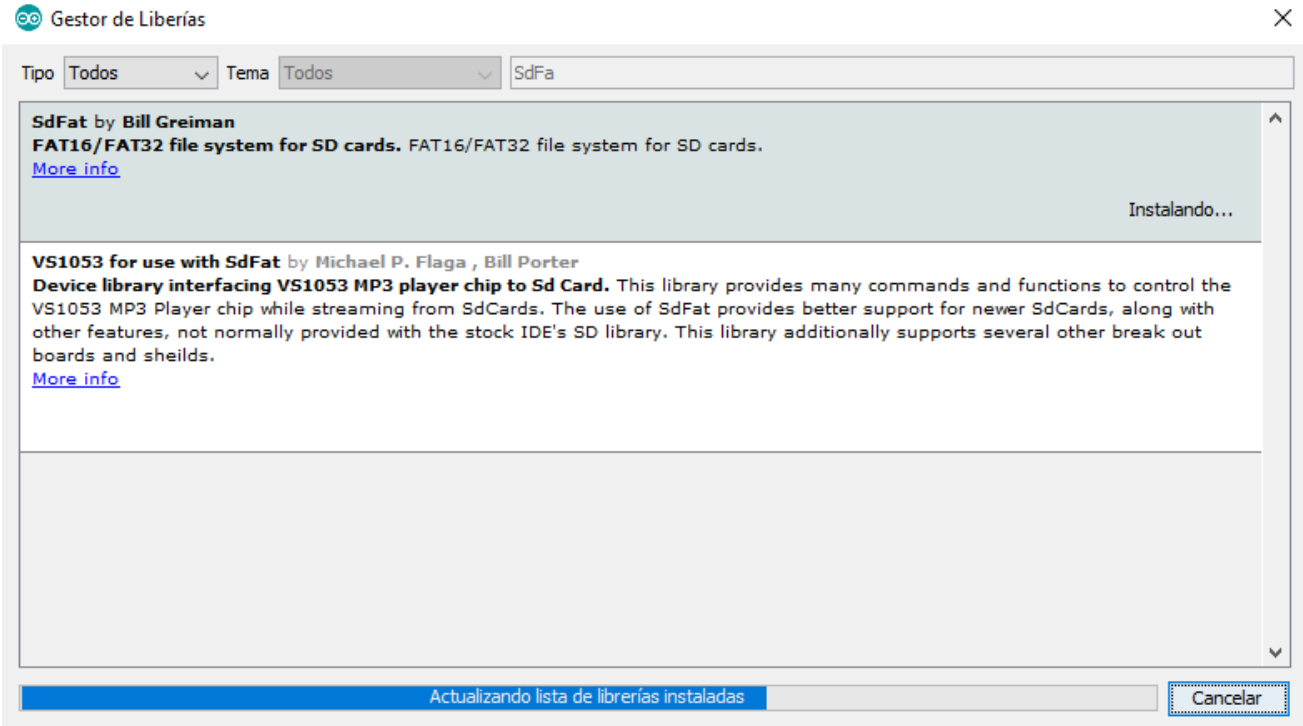
Conexión Encoder

- **CLOCK** –> Señal de reloj en el que se realiza cada medición.
- **DT** –> Señal de datos en el que se detecta un cambio de movimiento.
- **SW** –>Esta señal solamente nos proporciona la presión sobre el botón

