Proyecto Final Arquitectura de Software

1.0

Generado por Doxygen 1.11.0

Chapter 1

Índice jerárquico

1.1 Jerarquía de clases

Este listado de herencia está ordenado de forma general pero no está en orden alfabético estricto:

Context . . . . . ??
State . . . . . ??
- ConfigIsotopeState . . . . . ??
- ConfigRangeState . . . . . ??
- ConfigState . . . . . ??
- ErrorSampleState . . . . . ??
- InitState . . . . . ??
- MenuState . . . . . ??
- SamplingState . . . . . ??
- TestState . . . . . ??
- WaitingSampleState . . . . . ??

Chapter 2

Índice de clases

2.1 Lista de clases

Lista de clases, estructuras, uniones e interfaces con breves descripciones:

ConfigIsotopeState
: Clase que maneja el estado de configuración del isótopo en la interfaz . . . . . **??**

ConfigRangeState
: Clase que maneja el estado de configuración de los límites en la interfaz . . . . . **??**

ConfigState
: Clase que representa el estado de configuración del sistema . . . . . **??**

Context . . . . . **??**

ErrorSampleState
: Clase que representa el estado de error en el muestreo . . . . . **??**

InitState
: Clase que representa el estado de inicialización del sistema . . . . . **??**

MenuState
: Clase que representa el estado del menú del sistema . . . . . **??**

SamplingState
: Clase que maneja el estado de muestreo en la máquina de estados . . . . . **??**

State
: Clase base abstracta para los estados de la máquina de estados . . . . . **??**

TestState
: Clase que maneja el estado de prueba en la interfaz . . . . . **??**

WaitingSampleState
: Clase que maneja el estado de espera antes de iniciar el muestreo . . . . . **??**

Chapter 3

Índice de archivos

3.1 Lista de archivos

Lista de todos los archivos con breves descripciones:

sketch/config_state.h
: Declaración de la clase ConfigState y sus miembros . . . ??

sketch/config_state.ino . . . ??

sketch/configIsotope_state.h
: Definición de la clase ConfigIsotopeState para la configuración del isótopo en la interfaz . . . ??

sketch/configIsotope_state.ino . . . ??

sketch/configRange_state.h
: Definición de la clase ConfigRangeState para la configuración de límites en la interfaz . . . ??

sketch/configRange_state.ino . . . ??

sketch/context.h
: Definición de la clase Context para la gestión del estado de la aplicación . . . ??

sketch/context.ino . . . ??

sketch/custom_char.h
: Definiciones de caracteres personalizados para el LCD . . . ??

sketch/custom_char.ino . . . ??

sketch/errorSample_state.h
: Definición de la clase `ErrorSampleState`, que representa el estado de error en el muestreo ??

sketch/errorSample_state.ino . . . ??

sketch/init_state.h
: Declaración de la clase InitState y sus miembros . . . ??

sketch/init_state.ino . . . ??

sketch/menu_state.h
: Declaración de la clase MenuState y sus miembros . . . ??

sketch/menu_state.ino . . . ??

sketch/pins.h
: Declaración de pines y funciones de inicialización para el sistema . . . ??

sketch/pins.ino . . . ??

sketch/sampling_state.h
: Definición de la clase SamplingState para gestionar el estado de muestreo . . . ??

sketch/sampling_state.ino . . . ??

sketch/sketch.ino . . . ??

sketch/state.h . . . ??

sketch/test_state.h
: Definición de la clase TestState para gestionar una ventana de prueba en la interfaz . . . ??

sketch/test_state.ino . . . ??

sketch/waitingSample_state.h . . . ??

sketch/waitingSample_state.ino . . . ??

Chapter 4

Documentación de clases

4.1 Referencia de la clase ConfigIsotopeState

Clase que maneja el estado de configuración del isótopo en la interfaz.

`#include <configIsotope_state.h>`

Diagrama de herencia de ConfigIsotopeState

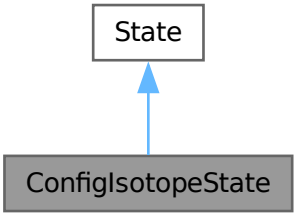
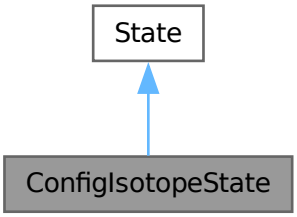


Diagrama de colaboración de ConfigIsotopeState:



Métodos públicos

- ConfigIsotopeState (LiquidCrystal_I2C &lcd)
  Constructor que inicializa el LCD y el índice del menú.
- void handleUp (Context ∗context) override
  Maneja la acción del botón "Up".
- void handleDown (Context ∗context) override
  Maneja la acción del botón "Down".
- void handleSelect (Context ∗context) override
  Maneja la acción del botón "Select".
- void handleBack (Context ∗context) override
  Maneja la acción del botón "Back".
- void displayMenu (Context ∗context) override
  Muestra el menú en la pantalla LCD.

Métodos públicos heredados de State

- virtual ∼State ()

Métodos privados

- void initializeLcd ()
  Inicializa la pantalla LCD.
- void printLogo ()
  Imprime el logo en la pantalla LCD.

Atributos privados

- LiquidCrystal_I2C & lcd
  Referencia al objeto LCD.
- int currentIndex
  Índice del elemento seleccionado en el menú.

Atributos estáticos privados

- static const char ∗ menuItems [ ]
  Elementos del menú para selección de isótopos.
- static const int menuLength
  Longitud del menú.

4.1.1 Descripción detallada

Clase que maneja el estado de configuración del isótopo en la interfaz.

Esta clase gestiona las acciones y la visualización para la selección de isótopos en la pantalla LCD. Hereda de la clase base `State` y proporciona implementaciones específicas para el manejo de botones y la visualización en el LCD.

4.1.2 Documentación de constructores y destructores

4.1.2.1 ConfigIsotopeState()

```
ConfigIsotopeState::ConfigIsotopeState (
            LiquidCrystal_I2C & lcd)
```

Constructor que inicializa el LCD y el índice del menú.

Parámetros

lcd	Referencia al objeto LiquidCrystal_I2C.

4.1.3 Documentación de funciones miembro

4.1.3.1 displayMenu()

```
void ConfigIsotopeState::displayMenu (
            Context * context)  [override], [virtual]
```

Muestra el menú en la pantalla LCD.

Implementa State.

4.1.3.2 handleBack()

```
void ConfigIsotopeState::handleBack (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Back".

Implementa State.

4.1.3.3 handleDown()

```
void ConfigIsotopeState::handleDown (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Down".

Implementa State.

4.1.3.4 handleSelect()

```
void ConfigIsotopeState::handleSelect (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Select".

Implementa State.

4.1.3.5 handleUp()

```
void ConfigIsotopeState::handleUp (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Up".

Implementa State.

4.1.3.6 initializeLcd()

```
void ConfigIsotopeState::initializeLcd ()  [private]
```

Inicializa la pantalla LCD.

4.1.3.7 printLogo()

```
void ConfigIsotopeState::printLogo ()  [private]
```

Imprime el logo en la pantalla LCD.

4.1.4 Documentación de datos miembro

4.1.4.1 currentIndex

```
int ConfigIsotopeState::currentIndex  [private]
```

Índice del elemento seleccionado en el menú.

4.1.4.2 lcd

```
LiquidCrystal_I2C& ConfigIsotopeState::lcd  [private]
```

Referencia al objeto LCD.

4.1.4.3 menuItems

```
const char* ConfigIsotopeState::menuItems[]  [static], [private]
```

Elementos del menú para selección de isótopos.

4.1.4.4 menuLength

```
const int ConfigIsotopeState::menuLength  [static], [private]
```

Longitud del menú.

La documentación de esta clase está generada del siguiente archivo:

- sketch/configIsotope_state.h

4.2 Referencia de la clase ConfigRangeState

Clase que maneja el estado de configuración de los límites en la interfaz.

`#include <configRange_state.h>`

Diagrama de herencia de ConfigRangeState

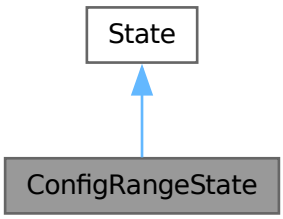
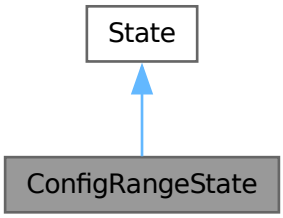


Diagrama de colaboración de ConfigRangeState:



Métodos públicos

- ConfigRangeState (LiquidCrystal_I2C &lcd)

  Constructor que inicializa el LCD, los límites y el estado de configuración.
- void handleUp (Context ∗context) override

  Maneja la acción del botón "Up".
- void handleDown (Context ∗context) override

  Maneja la acción del botón "Down".
- void handleSelect (Context ∗context) override

  Maneja la acción del botón "Select".
- void handleBack (Context ∗context) override

  Maneja la acción del botón "Back".
- void displayMenu (Context ∗context) override

  Muestra el menú de configuración en la pantalla LCD.

Métodos públicos heredados de State

- virtual ~State ()

Métodos privados

- void initializeLcd ()

  Inicializa la pantalla LCD.
- void printLogo ()

  Imprime el logo en la pantalla LCD.

Atributos privados

- LiquidCrystal_I2C & lcd

  Referencia al objeto LCD.
- int lowerLimit

  Límite inferior del rango.
- int upperLimit

  Límite superior del rango.
- bool settingLowerLimit

  Indica si se está configurando el límite inferior o superior.

4.2.1 Descripción detallada

Clase que maneja el estado de configuración de los límites en la interfaz.

Esta clase gestiona las acciones y la visualización para la configuración del rango de valores (límite inferior y superior) en la pantalla LCD. Hereda de la clase base `State` y proporciona implementaciones específicas para el manejo de botones y la visualización en el LCD.

4.2.2 Documentación de constructores y destructores

4.2.2.1 ConfigRangeState()

```
ConfigRangeState::ConfigRangeState (
            LiquidCrystal_I2C & lcd)
```

Constructor que inicializa el LCD, los límites y el estado de configuración.

Parámetros

lcd	Referencia al objeto LiquidCrystal_I2C.

4.2.3 Documentación de funciones miembro

4.2.3.1 displayMenu()

```
void ConfigRangeState::displayMenu (
            Context * context)  [override], [virtual]
```

Muestra el menú de configuración en la pantalla LCD.

Implementa State.

4.2.3.2 handleBack()

```
void ConfigRangeState::handleBack (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Back".

Implementa State.

4.2.3.3 handleDown()

```
void ConfigRangeState::handleDown (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Down".

