Librería de Arduino que permite ejecutar tareas de forma 'asincrona' https://www.luisllamas.es/libreria-ar...

6 commits 1 branch 0 releases 2 contributors Apache-2.0

Branch: master New pull request Find file Clone or download

luisllamasbinaburo Update library.properties Latest commit 9baeca5 on 18 Apr

examples	Add files via upload	9 months ago
src	Update AsyncTaskLib.cpp	8 months ago
LICENSE	Initial commit	9 months ago
README.md	Update README.md	9 months ago
keywords.txt	Add files via upload	9 months ago
library.properties	Update library.properties	4 months ago

README.md

Libería Arduino AsyncTask

La librería AsyncTask implementa una tarea que se ejecuta transcurrido un determinado periodo de microsegundos. Esto permite programar múltiples tareas de forma no bloqueante, consiguiendo un cierto comportamiento "asíncrono".

Más información https://www.luisllamas.es/libreria-arduino-asynctask/

Instrucciones de uso

En el constructor de la tarea podemos especificar el tiempo en milisegundos, y la función de Callback que se ejecutará una vez transcurrido el tiempo. También podemos especificar el tiempo con las funciones SetIntervalMillis() y SetIntervalMicros()

Para actualizar la AsyncTask es necesario llamar frecuentemente al método Update(), que comprueba el tiempo transcurrido desde el inicio de la tarea, y ejecuta las acciones pertinentes si el tiempo ha expirado.

Opcionalmente, la función Update admite como parámetro otra AsyncTask. Cuando la tarea expira, y tras ejecutar la función de Callback, iniciará la AsyncTask pasada como parámetro. Esto permite crear cadenas de tareas.

Además, las AsyncTask pueden ser AutoReset, bien a través del constructor, o activando la propiedad. Esto hace que la tarea vuelva a iniciarse cuando ha expirado.

Finalmente, también disponemos de métodos para iniciar, parar, reiniciar la tarea, determinar si la tarea está activa o detenida, si la tarea ha expirado, el tiempo transcurrido desde el inicio y el tiempo restante.

Constructor

La clase AsyncTask se instancia a través de uno de sus constructores

```
AsyncTask(unsigned long millisInterval);
AsyncTask(unsigned long millisInterval, Callback callback);

AsyncTask(unsigned long millisInterval, bool autoReset);
AsyncTask(unsigned long millisInterval, bool autoReset, Callback callback);
```

Uso AsyncTask

```
// Iniciar, parar, o reiniciar la AsyncTask
void Start();
void Stop();
void Reset();

// Si es AutoReset, la AsyncTask se reinicia al expirar
bool AutoReset;

// Actualiza el estado de la AsyncTask
// Esta funcion debe ser llamada frecuentemente desde el bucle principal
bool Update();
void Update(AsyncTask &next);

// Establecer el intervalo
void SetIntervalMillis(unsigned long interval);
void SetIntervalMicros(unsigned long interval);

// Estado de la AsyncTask
unsigned long GetStartTime();
unsigned long GetElapsedTime();
unsigned long GetRemainingTime();
bool IsActive() const;
bool IsExpired() const;
unsigned long Interval;

// Funcion de Callback al expirar la tarea
AsyncTaskCallback OnFinish;
```

Ejemplos

La librería AsyncTask incluye los siguientes ejemplos para ilustrar su uso.

- SimpleTask: Ejemplo que muestra una función temporizada sencilla

```
#include "AsyncTaskLib.h"

#define DEBUG(a) Serial.print(millis()); Serial.print(": "); Serial.println(a);

AsyncTask asyncTask(2000, true, []() { DEBUG("Expired"); });

void setup()
{
	Serial.begin(9600);
	Serial.println("Starting");

	asyncTask.Start();
}

void loop()
{
	asyncTask.Update();
}
```

- BlinkWithoutDelay: Ejemplo que hace parpadear el Led de la placa de forma no bloqueante, con funciones Lambda

```
#include "AsyncTaskLib.h"

AsyncTask asyncTask1(1000, []() { digitalWrite(LED_BUILTIN, HIGH); });
AsyncTask asyncTask2(2000, []() { digitalWrite(LED_BUILTIN, LOW); });

void setup()
{
	pinMode(LED_BUILTIN, OUTPUT);

	asyncTask1.Start();
}

void loop()
{
	asyncTask1.Update(asyncTask2);
	asyncTask2.Update(asyncTask1);
}
```

- AlternateTask: Ejemplo que muestra como hacer una función alternante (sin funciones Lambda)

```
#include "AsyncTaskLib.h"

#define DEBUG(a) Serial.print(millis()); Serial.print(": "); Serial.println(a);

AsyncTask asyncTask1(1000);
AsyncTask asyncTask2(2000);

void action1()
{
	DEBUG("Expired 1");
	asyncTask2.Start();
}

void action2()
{
	DEBUG("Expired 2");
	asyncTask1.Start();
}

void setup()
{
	Serial.begin(9600);
	Serial.println("Starting");

	asyncTask1.OnFinish = action2;
	asyncTask1.OnFinish = action1;
	asyncTask1.Start();
}

void loop()
{
	asyncTask1.Update();
	asyncTask2.Update();
}
```