Gestor de librerías de Arduino

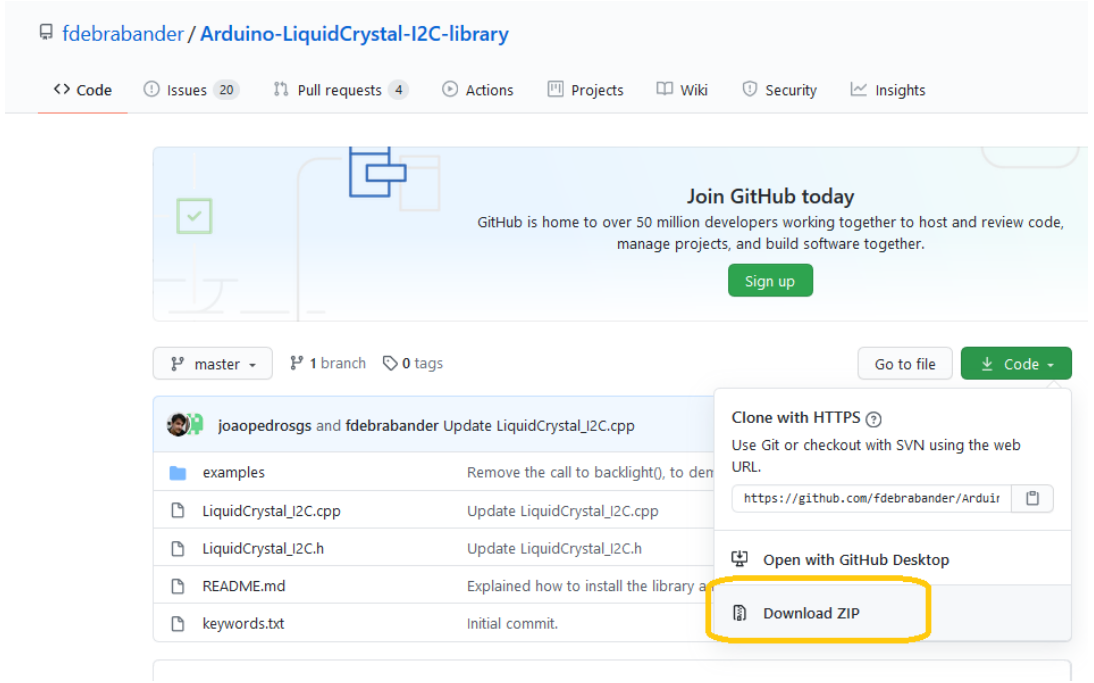
- Gestor de librerías
- Importar Librerías Arduino

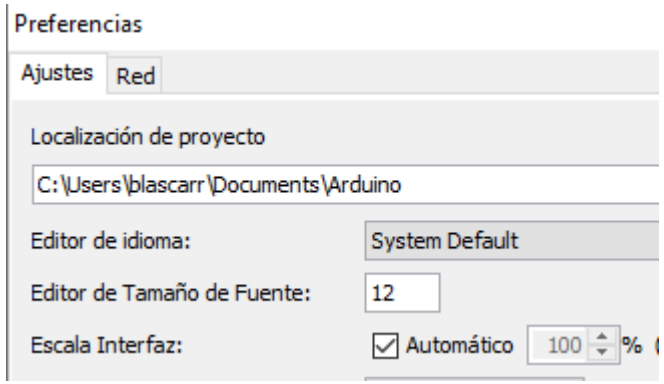


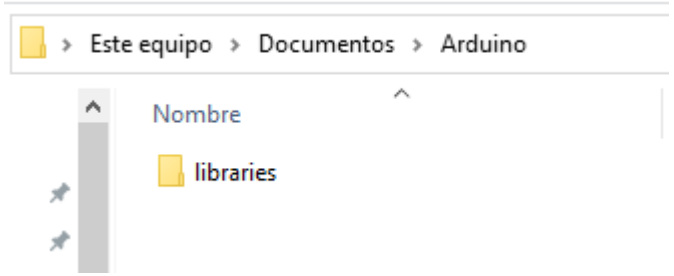


Instalar librerías externas

- Descargar librería
- Extraer el archivo comprimido
- Copiar en la ruta de librerías de Arduino