Implementa State.

4.2.3.4 handleSelect()

```
void ConfigRangeState::handleSelect (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Select".

Implementa State.

4.2.3.5 handleUp()

```
void ConfigRangeState::handleUp (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Up".

Implementa State.

4.2.3.6 initializeLcd()

```
void ConfigRangeState::initializeLcd ()  [private]
```

Inicializa la pantalla LCD.

4.2.3.7 printLogo()

```
void ConfigRangeState::printLogo ()  [private]
```

Imprime el logo en la pantalla LCD.

4.2.4 Documentación de datos miembro

4.2.4.1 lcd

```
LiquidCrystal_I2C& ConfigRangeState::lcd  [private]
```

Referencia al objeto LCD.

4.2.4.2 lowerLimit

```
int ConfigRangeState::lowerLimit  [private]
```

Límite inferior del rango.

4.2.4.3 settingLowerLimit

```
bool ConfigRangeState::settingLowerLimit  [private]
```

Indica si se está configurando el límite inferior o superior.

4.2.4.4 upperLimit

```
int ConfigRangeState::upperLimit  [private]
```

Límite superior del rango.

La documentación de esta clase está generada del siguiente archivo:

- sketch/configRange_state.h

4.3 Referencia de la clase ConfigState

Clase que representa el estado de configuración del sistema.

`#include <config_state.h>`

Diagrama de herencia de ConfigState

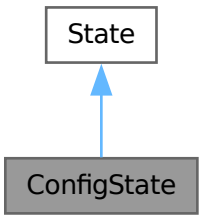
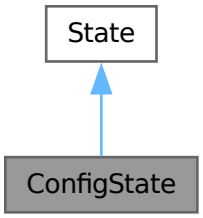


Diagrama de colaboración de ConfigState:



Métodos públicos

- ConfigState (LiquidCrystal_I2C &lcd)
  Constructor de la clase ConfigState.
- void handleUp (Context ∗context) override
  Maneja el evento de pulsar el botón hacia arriba.
- void handleDown (Context ∗context) override
  Maneja el evento de pulsar el botón hacia abajo.
- void handleSelect (Context ∗context) override
  Maneja el evento de pulsar el botón de selección.
- void handleBack (Context ∗context) override
  Maneja el evento de pulsar el botón de retroceso.
- void displayMenu (Context ∗context) override
  Muestra el menú de configuración en el LCD.

Métodos públicos heredados de State

- virtual ~State ()

Métodos privados

- void initializeLcd ()

  Inicializa las variables y pantalla.
- void printLogo ()

  Imprime el logo en la pantalla LCD.

Atributos privados

- int currentIndex

  Índice del ítem de menú actualmente seleccionado.
- LiquidCrystal_I2C & lcd

  Referencia al objeto LiquidCrystal_I2C para el control del LCD.

Atributos estáticos privados

- static const char * menuItems [ ]

  Arreglo de cadenas que contiene los ítems del menú de configuración.
- static const int menuLength

  Longitud del menú de configuración.

4.3.1 Descripción detallada

Clase que representa el estado de configuración del sistema.

4.3.2 Documentación de constructores y destructores

4.3.2.1 ConfigState()

```
ConfigState::ConfigState (
            LiquidCrystal_I2C & lcd)
```

Constructor de la clase ConfigState.

Parámetros

lcd	Referencia a un objeto LiquidCrystal_I2C para el control del LCD.

4.3.3 Documentación de funciones miembro

4.3.3.1 displayMenu()

```
void ConfigState::displayMenu (
            Context * context)  [override], [virtual]
```

Muestra el menú de configuración en el LCD.

Parámetros

context	Puntero al contexto de la máquina de estados.

Implementa State.

4.3.3.2 handleBack()

```
void ConfigState::handleBack (
            Context * context)  [override], [virtual]
```

Maneja el evento de pulsar el botón de retroceso.

Parámetros

context	Puntero al contexto de la máquina de estados.

Implementa State.

4.3.3.3 handleDown()

```
void ConfigState::handleDown (
            Context * context)  [override], [virtual]
```

Maneja el evento de pulsar el botón hacia abajo.

Parámetros

context	Puntero al contexto de la máquina de estados.

Implementa State.

4.3.3.4 handleSelect()

```
void ConfigState::handleSelect (
            Context * context)  [override], [virtual]
```

Maneja el evento de pulsar el botón de selección.

Parámetros

context	Puntero al contexto de la máquina de estados.

Implementa State.

4.3.3.5 handleUp()

```
void ConfigState::handleUp (
            Context * context)  [override], [virtual]
```

Maneja el evento de pulsar el botón hacia arriba.

Parámetros

context	Puntero al contexto de la máquina de estados.

Implementa State.

4.3.3.6 initializeLcd()

```
void ConfigState::initializeLcd ()  [private]
```

Inicializa las variables y pantalla.

4.3.3.7 printLogo()

```
void ConfigState::printLogo ()  [private]
```

Imprime el logo en la pantalla LCD.

4.3.4 Documentación de datos miembro

4.3.4.1 currentIndex

```
int ConfigState::currentIndex  [private]
```

Índice del ítem de menú actualmente seleccionado.

4.3.4.2 lcd

```
LiquidCrystal_I2C& ConfigState::lcd  [private]
```

Referencia al objeto LiquidCrystal_I2C para el control del LCD.

4.3.4.3 menuItems

```
const char* ConfigState::menuItems[]  [static], [private]
```

Arreglo de cadenas que contiene los ítems del menú de configuración.

4.3.4.4 menuLength

```
const int ConfigState::menuLength  [static], [private]
```

Longitud del menú de configuración.

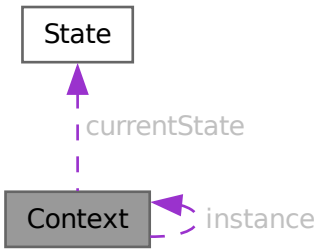
La documentación de esta clase está generada del siguiente archivo:

- sketch/config_state.h

4.4 Referencia de la clase Context

`#include <context.h>`

Diagrama de colaboración de Context:



Métodos públicos

- void setState (State *state)
  Establece el estado actual.
- void handleUp ()
  Maneja la entrada de botón hacia arriba.
- void handleDown ()
  Maneja la entrada de botón hacia abajo.
- void handleSelect ()
  Maneja la entrada de selección de botón.
- void handleBack ()
  Maneja la entrada de botón de retroceso.
- void displayMenu ()
  Establece el límite inferior.
- void setLowerLimit (int lowerLimit)
  Establece el límite inferior del rango de medicion.
- int getLowerLimit () const
  Obtiene el límite inferior del rango de medicion..
- void setUpperLimit (int upperLimit)
  Establece el límite superiordel rango de medicion.
- int getUpperLimit () const
  Obtiene el límite superiordel rango de medicion.
- void setIsotope (const char *isotope)
  Establece el isótopo actual.
- const char * getIsotope () const
  Obtiene el isótopo actual.
- LiquidCrystal_I2C & getLcd ()
  Obtiene el objeto de la pantalla LCD.

Métodos públicos estáticos

- static Context * getInstance ()

  Obtiene la instancia única del Contexto.

Métodos privados

- Context ()

  Constructor privado para implementar el patrón Singleton.

Atributos privados

- State * currentState

  Puntero al estado actual.
- LiquidCrystal_I2C lcd

  Objeto para controlar la pantalla LCD.
- int lowerLimit

  Límite inferior para el rango.
- int upperLimit

  Límite superior para el rango.
- const char * isotope

  Nombre del isótopo actua.

Atributos estáticos privados

- static Context * instance

  Instancia única del Contexto.

4.4.1 Documentación de constructores y destructores

4.4.1.1 Context()

```
Context::Context ()  [private]
```

Constructor privado para implementar el patrón Singleton.

4.4.2 Documentación de funciones miembro

4.4.2.1 displayMenu()

```
void Context::displayMenu ()
```

Establece el límite inferior.

Parámetros

lowerLimit	Valor del límite inferior.

4.4.2.2 getInstance()

```
static Context * Context::getInstance ()  [static]
```

Obtiene la instancia única del Contexto.

Devuelve

Puntero a la instancia del Contexto.

4.4.2.3 getIsotope()

```
const char * Context::getIsotope () const
```

Obtiene el isótopo actual.

Devuelve

Cadena con el nombre del isótopo.

4.4.2.4 getLcd()

```
LiquidCrystal_I2C & Context::getLcd ()
```

Obtiene el objeto de la pantalla LCD.

Devuelve

Referencia al objeto LCD.

4.4.2.5 getLowerLimit()

```
int Context::getLowerLimit () const
```

Obtiene el límite inferior del rango de medicion..

Devuelve

Valor del límite inferior.

4.4.2.6 getUpperLimit()

```
int Context::getUpperLimit () const
```

Obtiene el límite superiordel rango de medicion.

Devuelve

Valor del límite superior.

4.4.2.7 handleBack()

```
void Context::handleBack ()
```

Maneja la entrada de botón de retroceso.

4.4.2.8 handleDown()

```
void Context::handleDown ()
```

Maneja la entrada de botón hacia abajo.

4.4.2.9 handleSelect()

```
void Context::handleSelect ()
```

Maneja la entrada de selección de botón.

4.4.2.10 handleUp()

```
void Context::handleUp ()
```

Maneja la entrada de botón hacia arriba.

4.4.2.11 setIsotope()

```
void Context::setIsotope (
            const char * isotope)
```

Establece el isótopo actual.

Parámetros

isotope	Cadena con el nombre del isótopo.

4.4.2.12 setLowerLimit()

```
void Context::setLowerLimit (
            int lowerLimit)
```

Establece el límite inferior del rango de medicion.

Parámetros

lowerLimit	Valor del límite inferior.

4.4.2.13 setState()

```
void Context::setState (
            State * state)
```

Establece el estado actual.

Parámetros

state	Puntero al nuevo estado.

4.4.2.14 setUpperLimit()

```
void Context::setUpperLimit (
            int upperLimit)
```

Establece el límite superiordel rango de medicion.

Parámetros

upperLimit	Valor del límite superior.

4.4.3 Documentación de datos miembro

4.4.3.1 currentState

```
State* Context::currentState  [private]
```

Puntero al estado actual.

4.4.3.2 instance

```
Context* Context::instance  [static], [private]
```

Instancia única del Contexto.

4.4.3.3 isotope

```
const char* Context::isotope  [private]
```

Nombre del isótopo actua.

4.4.3.4 lcd

```
LiquidCrystal_I2C Context::lcd  [private]
```

Objeto para controlar la pantalla LCD.

4.4.3.5 lowerLimit

```
int Context::lowerLimit  [private]
```

Límite inferior para el rango.

4.4.3.6 upperLimit

```
int Context::upperLimit  [private]
```

Límite superior para el rango.

La documentación de esta clase está generada del siguiente archivo:

- sketch/context.h

4.5 Referencia de la clase ErrorSampleState

Clase que representa el estado de error en el muestreo.

```
#include <errorSample_state.h>
```

Diagrama de herencia de ErrorSampleState

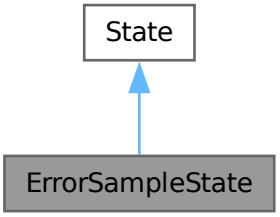
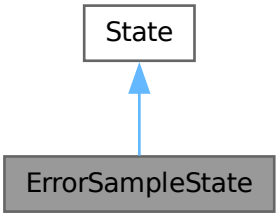


Diagrama de colaboración de ErrorSampleState:



Métodos públicos

- ErrorSampleState (LiquidCrystal_I2C &lcd)

  Constructor de `ErrorSampleState`.
- void handleUp (Context *context) override

  Maneja la entrada de botón hacia arriba.
- void handleDown (Context *context) override

  Maneja la entrada de botón hacia abajo.
- void handleSelect (Context *context) override

  Maneja la entrada de selección de botón.
- void handleBack (Context *context) override

  Maneja la entrada de botón de retroceso.
- void displayMenu (Context *context) override

  Muestra la información del estado de error en la pantalla LCD.

Métodos públicos heredados de State

- virtual ~State ()

Métodos privados

- void initializeLcd ()

  Inicializa la pantalla LCD para el estado de error.
- void printLogo ()

  Imprime el logotipo en la pantalla LCD.

Atributos privados

- LiquidCrystal_I2C & lcd

  Referencia al objeto `LiquidCrystal_I2C`

4.5.1 Descripción detallada

Clase que representa el estado de error en el muestreo.

Hereda de `State` y se encarga de manejar las acciones específicas y la visualización en el LCD cuando ocurre un error en el proceso de muestreo.

4.5.2 Documentación de constructores y destructores

4.5.2.1 ErrorSampleState()

```
ErrorSampleState::ErrorSampleState (
            LiquidCrystal_I2C & lcd)
```

Constructor de `ErrorSampleState`.

Parámetros

lcd	Referencia al objeto `LiquidCrystal_I2C` para controlar la pantalla LCD.

4.5.3 Documentación de funciones miembro

4.5.3.1 displayMenu()

```
void ErrorSampleState::displayMenu (
            Context * context)  [override], [virtual]
```

Muestra la información del estado de error en la pantalla LCD.

Parámetros

context	Puntero al objeto `Context` que gestiona el estado.

Implementa State.

4.5.3.2 handleBack()

```
void ErrorSampleState::handleBack (
            Context * context)  [override], [virtual]
```

Maneja la entrada de botón de retroceso.

Parámetros

context	Puntero al objeto `Context` que gestiona el estado.

Implementa State.

4.5.3.3 handleDown()

```
void ErrorSampleState::handleDown (
            Context * context)  [override], [virtual]
```

Maneja la entrada de botón hacia abajo.

Parámetros

context	Puntero al objeto `Context` que gestiona el estado.

Implementa State.

4.5.3.4 handleSelect()

```
void ErrorSampleState::handleSelect (
            Context * context)  [override], [virtual]
```

Maneja la entrada de selección de botón.

Parámetros

context	Puntero al objeto `Context` que gestiona el estado.

Implementa State.

4.5.3.5 handleUp()

```
void ErrorSampleState::handleUp (
            Context * context)  [override], [virtual]
```

Maneja la entrada de botón hacia arriba.

Parámetros

context	Puntero al objeto `Context` que gestiona el estado.

Implementa State.

4.5.3.6 initializeLcd()

```
void ErrorSampleState::initializeLcd ()  [private]
```

Inicializa la pantalla LCD para el estado de error.

4.5.3.7 printLogo()

```
void ErrorSampleState::printLogo ()  [private]
```

Imprime el logotipo en la pantalla LCD.

4.5.4 Documentación de datos miembro

4.5.4.1 lcd

```
LiquidCrystal_I2C& ErrorSampleState::lcd  [private]
```

Referencia al objeto `LiquidCrystal_I2C`

La documentación de esta clase está generada del siguiente archivo:

- sketch/errorSample_state.h

4.6 Referencia de la clase InitState

Clase que representa el estado de inicialización del sistema.

`#include <init_state.h>`

Diagrama de herencia de InitState

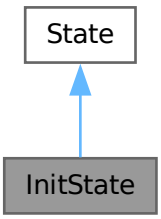
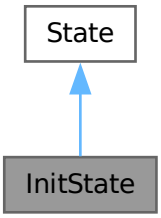


Diagrama de colaboración de InitState:



Métodos públicos

- InitState (LiquidCrystal_I2C &lcd)

  Constructor de la clase InitState.
- void handleUp (Context ∗context) override

  No action needed.
- void handleDown (Context ∗context) override

  No action needed.
- void handleSelect (Context ∗context) override

  Maneja el evento de pulsar el botón de selección.
- void handleBack (Context ∗context) override

  No action needed.
- void displayMenu (Context ∗context) override

  Muestra el menú de inicialización en el LCD.

Métodos públicos heredados de State

- virtual ~State ()

Métodos privados

- void initializeLcd ()

  Inicializa las variables y pantalla.
- void printLogo ()

  Imprime el logo en la pantalla LCD.

Atributos privados

- LiquidCrystal_I2C & lcd

  Referencia al objeto LiquidCrystal_I2C para el control del LCD.

4.6.1 Descripción detallada

Clase que representa el estado de inicialización del sistema.

4.6.2 Documentación de constructores y destructores

4.6.2.1 InitState()

```
InitState::InitState (
            LiquidCrystal_I2C & lcd)
```

Constructor de la clase InitState.

Parámetros

lcd	Referencia a un objeto LiquidCrystal_I2C para el control del LCD.

4.6.3 Documentación de funciones miembro

4.6.3.1 displayMenu()

```
void InitState::displayMenu (
            Context * context)  [override], [virtual]
```

Muestra el menú de inicialización en el LCD.

Parámetros

context	Puntero al contexto de la máquina de estados.

Implementa State.

4.6.3.2 handleBack()

```
void InitState::handleBack (
            Context * context)  [override], [virtual]
```

No action needed.

Implementa State.

4.6.3.3 handleDown()

```
void InitState::handleDown (
            Context * context)  [override], [virtual]
```

No action needed.

Implementa State.

4.6.3.4 handleSelect()

```
void InitState::handleSelect (
            Context * context)  [override], [virtual]
```

Maneja el evento de pulsar el botón de selección.

Parámetros

context	Puntero al contexto de la máquina de estados.

Implementa State.

4.6.3.5 handleUp()

```
void InitState::handleUp (
            Context * context)  [override], [virtual]
```

No action needed.

Implementa State.

4.6.3.6 initializeLcd()

```
void InitState::initializeLcd ()  [private]
```

Inicializa las variables y pantalla.

4.6.3.7 printLogo()

```
void InitState::printLogo ()  [private]
```

Imprime el logo en la pantalla LCD.

4.6.4 Documentación de datos miembro

4.6.4.1 lcd

```
LiquidCrystal_I2C& InitState::lcd  [private]
```

Referencia al objeto LiquidCrystal_I2C para el control del LCD.

La documentación de esta clase está generada del siguiente archivo:

- sketch/init_state.h

4.7 Referencia de la clase MenuState

Clase que representa el estado del menú del sistema.

```
#include <menu_state.h>
```

Diagrama de herencia de MenuState

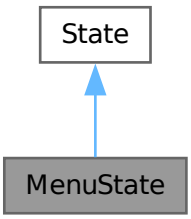
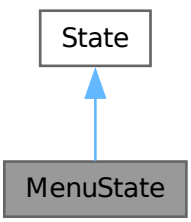


Diagrama de colaboración de MenuState:



Métodos públicos

- MenuState (LiquidCrystal_I2C &lcd)

  Constructor de la clase MenuState.
- void handleUp (Context *context) override

  Maneja el evento de pulsar el botón hacia arriba.
- void handleDown (Context *context) override

  Maneja el evento de pulsar el botón hacia abajo.
- void handleSelect (Context *context) override

  Maneja el evento de pulsar el botón de selección.
- void handleBack (Context *context) override

  Maneja el evento de pulsar el botón de retroceso.
- void displayMenu (Context *context) override

  Muestra el menú principal en el LCD.

Métodos públicos heredados de State

- virtual ~State ()

Métodos privados

- void initializeLcd ()

  Inicializa la pantalla LCD.
- void printLogo ()

  Imprime el logo en la pantalla LCD.

Atributos privados

- int currentIndex

  Índice actual del menú.
- LiquidCrystal_I2C & lcd

  Referencia al objeto LiquidCrystal_I2C para el control del LCD.

Atributos estáticos privados

- static const char * menuItems [ ]

  Elementos del menú.
- static const int menuLength

  Longitud del menú.

4.7.1 Descripción detallada

Clase que representa el estado del menú del sistema.

4.7.2 Documentación de constructores y destructores

4.7.2.1 MenuState()

```
MenuState::MenuState (
            LiquidCrystal_I2C & lcd)
```

Constructor de la clase MenuState.

Parámetros

lcd	Referencia a un objeto LiquidCrystal_I2C para el control del LCD.

4.7.3 Documentación de funciones miembro

4.7.3.1 displayMenu()

```
void MenuState::displayMenu (
            Context * context)  [override], [virtual]
```

Muestra el menú principal en el LCD.

Parámetros

context	Puntero al contexto de la máquina de estados.

Implementa State.

4.7.3.2 handleBack()

```
void MenuState::handleBack (
            Context * context)  [override], [virtual]
```

Maneja el evento de pulsar el botón de retroceso.

Parámetros

context	Puntero al contexto de la máquina de estados.

Implementa State.

4.7.3.3 handleDown()

```
void MenuState::handleDown (
            Context * context)  [override], [virtual]
```

Maneja el evento de pulsar el botón hacia abajo.

Parámetros

context	Puntero al contexto de la máquina de estados.

Implementa State.

4.7.3.4 handleSelect()

```
void MenuState::handleSelect (
            Context * context)  [override], [virtual]
```

Maneja el evento de pulsar el botón de selección.

Parámetros

context	Puntero al contexto de la máquina de estados.

Implementa State.

4.7.3.5 handleUp()

```
void MenuState::handleUp (
            Context * context)  [override], [virtual]
```

Maneja el evento de pulsar el botón hacia arriba.