Programación de un Encoder Rotatorio

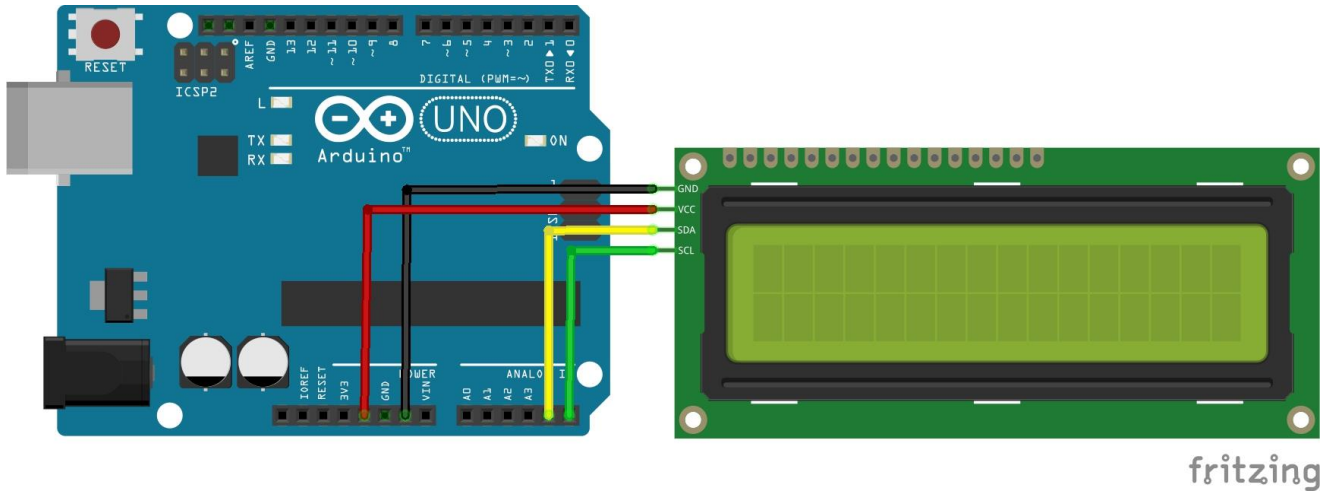
```
#include <Encoder.h>
Encoder myEnc(3, 4);


void setup() {
  Serial.begin(9600);
  Serial.println("Basic Encoder Test:");
}

long oldPosition  = -999;

void loop() {
  long newPosition = myEnc.read();
  if (newPosition != oldPosition) {
    oldPosition = newPosition;
    Serial.println(newPosition);
  }
}
```

La manera más fácil de programar un Encoder es con una librería_Encoder .

- Solo hay que definir los pines a los que se conectan los pines del Encoder. (3 y 4)
- Valor incremental.
- Se puede presionar el Encoder con una nueva señal.

Enlace para programar Encoders

Conexión Arduino LCD



[Conexión por I2C]

- SDA – Pin A4
- SCL – Pin A5
- GND
- VCC

Programación de una pantalla LCD

```
#include <Wire.h>  // Comes with Arduino IDE
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x3f,16,2);

void setup()
{
  Serial.begin(9600);
  Serial.println("Goodnight Moon");
  lcd.init();
  //lcd.begin(16,2);   // initialize the lcd for 16 chars 2 lines, turn on backlight

  lcd.backlight(); // finish with backlight on

  lcd.setCursor(0,0);
  lcd.print("Hello, world!");
}

void loop()
{

}
```