Parámetros

context	Puntero al contexto de la máquina de estados.

Implementa State.

4.7.3.6 initializeLcd()

```
void MenuState::initializeLcd ()  [private]
```

Inicializa la pantalla LCD.

4.7.3.7 printLogo()

```
void MenuState::printLogo ()  [private]
```

Imprime el logo en la pantalla LCD.

4.7.4 Documentación de datos miembro

4.7.4.1 currentIndex

```
int MenuState::currentIndex  [private]
```

Índice actual del menú.

4.7.4.2 lcd

```
LiquidCrystal_I2C& MenuState::lcd  [private]
```

Referencia al objeto LiquidCrystal_I2C para el control del LCD.

4.7.4.3 menuItems

```
const char* MenuState::menuItems[]  [static], [private]
```

Elementos del menú.

4.7.4.4 menuLength

```
const int MenuState::menuLength  [static], [private]
```

Longitud del menú.

La documentación de esta clase está generada del siguiente archivo:

- sketch/menu_state.h

4.8 Referencia de la clase SamplingState

Clase que maneja el estado de muestreo en la máquina de estados.

```
#include <sampling_state.h>
```

Diagrama de herencia de SamplingState

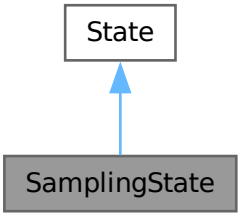
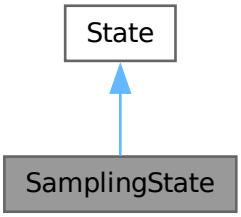


Diagrama de colaboración de SamplingState:



Métodos públicos

- SamplingState (LiquidCrystal_I2C &lcd)

  Constructor que inicializa el LCD.
- void handleUp (Context ∗context) override

  Maneja la acción del botón "Up".
- void handleDown (Context ∗context) override

  Maneja la acción del botón "Down".
- void handleSelect (Context ∗context) override

  Maneja la acción del botón "Select".
- void handleBack (Context ∗context) override

  Maneja la acción del botón "Back".
- void displayMenu (Context ∗context) override

  Muestra el menú en la pantalla LCD.

Métodos públicos heredados de State

- virtual ∼State ()

Métodos privados

- void initializeLcd ()

  Inicializa la pantalla LCD.
- void printLogo ()

  Imprime el logo en la pantalla LCD.
- void buzzerFunc (int value, int Rlow, int Rup)

  Controla el buzzer basado en el valor de muestreo.
- void countNumbersInRange (Context ∗contex)

  Cuenta los números en el rango definido.
- void updateLCD (int count)

  Actualiza la pantalla LCD con el conteo de números.

Atributos privados

- int currentIndex

  Índice actual para el muestreo.
- LiquidCrystal_I2C & lcd

  Referencia al objeto LCD.

4.8.1 Descripción detallada

Clase que maneja el estado de muestreo en la máquina de estados.

Esta clase gestiona las acciones y la visualización del estado de muestreo. Hereda de la clase base `State` y proporciona implementaciones específicas para el manejo de botones y la visualización en el LCD.

4.8.2 Documentación de constructores y destructores

4.8.2.1 SamplingState()

```
SamplingState::SamplingState (
            LiquidCrystal_I2C & lcd)
```

Constructor que inicializa el LCD.

4.8.3 Documentación de funciones miembro

4.8.3.1 buzzerFunc()

```
void SamplingState::buzzerFunc (
            int value,
            int Rlow,
            int Rup)  [private]
```

Controla el buzzer basado en el valor de muestreo.

4.8.3.2 countNumbersInRange()

```
void SamplingState::countNumbersInRange (
            Context * contex)  [private]
```

Cuenta los números en el rango definido.

4.8.3.3 displayMenu()

```
void SamplingState::displayMenu (
            Context * context)  [override], [virtual]
```

Muestra el menú en la pantalla LCD.

Implementa State.

4.8.3.4 handleBack()

```
void SamplingState::handleBack (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Back".

Implementa State.

4.8.3.5 handleDown()

```
void SamplingState::handleDown (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Down".

Implementa State.

4.8.3.6 handleSelect()

```
void SamplingState::handleSelect (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Select".

Implementa State.

4.8.3.7 handleUp()

```
void SamplingState::handleUp (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Up".

Implementa State.

4.8.3.8 initializeLcd()

```
void SamplingState::initializeLcd ()  [private]
```

Inicializa la pantalla LCD.

4.8.3.9 printLogo()

```
void SamplingState::printLogo ()  [private]
```

Imprime el logo en la pantalla LCD.

4.8.3.10 updateLCD()

```
void SamplingState::updateLCD (
            int count)  [private]
```

Actualiza la pantalla LCD con el conteo de números.

4.8.4 Documentación de datos miembro

4.8.4.1 currentIndex

```
int SamplingState::currentIndex  [private]
```

Índice actual para el muestreo.

4.8.4.2 lcd

```
LiquidCrystal_I2C& SamplingState::lcd  [private]
```

Referencia al objeto LCD.

La documentación de esta clase está generada del siguiente archivo:

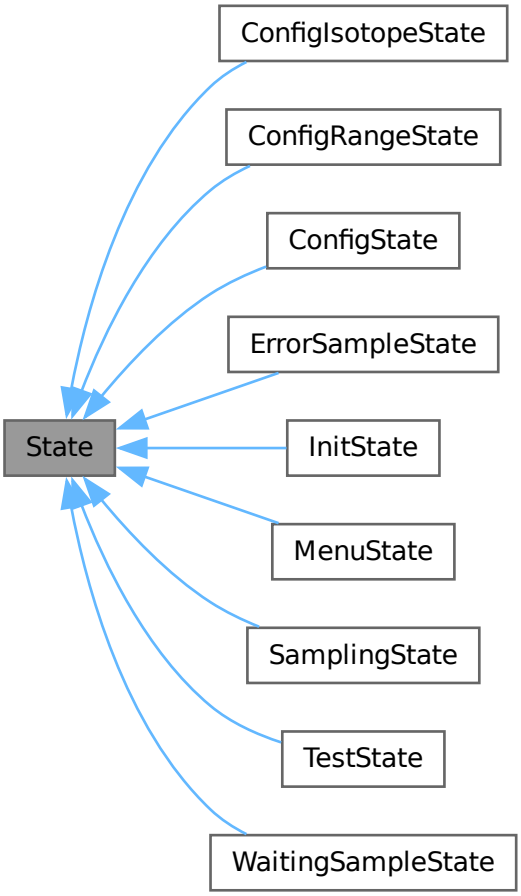
- sketch/sampling_state.h

4.9 Referencia de la clase State

Clase base abstracta para los estados de la máquina de estados.

```
#include <state.h>
```

Diagrama de herencia de State



Métodos públicos

- virtual ~State ()
- virtual void handleUp (Context *context)=0

  Maneja la acción del botón "Up".
- virtual void handleDown (Context *context)=0

  Maneja la acción del botón "Down".
- virtual void handleSelect (Context *context)=0

  Maneja la acción del botón "Select".
- virtual void handleBack (Context *context)=0

  Maneja la acción del botón "Back".
- virtual void displayMenu (Context *context)=0

  Muestra el menú o la información del estado en el LCD.

4.9.1 Descripción detallada

Clase base abstracta para los estados de la máquina de estados.

Esta clase define una interfaz común que todos los estados deben implementar. Los métodos virtuales puros aseguran que cada estado concrete sus propias versiones de estos métodos.

4.9.2 Documentación de constructores y destructores

4.9.2.1 ~State()

```
virtual State::~State ()  [inline], [virtual]
```

4.9.3 Documentación de funciones miembro

4.9.3.1 displayMenu()

```
virtual void State::displayMenu (
            Context * context)  [pure virtual]
```

Muestra el menú o la información del estado en el LCD.

Parámetros

context	Puntero al contexto actual de la máquina de estados.

Implementado en ConfigIsotopeState, ConfigRangeState, ConfigState, ErrorSampleState, InitState, MenuState, SamplingState, TestState y WaitingSampleState.

4.9.3.2 handleBack()

```
virtual void State::handleBack (
            Context * context)  [pure virtual]
```

Maneja la acción del botón "Back".

Parámetros

context	Puntero al contexto actual de la máquina de estados.

Implementado en ConfigIsotopeState, ConfigRangeState, ConfigState, ErrorSampleState, InitState, MenuState, SamplingState, TestState y WaitingSampleState.

4.9.3.3 handleDown()

```
virtual void State::handleDown (
            Context * context)  [pure virtual]
```

Maneja la acción del botón "Down".

Parámetros

context	Puntero al contexto actual de la máquina de estados.

Implementado en ConfigIsotopeState, ConfigRangeState, ConfigState, ErrorSampleState, InitState, MenuState, SamplingState, TestState y WaitingSampleState.

4.9.3.4 handleSelect()

```
virtual void State::handleSelect (
            Context * context)  [pure virtual]
```

Maneja la acción del botón "Select".

Parámetros

context	Puntero al contexto actual de la máquina de estados.

Implementado en ConfigIsotopeState, ConfigRangeState, ConfigState, ErrorSampleState, InitState, MenuState, SamplingState, TestState y WaitingSampleState.

4.9.3.5 handleUp()

```
virtual void State::handleUp (
            Context * context)  [pure virtual]
```

Maneja la acción del botón "Up".

Parámetros

context	Puntero al contexto actual de la máquina de estados.

Implementado en ConfigIsotopeState, ConfigRangeState, ConfigState, ErrorSampleState, InitState, MenuState, SamplingState, TestState y WaitingSampleState.

La documentación de esta clase está generada del siguiente archivo:

- sketch/state.h

4.10 Referencia de la clase TestState

Clase que maneja el estado de prueba en la interfaz.

`#include <test_state.h>`

Diagrama de herencia de TestState

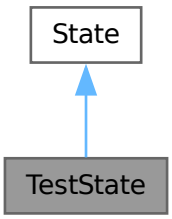
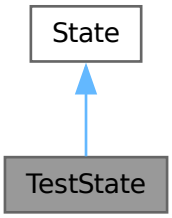


Diagrama de colaboración de TestState:



Métodos públicos

- TestState (LiquidCrystal_I2C &lcd)
  Constructor que inicializa el LCD.
- void handleUp (Context *context) override
  Maneja la acción del botón "Up".
- void handleDown (Context *context) override
  Maneja la acción del botón "Down".
- void handleSelect (Context *context) override
  Maneja la acción del botón "Select".
- void handleBack (Context *context) override
  Maneja la acción del botón "Back".
- void displayMenu (Context *context) override
  Muestra el menú en la pantalla LCD.