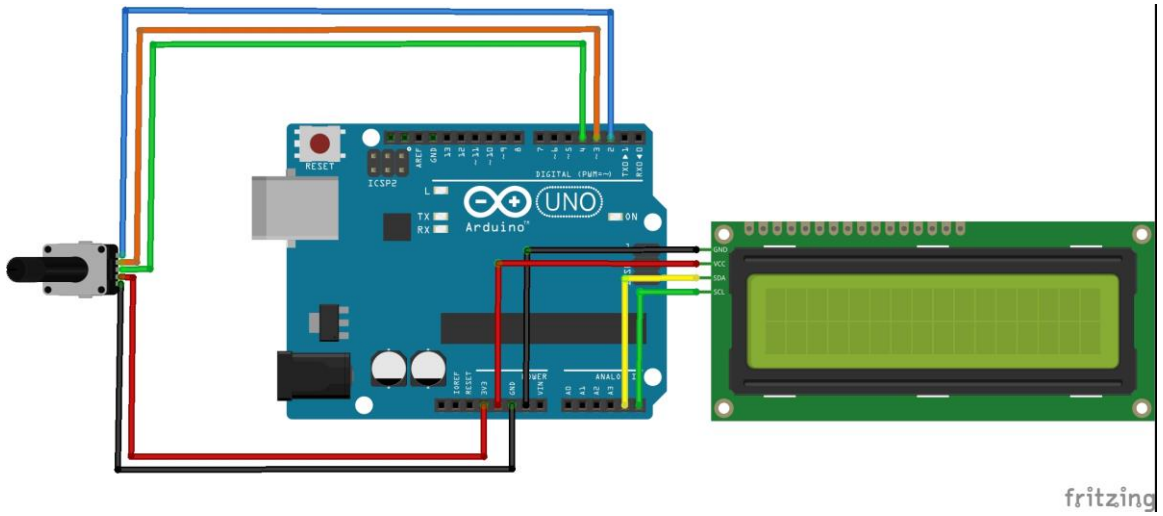
Para programar una pantalla LCD se necesita descargar e instalar la librería LiquidCrystal_I2C

- **Dirección I2C**
- **Numero de columnas**
- **Numero de filas**

La dirección I2C más común es **0x3f**

No olvidar encender el Backlight.

Arduino + Encoder + LCD



MVC_Basic §

```
#include <Wire.h>
#include "Encoder.h"
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd( 0x3f,16,2 );
EncoderMenu myEnc( 3,4 );

long oldPosition  = -999;

void setup() {
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();
}

void loop() {
  long newPosition = myEnc.read();
  if (newPosition != oldPosition) {
    oldPosition = newPosition;
    lcd.print( newPosition );
  }
}
```

Librerías para el Modelo

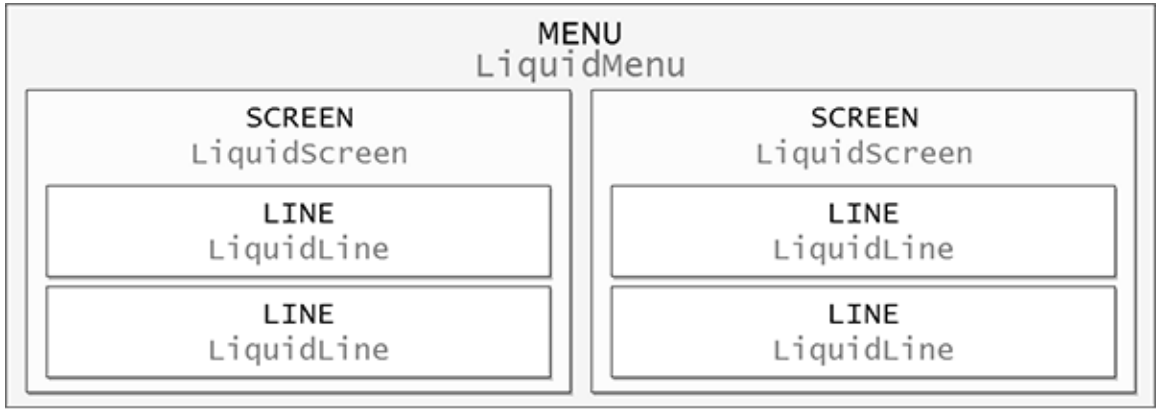
- El **Modelo** es un formato en el que la **vista** (Pantalla) y el **controlador** (Encoder) se comunican para realizar acciones de tal forma que **el usuario pueda interactuar con el proyecto** , ejecutar acciones, configurar opciones, etc...

Librerías compatibles con el modelo para **Arduino**

- LiquidMenu – No integra las acciones de visor – controlador.
- ArduinoMenu – Integra gran variedad de visores y controladores, pero es más complejo.

Programación del Modelo LiquidMenu

- Este formato se basa en declarar por separado un **menú principal**, **submenús** vinculados y **acciones a cada opción** que se definen en cada **Menú**.



Programación del Modelo LiquidMenu

LiquidMenu_Navigation §

```
1  #include <Wire.h>
2  #include <LiquidCrystal_I2C.h>
3  #include "Encoder.h"
4  #include <LiquidMenu.h>
5
6  LiquidCrystal_I2C lcd(0x3f,16,2);
7  Encoder rotary( 3,4 );
8
9  char* playermenu[] = {"Play", "Next", "Prev", "Stop", "Resume"};
10 int npointer = 0;
11 int* pointermenu = &npointer;
12
13 char uptext[] = "Play  ";
14 char *uptr = &(uptext[0] );
15
16 char downtext[] = "Next   ";
17 char *dptr = &(downtext[0] );
18
19
20
21 LiquidLine upline (3, 0, uptr );
22 LiquidLine downline (3, 1, dptr );
23
24 LiquidScreen screenplayer( upline, downline );
25
26 LiquidMenu menu(lcd);
27
```

- Las opciones del menú las almacenamos en una lista.
- Como la lista no cabe en dos filas necesitamos una variable que defina el puntero.
- Punteros dinámicos para las dos filas.
- Primero se definen dos líneas **LiquidLine** para las dos filas, con sus punteros correspondientes.
- Acto seguido se define el **Screen**.
- Y por último se declara la instancia **Menu**.

Programación del Modelo LiquidMenu

```
102 void setup() {
103   Serial.begin(9600);
104   lcd.init();
105   lcd.backlight();
106
107   menu.add_screen( screenplayer );
108
109   upline.attach_function(1, line_fn );
110   downline.attach function(1, line fn);
111   menu.set_focusSymbol(Position::LEFT, rFocus);
112   menu.set_focusPosition(Position::LEFT);
113
114
115
116
117 }
118
119 void loop() {
120   rotary.read();
121   menu.update();
122 }
```

- En el **Setup** se definen las relaciones entre los elementos.
- **IMPORTANTE :** Se adjuntan con el método “**attach_function**” las acciones del menú con una función declarada a parte para ejecutar al seleccionar una opción en el menú.
- En el bucle se ha de actualizar el menú, como, leer el encoder y visualizar en la pantalla los cambios.

Programación del Modelo ArduinoMenu

```
1
2   #include <Wire.h>
3   #include <LiquidCrystal_I2C.h>//F. Malpartida LCD's driver
4   #include <menu.h>//menu macros and objects
5   #include <menuIO/lcdOut.h>//malpartidas lcd menu output
6   #include <menuIO/serialIn.h>//Serial input
7   #include <menuIO/encoderIn.h>//quadrature encoder driver and fake stream
8   #include <menuIO/keyIn.h>//keyboard driver and fake stream (for the encoder button)
9   #include <menuIO/chainStream.h>// concatenate multiple input streams (this allows adding a button to the encoder)
10
11  using namespace Menu;
12  LiquidCrystal_I2C lcd(0x3f,16,2);
13
14  // Encoder ////////////////////////////////////
15  #define encA 2
16  #define encB 3
17  //this encoder has a button here
18  #define encBtn 4
19
20  encoderIn<encA,encB> encoder;//simple quad encoder driver
21  #define ENC_SENSIVITY 4
22  encoderInStream<encA,encB> encStream(encoder,ENC_SENSIVITY);// simple quad encoder fake Stream
23
24  //a keyboard with only one key as the encoder button
25  keyMap encBtn_map[]={{-encBtn,defaultNavCodes[enterCmd].ch}};//negative pin numbers use internal pull-up, this is on when low
26  keyIn<1> encButton(encBtn_map);//1 is the number of keys
27
28  serialIn serial(Serial);
29
```

Módulos del menú

Pantalla LCD

Encoder

Configuración del Encoder más el botón de presión Switch.