Métodos públicos heredados de State

- virtual ~State ()

Métodos privados

- void initializeLcd ()

  Inicializa la pantalla LCD.
- void printLogo ()

  Imprime el logo en la pantalla LCD.

Atributos privados

- LiquidCrystal_I2C & lcd

  Referencia al objeto LCD.

4.10.1 Descripción detallada

Clase que maneja el estado de prueba en la interfaz.

Esta clase gestiona las acciones y la visualización para una ventana de prueba. Hereda de la clase base `State` y proporciona implementaciones específicas para el manejo de botones y la visualización en el LCD.

4.10.2 Documentación de constructores y destructores

4.10.2.1 TestState()

```
TestState::TestState (
            LiquidCrystal_I2C & lcd)
```

Constructor que inicializa el LCD.

Parámetros

lcd	Referencia al objeto LiquidCrystal_I2C.

4.10.3 Documentación de funciones miembro

4.10.3.1 displayMenu()

```
void TestState::displayMenu (
            Context * context)  [override], [virtual]
```

Muestra el menú en la pantalla LCD.

Implementa State.

4.10.3.2 handleBack()

```
void TestState::handleBack (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Back".

Implementa State.

4.10.3.3 handleDown()

```
void TestState::handleDown (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Down".

Implementa State.

4.10.3.4 handleSelect()

```
void TestState::handleSelect (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Select".

Implementa State.

4.10.3.5 handleUp()

```
void TestState::handleUp (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Up".

Implementa State.

4.10.3.6 initializeLcd()

```
void TestState::initializeLcd ()  [private]
```

Inicializa la pantalla LCD.

4.10.3.7 printLogo()

```
void TestState::printLogo ()  [private]
```

Imprime el logo en la pantalla LCD.

4.10.4 Documentación de datos miembro

4.10.4.1 lcd

```
LiquidCrystal_I2C& TestState::lcd  [private]
```

Referencia al objeto LCD.

La documentación de esta clase está generada del siguiente archivo:

- sketch/test_state.h

4.11 Referencia de la clase WaitingSampleState

Clase que maneja el estado de espera antes de iniciar el muestreo.

```
#include <waitingSample_state.h>
```

Diagrama de herencia de WaitingSampleState

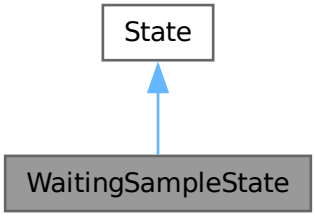
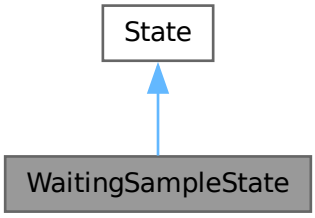


Diagrama de colaboración de WaitingSampleState:



Métodos públicos

- WaitingSampleState (LiquidCrystal_I2C &lcd)
  Constructor que inicializa el LCD.
- void handleUp (Context *context) override
  Maneja la acción del botón "Up".
- void handleDown (Context *context) override
  Maneja la acción del botón "Down".
- void handleSelect (Context *context) override
  Maneja la acción del botón "Select".
- void handleBack (Context *context) override
  Maneja la acción del botón "Back".
- void displayMenu (Context *context) override
  Muestra el menú en la pantalla LCD.

Métodos públicos heredados de State

- virtual ~State ()

Métodos privados

- void initializeLcd ()
  Inicializa la pantalla LCD.
- void printLogo ()
  Imprime el logo en la pantalla LCD.

Atributos privados

- int currentIndex
  Índice actual (no utilizado en esta implementación).
- LiquidCrystal_I2C & lcd
  Referencia al objeto LCD.

4.11.1 Descripción detallada

Clase que maneja el estado de espera antes de iniciar el muestreo.

Esta clase gestiona las acciones y la visualización del estado de espera. Hereda de la clase base `State` y proporciona implementaciones específicas para el manejo de botones y la visualización en el LCD.

4.11.2 Documentación de constructores y destructores

4.11.2.1 WaitingSampleState()

```
WaitingSampleState::WaitingSampleState (
            LiquidCrystal_I2C & lcd)
```

Constructor que inicializa el LCD.

4.11.3 Documentación de funciones miembro

4.11.3.1 displayMenu()

```
void WaitingSampleState::displayMenu (
            Context * context)  [override], [virtual]
```

Muestra el menú en la pantalla LCD.

Implementa State.

4.11.3.2 handleBack()

```
void WaitingSampleState::handleBack (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Back".

Implementa State.

4.11.3.3 handleDown()

```
void WaitingSampleState::handleDown (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Down".

Implementa State.

4.11.3.4 handleSelect()

```
void WaitingSampleState::handleSelect (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Select".

Implementa State.

4.11.3.5 handleUp()

```
void WaitingSampleState::handleUp (
            Context * context)  [override], [virtual]
```

Maneja la acción del botón "Up".

Implementa State.

4.11.3.6 initializeLcd()

```
void WaitingSampleState::initializeLcd ()  [private]
```

Inicializa la pantalla LCD.

4.11.3.7 printLogo()

```
void WaitingSampleState::printLogo ()  [private]
```

Imprime el logo en la pantalla LCD.

4.11.4 Documentación de datos miembro

4.11.4.1 currentIndex

```
int WaitingSampleState::currentIndex  [private]
```

Índice actual (no utilizado en esta implementación).

4.11.4.2 lcd

```
LiquidCrystal_I2C& WaitingSampleState::lcd  [private]
```

Referencia al objeto LCD.

La documentación de esta clase está generada del siguiente archivo:

- sketch/waitingSample_state.h

Chapter 5

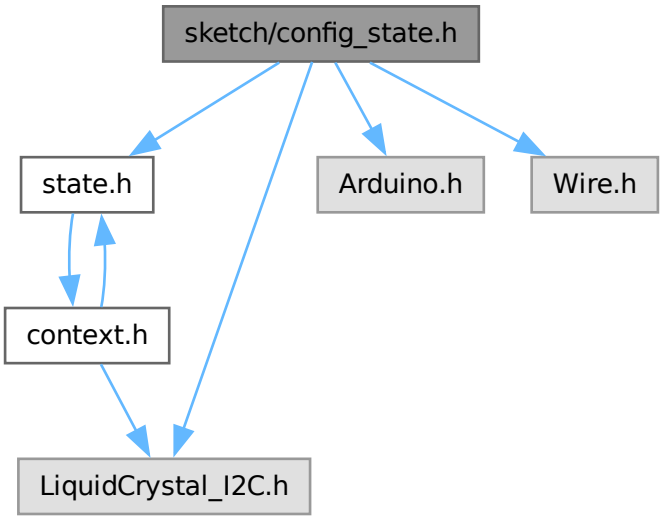
Documentación de archivos

5.1 Referencia del archivo sketch/config_state.h

Declaración de la clase ConfigState y sus miembros.

```
#include "state.h"
#include <Arduino.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Gráfico de dependencias incluidas en config_state.h:



Clases

- class ConfigState

  Clase que representa el estado de configuración del sistema.

5.1.1 Descripción detallada

Declaración de la clase ConfigState y sus miembros.

Autor

Lara Torletti (lara.a.torletti@gmail.com)

Esta clase maneja el estado de configuración en una máquina de estados para un sistema basado en Arduino. Proporciona la interfaz y los métodos necesarios para gestionar el menú de configuración del sistema.

Versión

0.1

Fecha

2024-07-1

Copyright

Copyright (c) 2024

5.2 config_state.h

Ir a la documentación de este archivo.

```
00001
00016 #ifndef CONFIG_STATE_H
00017 #define CONFIG_STATE_H
00018
00019 #include "state.h"
00020 #include <Arduino.h>
00021 #include <Wire.h>
00022 #include <LiquidCrystal_I2C.h>
00023
00028 class ConfigState : public State
00029 {
00030 public:
00036     ConfigState(LiquidCrystal_I2C &lcd);
00037
00043     void handleUp(Context *context) override;
00044
00050     void handleDown(Context *context) override;
00051
00057     void handleSelect(Context *context) override;
00058
00064     void handleBack(Context *context) override;
00065
00071     void displayMenu(Context *context) override;
00072
00073 private:
00074     int currentIndex;
00075     LiquidCrystal_I2C &lcd;
00076
00080     void initializeLcd();
00081
00085     void printLogo();
00086
00087     static const char *menuItems[];
00088     static const int menuLength;
00089 };
00090
00091 #endif // CONFIG_STATE_H
```

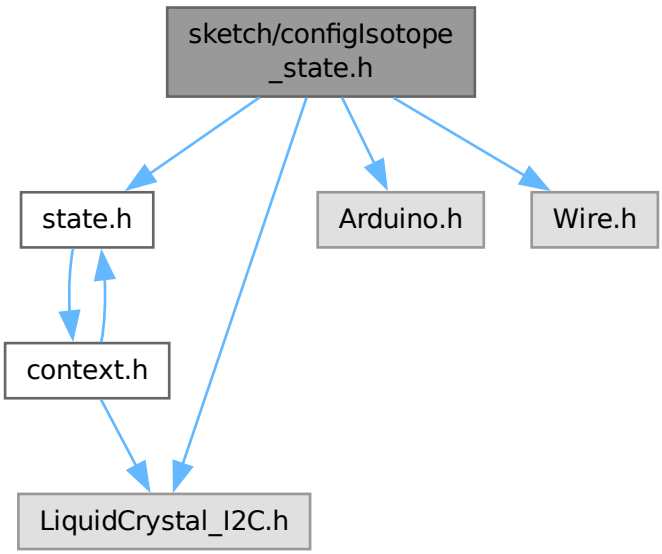
5.3 Referencia del archivo sketch/config_state.ino

5.4 Referencia del archivo sketch/configIsotope_state.h

Definición de la clase ConfigIsotopeState para la configuración del isótopo en la interfaz.

```
#include "state.h"
#include <Arduino.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Gráfico de dependencias incluidas en configIsotope_state.h:



Clases

- class ConfigIsotopeState

  Clase que maneja el estado de configuración del isótopo en la interfaz.

5.4.1 Descripción detallada

Definición de la clase ConfigIsotopeState para la configuración del isótopo en la interfaz.

Autor

Lara Torletti (lara.a.torletti@gmail.com)

Versión

0.1

Fecha

2024-07-01

Copyright

5.5 configIsotope_state.h

Ir a la documentación de este archivo.