Modelo ArduinoMenu - Inputs

```
//input from the encoder + encoder button + serial
menuIn* inputsList[]={&encStream,&encButton,&serial};
chainStream<3> in(inputsList);//3 is the number of inputs
```

Inputs – Encoder + Boton + Puerto Serie

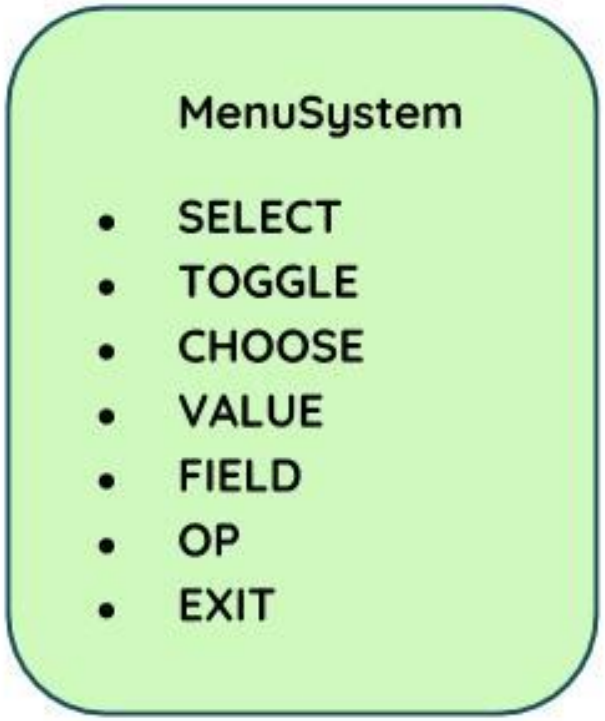
Modelo ArduinoMenu - Outputs

```
MENU_OUTPUTS(out,MAX_DEPTH
  ,LCD_OUT(lcd,{0,0,16,2})
  ,NONE
);
```

Outputs – Pantalla LCD

Modelo ArduinoMenu

Acciones de Menú



Acciones asociadas a opciones del Menú

- **SELECT ->** Selecciona la opción que hemos escogido y ejecuta una función.
- **TOGGLE ->** Modo palanca, solo permite 2 opciones activado o desactivado.
- **CHOOSE ->** Elige de entre un conjunto de opciones.
- **VALUE ->** Selecciona y modifica el valor de una variable asociado a este campo.
- **FIELD** -> Crea un campo que modifica una variable definida en un intervalo.
- **OP ->** Opciones.
- **EXIT ->** Salida del menu, para ir hacia otro

Modelo ArduinoMenu
Definición de un Modelo Menu

```
MENU(mainMenu,"Main menu",doNothing,noEvent,wrapStyle
  ,OP("Op1",action1,anyEvent)
  ,OP("Op2",action2,enterEvent)
  //,SUBMENU(togOp)
  ,FIELD(test,"Test","%",0,100,10,1,doNothing,noEvent,wrapStyle)
  ,SUBMENU(subMenu)
  ,SUBMENU(setLed)
  ,OP("LED On",myLedOn,enterEvent)
  ,OP("LED Off",myLedOff,enterEvent)
  ,SUBMENU(selMenu)
  ,SUBMENU(chooseMenu)
  ,OP("Alert test",doAlert,enterEvent)
  ,EDIT("Hex",buf1,hexNr,doNothing,noEvent,noStyle)
  ,EXIT("<Back")
);
```

Definición de un Modelo Menú

Cada acción de menú tiene un constructor definido.