```
00001
00012 #ifndef CONFIGISOTOPE_STATE_H
00013 #define CONFIGISOTOPE_STATE_H
00014
00015 #include "state.h"
00016 #include <Arduino.h>
00017 #include <Wire.h>
00018 #include <LiquidCrystal_I2C.h>
00019
00028 class ConfigIsotopeState : public State
00029 {
00030 private:
00031     LiquidCrystal_I2C &lcd;
00032     int currentIndex;
00033     static const char *menuItems[];
00034     static const int menuLength;
00035
00036     void initializeLcd();
00037     void printLogo();
00038
00039 public:
00044     ConfigIsotopeState(LiquidCrystal_I2C &lcd);
00045
00046     void handleUp(Context *context) override;
00047     void handleDown(Context *context) override;
00048     void handleSelect(Context *context) override;
00049     void handleBack(Context *context) override;
00050     void displayMenu(Context *context) override;
00051 };
00052
00053 #endif // CONFIGISOTOPE_STATE_H
```

5.6 Referencia del archivo sketch/configIsotope_state.ino

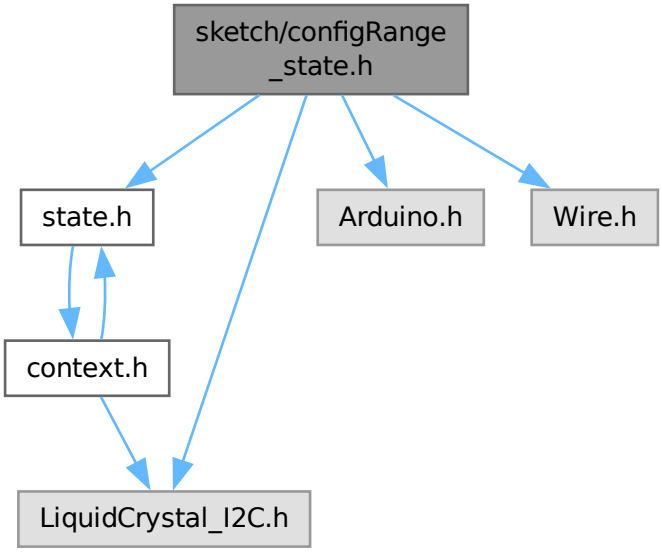
5.7 Referencia del archivo sketch/configRange_state.h

Definición de la clase ConfigRangeState para la configuración de límites en la interfaz.

```
#include "state.h"
#include <Arduino.h>
#include <Wire.h>
```

```
#include <LiquidCrystal_I2C.h>
```

Gráfico de dependencias incluidas en configRange_state.h:



Clases

- class ConfigRangeState

  Clase que maneja el estado de configuración de los límites en la interfaz.

5.7.1 Descripción detallada

Definición de la clase ConfigRangeState para la configuración de límites en la interfaz.

Autor

Lara Torletti (lara.a.torletti@gmail.com)

Versión

0.1

Fecha

2024-07-29

Copyright

Copyright (c) 2024

5.8 configRange_state.h

Ir a la documentación de este archivo.

```
00001
00012 #ifndef CONFIGRANGE_STATE_H
00013 #define CONFIGRANGE_STATE_H
00014
00015 #include "state.h"
00016 #include <Arduino.h>
00017 #include <Wire.h>
00018 #include <LiquidCrystal_I2C.h>
00019
00028 class ConfigRangeState : public State
00029 {
00030 private:
00031     LiquidCrystal_I2C &lcd;
00032     int lowerLimit;
00033     int upperLimit;
00034     bool settingLowerLimit;
00035     void initializeLcd();
00036     void printLogo();
00037
00038 public:
00043     ConfigRangeState(LiquidCrystal_I2C &lcd);
00044
00045     void handleUp(Context *context) override;
00046     void handleDown(Context *context) override;
00047     void handleSelect(Context *context) override;
00048     void handleBack(Context *context) override;
00049     void displayMenu(Context *context) override;
00050 };
00051
00052 #endif // CONFIGRANGE_STATE_H
```

5.9 Referencia del archivo sketch/configRange_state.ino

5.10 Referencia del archivo sketch/context.h

Definición de la clase Context para la gestión del estado de la aplicación.

```
#include "state.h"
#include <LiquidCrystal_I2C.h>
```

Gráfico de dependencias incluidas en context.h:

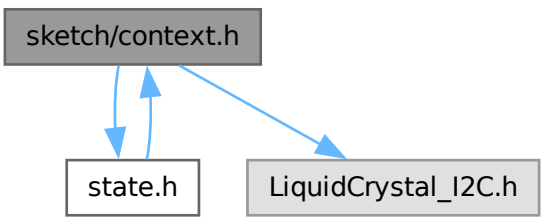
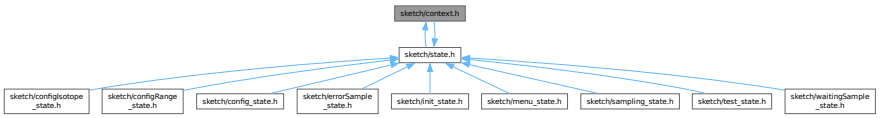


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Clases

- class Context

5.10.1 Descripción detallada

Definición de la clase Context para la gestión del estado de la aplicación.

Autor

Lara Torletti (lara.a.torletti@gmail.com)

Versión

0.1

Fecha

2024-07-01

La clase `Context` sigue el patrón de diseño de estado y actúa como un gestor central que mantiene el estado actual de la aplicación y facilita la transición entre estados. Proporciona métodos para manejar entradas y actualizar la interfaz de usuario del LCD.

Copyright

Copyright (c) 2024

5.11 context.h

Ir a la documentación de este archivo.

```
00001
00015 #ifndef CONTEXT_H
00016 #define CONTEXT_H
00017
00018 #include "state.h"
00019 #include <LiquidCrystal_I2C.h>
00020
00021 class State;
00022
00023 class Context
00024 {
00025 public:
00030     static Context *getInstance();
00031
00036     void setState(State *state);
```

```
00037
00041     void handleUp();
00042
00046     void handleDown();
00047
00051     void handleSelect();
00052
00056     void handleBack();
00057
00062     void displayMenu();
00063
00068     void setLowerLimit(int lowerLimit);
00069
00074     int getLowerLimit() const;
00075
00080     void setUpperLimit(int upperLimit);
00081
00086     int getUpperLimit() const;
00087
00092     void setIsotope(const char *isotope);
00093
00098     const char *getIsotope() const;
00099
00104     LiquidCrystal_I2C &getLcd();
00105
00106 private:
00107     Context();
00108     State *currentState;
00109     LiquidCrystal_I2C lcd;
00110     static Context *instance;
00111
00112     int lowerLimit;
00113     int upperLimit;
00114     const char *isotope;
00115 };
00116
00117 #endif // CONTEXT_H
```

5.12 Referencia del archivo sketch/context.ino

5.13 Referencia del archivo sketch/custom_char.h

Definiciones de caracteres personalizados para el LCD.

defines

- #define printByte(args) print(args, BYTE);

  Macro para imprimir un byte en el LCD en versiones antiguas de Arduino.

Variables

- uint8_t bell [ ]

  Array para el icono de la campana.
- uint8_t note [ ]

  Array para el icono de la nota musical.
- uint8_t clock [ ]

  Array para el icono del reloj.
- uint8_t heart [ ]

  Array para el icono del corazón.
- uint8_t duck [ ]

  Array para el icono del pato.
- uint8_t check [ ]

Array para el icono de marca de verificación.

- uint8_t cross [ ]

  Array para el icono de cruz.
- uint8_t retarrow [ ]

  Array para el icono de flecha hacia atrás.
- uint8_t menu_icon [ ]

  Array para el icono del menú
- uint8_t logo_2x2 [4][8]

  Array para los iconos de logo en formato 2x2.
- uint8_t sample_icon [ ]

  Array para el icono de muestreo.
- uint8_t conf_icon [ ]

  Array para el icono de configuración.

5.13.1 Descripción detallada

Definiciones de caracteres personalizados para el LCD.

Autor

Lara Torletti (lara.a.torletti@gmail.com)

Versión

0.1

Fecha

2024-07-01

Copyright

Copyright (c) 2024

5.13.2 Documentación de «define»

5.13.2.1 printByte

```
#define printByte(
              args) print(args, BYTE);
```

Macro para imprimir un byte en el LCD en versiones antiguas de Arduino.

5.13.3 Documentación de variables

5.13.3.1 bell

```
uint8_t bell[]  [extern]
```

Array para el icono de la campana.

5.13.3.2 check

```
uint8_t check[]  [extern]
```

Array para el icono de marca de verificación.

5.13.3.3 clock

```
uint8_t clock[]  [extern]
```

Array para el icono del reloj.

5.13.3.4 conf_icon

```
uint8_t conf_icon[]  [extern]
```

Array para el icono de configuración.

5.13.3.5 cross

```
uint8_t cross[]  [extern]
```

Array para el icono de cruz.

5.13.3.6 duck

```
uint8_t duck[]  [extern]
```

Array para el icono del pato.

5.13.3.7 heart

```
uint8_t heart[]  [extern]
```

Array para el icono del corazón.

5.13.3.8 logo_2x2

```
uint8_t logo_2x2[4][8]  [extern]
```

Array para los iconos de logo en formato 2x2.

5.13.3.9 menu_icon

```
uint8_t menu_icon[]  [extern]
```

Array para el icono del menú

5.13.3.10 note

```
uint8_t note[]  [extern]
```

Array para el icono de la nota musical.

5.13.3.11 retarrow

```
uint8_t retarrow[]  [extern]
```

Array para el icono de flecha hacia atrás.

5.13.3.12 sample_icon

```
uint8_t sample_icon[]  [extern]
```

Array para el icono de muestreo.

5.14 custom_char.h

Ir a la documentación de este archivo.