Modelo ArduinoMenu
Declaración del modelo de navegación

Modelo Menú + Entradas + Salidas

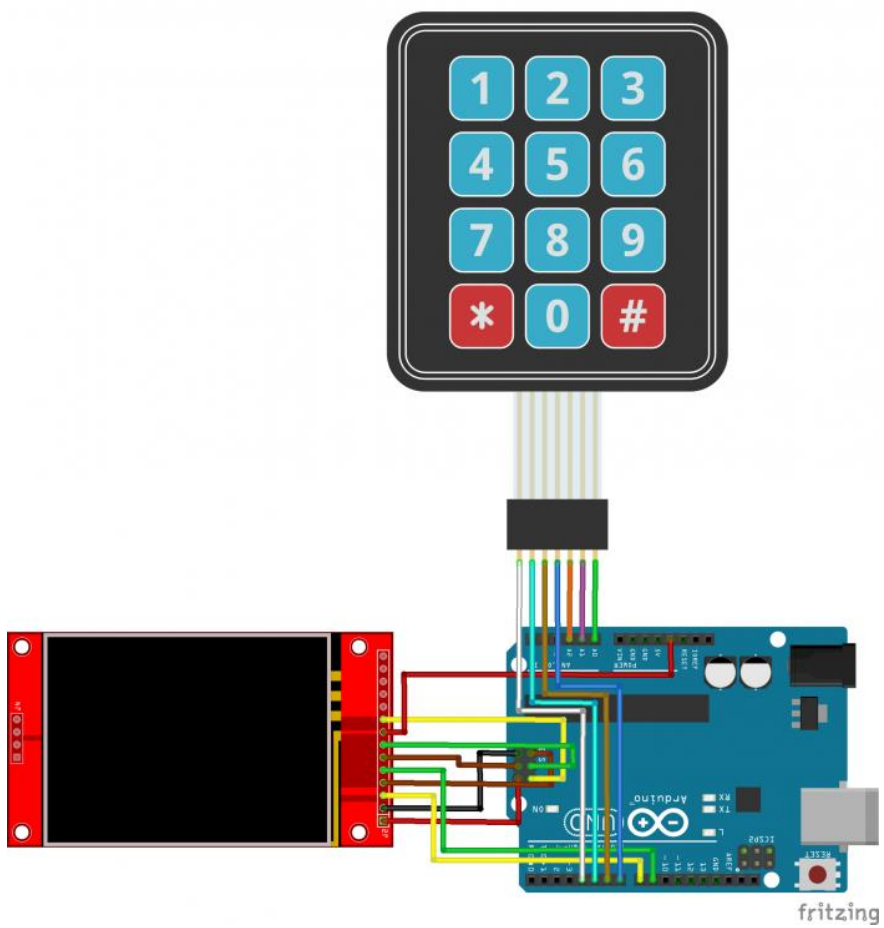
```
NAVROOT(nav,mainMenu,MAX_DEPTH,in,out);//the navigation root object
```

*__MAX_DEPTH__ es el número de submenús.

```
void setup() {

  Serial.begin(115200);
  encoder.begin();
  lcd.init();
  lcd.backlight();
  lcd.begin(16,2);
  nav.idleTask=idle;//point a function to be used when menu is suspended
  mainMenu[1].enabled=disabledStatus;
  nav.showTitle=false;

}

void loop() {
  nav.poll();
}
```

- Inicialización de pantalla y variables iniciales en el **Setup**
- En el bucle, solo se encuentra la actualización del Menú definida como **nav.poll()**

LiquidMenu vs ArduinoMenu



- La extensión a otros modelos de proyecto con un visor o un controlador diferente se realiza mejor con **ArduinoMenu**.
- **LiquidMenu** es más simple, pero no soporta las integraciones de entradas (Controlador) y Salidas (Visor) Hay que gestionarlas por separado.

Gracias por participar en nuestro Taller.

Make It Special

makeitspecial@ibercivis.es
www.makeitspecial.ibercivis.es