```
00001
00012 #ifndef CUSTOM_CHAR_H
00013 #define CUSTOM_CHAR_H
00014
00015 #if defined(ARDUINO) && ARDUINO >= 100
00016 #define printByte(args) write(args);
00017 #else
00018 #define printByte(args) print(args, BYTE);
00019 #endif
00020
00021 extern uint8_t bell[];
00022 extern uint8_t note[];
00023 extern uint8_t clock[];
00024 extern uint8_t heart[];
00025 extern uint8_t duck[];
00026 extern uint8_t check[];
00027 extern uint8_t cross[];
00028 extern uint8_t retarrow[];
00029 extern uint8_t menu_icon[];
00030 extern uint8_t logo_2x2[4][8];
00031 extern uint8_t sample_icon[];
00032 extern uint8_t conf_icon[];
00033
00034 #endif // CUSTOM_CHAR_H
```

5.15 Referencia del archivo sketch/custom_char.ino

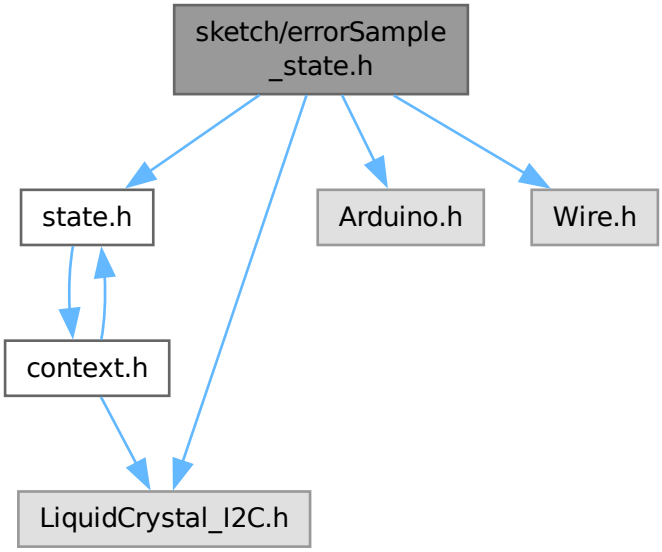
5.16 Referencia del archivo sketch/errorSample_state.h

Definición de la clase `ErrorSampleState`, que representa el estado de error en el muestreo.

```
#include "state.h"
#include <Arduino.h>
#include <Wire.h>
```

```
#include <LiquidCrystal_I2C.h>
```

Gráfico de dependencias incluidas en errorSample_state.h:



Clases

- class ErrorSampleState

  Clase que representa el estado de error en el muestreo.

5.16.1 Descripción detallada

Definición de la clase `ErrorSampleState`, que representa el estado de error en el muestreo.

Autor

Lara Torletti (lara.a.torletti@gmail.com)

Versión

0.1

Fecha

2024-07-01

La clase `ErrorSampleState` maneja el estado en el que se detecta un error durante el muestreo. Hereda de `State` y proporciona métodos para manejar las entradas del usuario y actualizar la pantalla LCD en caso de error.

Copyright

Copyright (c) 2024

5.17 errorSample_state.h

Ir a la documentación de este archivo.

```
00001
00014 #ifndef ERRORSAMPLE_STATE_H
00015 #define ERRORSAMPLE_STATE_H
00016
00017 #include "state.h"
00018 #include <Arduino.h>
00019 #include <Wire.h>
00020 #include <LiquidCrystal_I2C.h>
00021
00029 class ErrorSampleState : public State
00030 {
00031 private:
00032     LiquidCrystal_I2C &lcd;
00033
00037     void initializeLcd();
00038
00042     void printLogo();
00043
00044 public:
00049     ErrorSampleState(LiquidCrystal_I2C &lcd);
00050
00055     void handleUp(Context *context) override;
00056
00061     void handleDown(Context *context) override;
00062
00067     void handleSelect(Context *context) override;
00068
00073     void handleBack(Context *context) override;
00074
00079     void displayMenu(Context *context) override;
00080 };
00081
00082 #endif // ERRORSAMPLE_STATE_H
```

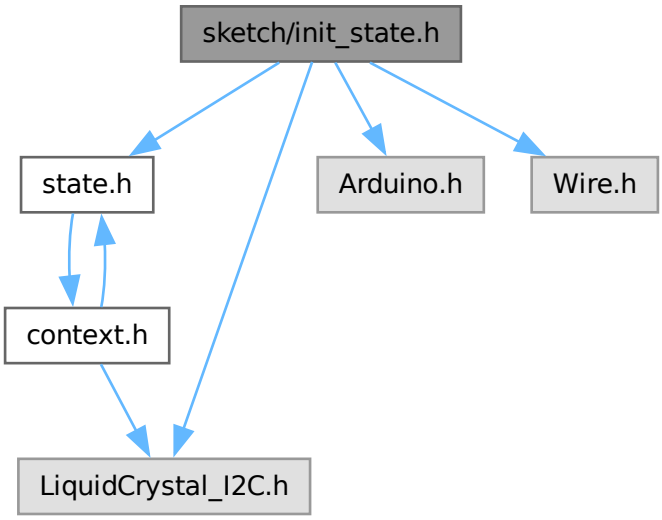
5.18 Referencia del archivo sketch/errorSample_state.ino

5.19 Referencia del archivo sketch/init_state.h

Declaración de la clase InitState y sus miembros.

```
#include "state.h"
#include <Arduino.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Gráfico de dependencias incluidas en init_state.h:



Clases

- class InitState

  Clase que representa el estado de inicialización del sistema.

5.19.1 Descripción detallada

Declaración de la clase InitState y sus miembros.

Autor

Lara Torletti (lara.a.torletti@gmail.com)

Versión

0.1

Fecha

2024-07-1

Esta clase maneja el estado de inicialización en una máquina de estados para un sistema basado en Arduino. Proporciona la interfaz y los métodos necesarios para gestionar la pantalla de bienvenida del sistema.

Copyright

5.20 init_state.h

Ir a la documentación de este archivo.

```
00001
00014 #ifndef INIT_STATE_H
00015 #define INIT_STATE_H
00016
00017 #include "state.h"
00018 #include <Arduino.h>
00019 #include <Wire.h>
00020 #include <LiquidCrystal_I2C.h>
00021
00026 class InitState : public State
00027 {
00028 public:
00034     InitState(LiquidCrystal_I2C &lcd);
00035
00036     void handleUp(Context *context) override;
00037     void handleDown(Context *context) override;
00038
00044     void handleSelect(Context *context) override;
00045     void handleBack(Context *context) override;
00046
00052     void displayMenu(Context *context) override;
00053
00054 private:
00055     LiquidCrystal_I2C &lcd;
00056
00060     void initializeLcd();
00061
00065     void printLogo();
00066 };
00067
00068 #endif // INIT_STATE_H
```

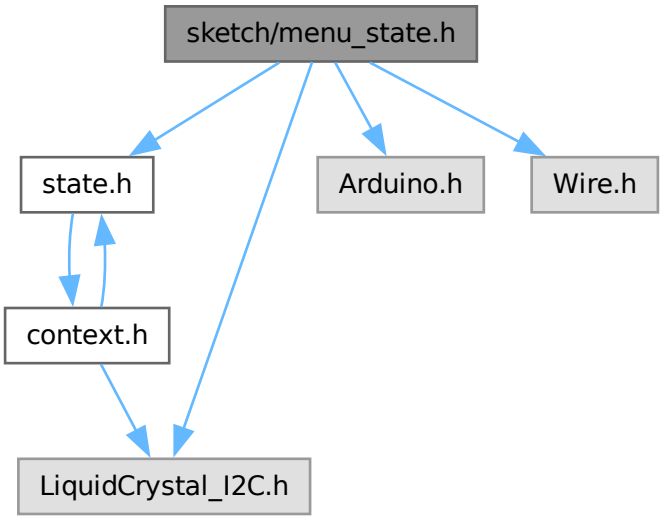
5.21 Referencia del archivo sketch/init_state.ino

5.22 Referencia del archivo sketch/menu_state.h

Declaración de la clase MenuState y sus miembros.

```
#include "state.h"
#include <Arduino.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Gráfico de dependencias incluidas en menu_state.h:



Clases

- class MenuState

  Clase que representa el estado del menú del sistema.

5.22.1 Descripción detallada

Declaración de la clase MenuState y sus miembros.

Autor

Lara Torletti (lara.a.torletti@gmail.com)

Versión

0.1

Fecha

2024-07-01

Esta clase maneja el estado del menú en una máquina de estados para un sistema basado en Arduino. Proporciona la interfaz y los métodos necesarios para gestionar el menú principal del sistema.

Copyright

Copyright (c) 2024

5.23 menu_state.h

Ir a la documentación de este archivo.

```
00001
00014 #ifndef MENU_STATE_H
00015 #define MENU_STATE_H
00016
00017 #include "state.h"
00018 #include <Arduino.h>
00019 #include <Wire.h>
00020 #include <LiquidCrystal_I2C.h>
00021
00026 class MenuState : public State
00027 {
00028 public:
00034     MenuState(LiquidCrystal_I2C &lcd);
00035
00041     void handleUp(Context *context) override;
00042
00048     void handleDown(Context *context) override;
00049
00055     void handleSelect(Context *context) override;
00056
00062     void handleBack(Context *context) override;
00063
00069     void displayMenu(Context *context) override;
00070
00071 private:
00072     int currentIndex;
00073     LiquidCrystal_I2C &lcd;
00074
00078     void initializeLcd();
00079
00083     void printLogo();
00084     static const char *menuItems[];
00085     static const int menuLength;
00086 };
00087
00088 #endif // MENU_STATE_H
```

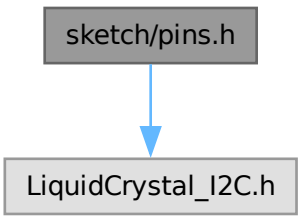
5.24 Referencia del archivo sketch/menu_state.ino

5.25 Referencia del archivo sketch/pins.h

Declaración de pines y funciones de inicialización para el sistema.

`#include <LiquidCrystal_I2C.h>`

Gráfico de dependencias incluidas en pins.h:



Funciones

- void initPins ()

  Inicializa los pines configurándolos como entradas o salidas según corresponda.

Variables

- const int button_up

  Pin del botón para subir.
- const int button_down

  Pin del botón para bajar.
- const int button_select

  Pin del botón para seleccionar.
- const int button_back

  Pin del botón para regresar.
- const int led_red

  Pin del LED rojo.
- const int led_yellow

  Pin del LED amarillo.
- const int buzzer

  Pin del buzzer.

5.25.1 Descripción detallada

Declaración de pines y funciones de inicialización para el sistema.

Autor

Lara Torletti (lara.a.torletti@gmail.com)

Versión

0.1

Fecha

2024-07-01

Este archivo contiene las declaraciones de los pines utilizados en el sistema, así como la función para inicializarlos. Está diseñado para ser utilizado en un proyecto basado en Arduino.

Copyright

Copyright (c) 2024

5.25.2 Documentación de funciones

5.25.2.1 initPins()

```
void initPins ()
```

Inicializa los pines configurándolos como entradas o salidas según corresponda.

5.25.3 Documentación de variables

5.25.3.1 button_back

```
const int button_back  [extern]
```

Pin del botón para regresar.

5.25.3.2 button_down

```
const int button_down  [extern]
```

Pin del botón para bajar.

5.25.3.3 button_select

```
const int button_select  [extern]
```

Pin del botón para seleccionar.

5.25.3.4 button_up

```
const int button_up  [extern]
```

Pin del botón para subir.

5.25.3.5 buzzer

```
const int buzzer  [extern]
```

Pin del buzzer.

5.25.3.6 led_red

```
const int led_red  [extern]
```

Pin del LED rojo.

5.25.3.7 led_yellow

```
const int led_yellow  [extern]
```

Pin del LED amarillo.

5.26 pins.h

Ir a la documentación de este archivo.

```
00001
00014 #ifndef PINS_H
00015 #define PINS_H
00016
00017 #include <LiquidCrystal_I2C.h>
00018
00019 extern const int button_up;
00020 extern const int button_down;
00021 extern const int button_select;
00022 extern const int button_back;
00023
00024 extern const int led_red;
00025 extern const int led_yellow;
00026
00027 extern const int buzzer;
00028
00032 void initPins();
00033
00034 #endif // PINS_H
```

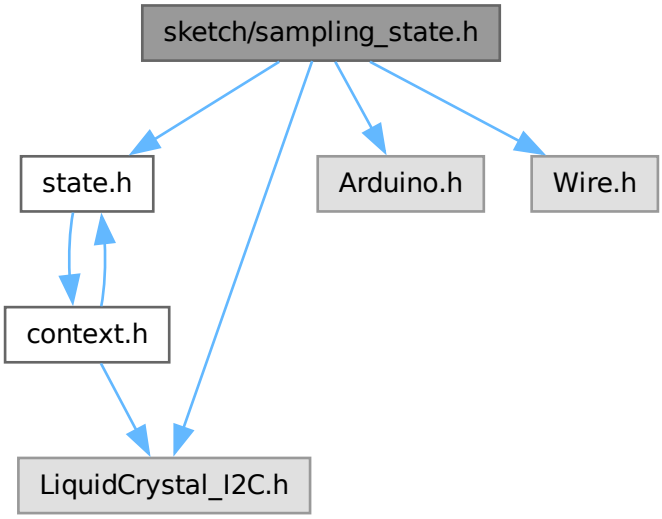
5.27 Referencia del archivo sketch/pins.ino

5.28 Referencia del archivo sketch/sampling_state.h

Definición de la clase SamplingState para gestionar el estado de muestreo.

```
#include "state.h"
#include <Arduino.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Gráfico de dependencias incluidas en sampling_state.h:



Clases

- class SamplingState

  Clase que maneja el estado de muestreo en la máquina de estados.

5.28.1 Descripción detallada

Definición de la clase SamplingState para gestionar el estado de muestreo.

Autor

Lara Torletti (lara.a.torletti@gmail.com)

Versión

0.1

0.1

Fecha

2024-07-01

Copyright

Copyright (c) 2024

5.29 sampling_state.h

Ir a la documentación de este archivo.

```
00001
00013 #ifndef SAMPLING_STATE_H
00014 #define SAMPLING_STATE_H
00015
00016 #include "state.h"
00017 #include <Arduino.h>
00018 #include <Wire.h>
00019 #include <LiquidCrystal_I2C.h>
00020
00029 class SamplingState : public State
00030 {
00031 private:
00032     int currentIndex;
00033     LiquidCrystal_I2C &lcd;
00034     void initializeLcd();
00035     void printLogo();
00036     void buzzerFunc(int value, int Rlow, int Rup);
00037     void countNumbersInRange(Context *contex);
00038     void SamplingState::updateLCD(int count);
00039
00040 public:
00041     SamplingState(LiquidCrystal_I2C &lcd);
00042
00043     void handleUp(Context *context) override;
00044     void handleDown(Context *context) override;
00045     void handleSelect(Context *context) override;
00046     void handleBack(Context *context) override;
00047     void displayMenu(Context *context) override;
00048 };
00049
00050 #endif // SAMPLING_STATE_H
```

5.30 Referencia del archivo sketch/sampling_state.ino

5.31 Referencia del archivo sketch/sketch.ino

5.32 Referencia del archivo sketch/state.h

```
#include "context.h"
```

Gráfico de dependencias incluidas en state.h:

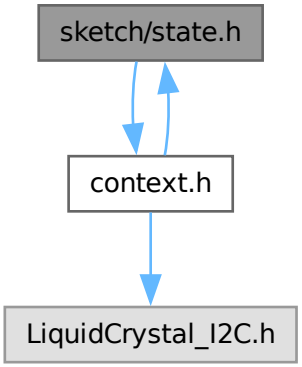
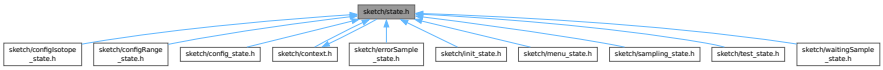


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Clases

- class State

  Clase base abstracta para los estados de la máquina de estados.

5.33 state.h

Ir a la documentación de este archivo.

```
00001
00012 #ifndef STATE_H
00013 #define STATE_H
00014
00015 #include "context.h"
00016
00017 class Context;
00018
```

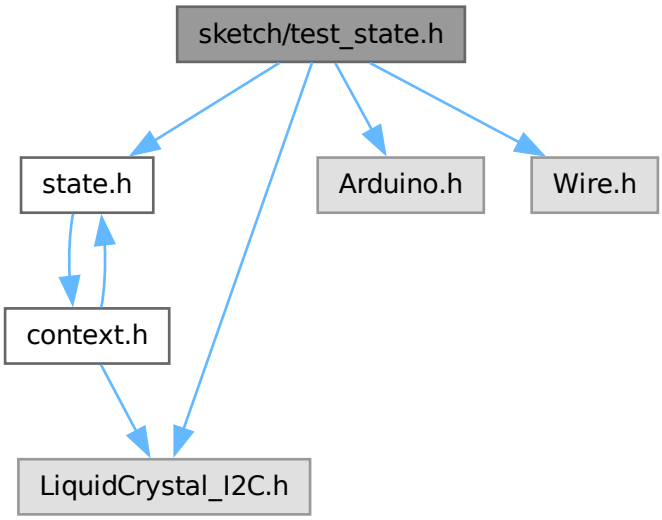
```
00027 class State
00028 {
00029 public:
00030     // Destructor virtual para permitir la eliminación correcta de objetos derivados
00031     virtual ~State() {}
00032
00037     virtual void handleUp(Context *context) = 0;
00038
00043     virtual void handleDown(Context *context) = 0;
00044
00049     virtual void handleSelect(Context *context) = 0;
00050
00055     virtual void handleBack(Context *context) = 0;
00056
00061     virtual void displayMenu(Context *context) = 0;
00062 };
00063
00064 #endif // STATE_H
```

5.34 Referencia del archivo sketch/test_state.h

Definición de la clase TestState para gestionar una ventana de prueba en la interfaz.

```
#include "state.h"
#include <Arduino.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Gráfico de dependencias incluidas en test_state.h:



Clases

- class TestState

  Clase que maneja el estado de prueba en la interfaz.

5.34.1 Descripción detallada

Definición de la clase TestState para gestionar una ventana de prueba en la interfaz.

Autor

Lara Torletti (lara.a.torletti@gmail.com)

Versión

0.1

Fecha

2024-07-29

Copyright

5.35 test_state.h

Ir a la documentación de este archivo.

```
00001
00012 #ifndef TEST_STATE_H
00013 #define TEST_STATE_H
00014
00015 #include "state.h"
00016 #include <Arduino.h>
00017 #include <Wire.h>
00018 #include <LiquidCrystal_I2C.h>
00019
00028 class TestState : public State
00029 {
00030 public:
00035     TestState(LiquidCrystal_I2C &lcd);
00036
00037     void handleUp(Context *context) override;
00038     void handleDown(Context *context) override;
00039     void handleSelect(Context *context) override;
00040     void handleBack(Context *context) override;
00041     void displayMenu(Context *context) override;
00042
00043 private:
00044     LiquidCrystal_I2C &lcd;
00045     void initializeLcd();
00046     void printLogo();
00047 };
00048
00049 #endif // TEST_STATE_H
```

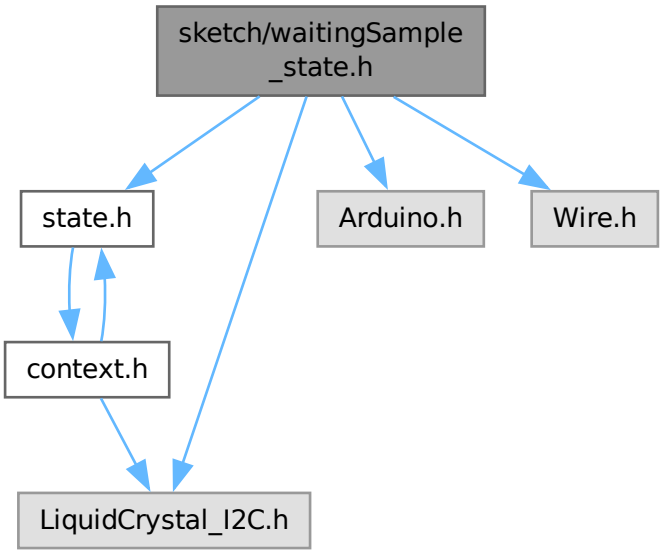
5.36 Referencia del archivo sketch/test_state.ino

5.37 Referencia del archivo sketch/waitingSample_state.h

```
#include "state.h"
#include <Arduino.h>
#include <Wire.h>
```

```
#include <LiquidCrystal_I2C.h>
```

Gráfico de dependencias incluidas en waitingSample_state.h:



Clases

- class WaitingSampleState

  Clase que maneja el estado de espera antes de iniciar el muestreo.

5.38 waitingSample_state.h

Ir a la documentación de este archivo.

```
00001
00012 #ifndef WAITINGSAMPLE_STATE_H
00013 #define WAITINGSAMPLE_STATE_H
00014
00015 #include "state.h"
00016 #include <Arduino.h>
00017 #include <Wire.h>
00018 #include <LiquidCrystal_I2C.h>
00019
00028 class WaitingSampleState : public State
00029 {
00030 private:
00031     int currentIndex;
00032     LiquidCrystal_I2C &lcd;
00033
00034     void initializeLcd();
00035     void printLogo();
00036
00037 public:
00038     WaitingSampleState(LiquidCrystal_I2C &lcd);
00039
00040     void handleUp(Context *context) override;
00041     void handleDown(Context *context) override;
00042     void handleSelect(Context *context) override;
00043     void handleBack(Context *context) override;
00044     void displayMenu(Context *context) override;
00045 };
00046
00047 #endif // WAITINGSAMPLE_STATE_H
```

5.39 Referencia del archivo sketch/waitingSample_state.ino