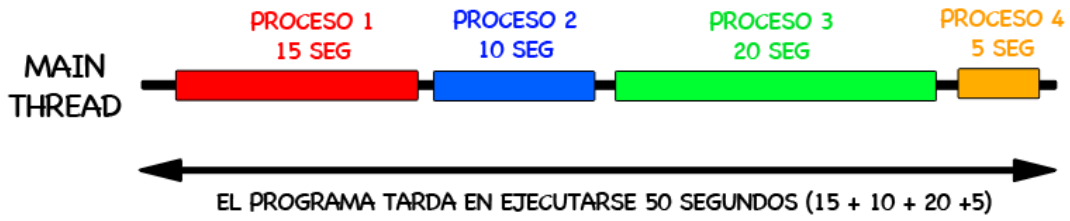
NXA

LUA APPs

Todas las funciones LUA son invocadas por el servidor de tareas, los eventos predefinidos (callbacks de la Sección 4.6.10), las funciones propias del sistema así como las funciones creadas en los scripts LUA creados por los usuarios, son ejecutadas en el mismo contexto.

Esto quiere decir que las funciones se ejecutan en el mismo hilo, lo que provoca que se ejecuten de forma secuencial.



Esta forma de ejecución tiene el problema que para la realización de una tarea anteriormente tiene que haber terminado todas las anteriores. Esto en algunos momentos puede que no nos convenga.

Podemos querer realizar la ejecución de varias funciones en paralelo, para ello hacemos uso de la función de nxa startApp.

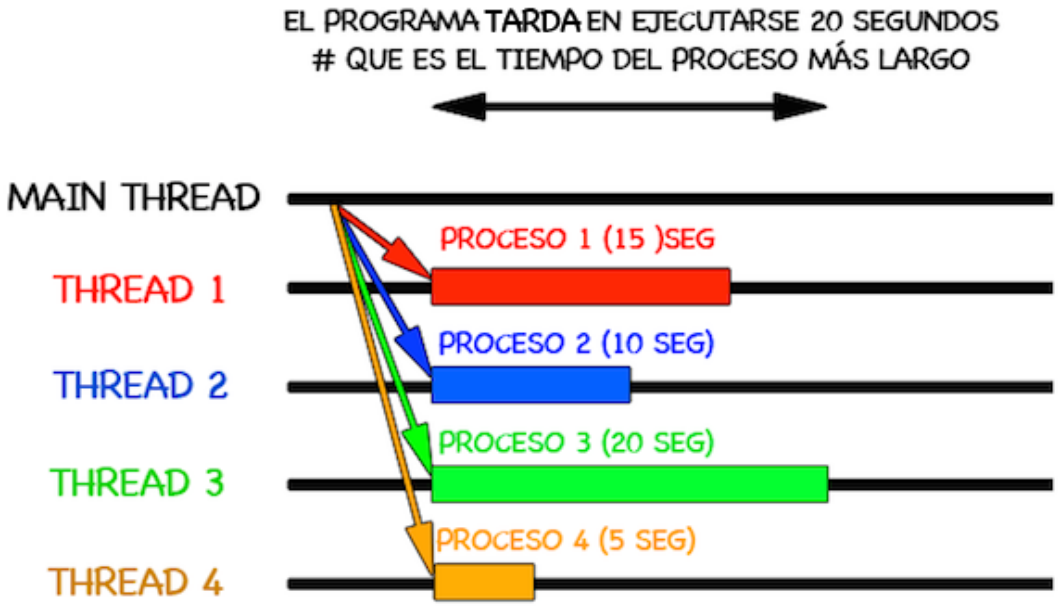
Por ejemplo podríamos realizar en un hilo una tarea periódica sin que interfiriera en la ejecución de cualquier otra tarea.

Esta función tiene como entrada dos parámetros:

- Archivo LUA: Se le introduce el nombre del archivo LUA (.lua) donde esté definida la función que queremos ejecutar en un hilo independiente.
- Función LUA: Se le introduce la función que quiere ser invocada en un hilo independiente.

Cuando esta función es llamada se encarga automáticamente de crear un hilo independiente del resto de la ejecución, en dicho hilo ejecuta la función que le hayamos metido por parámetros.

Este tipo de ejecución tiene la ventaja de no bloquear cualquier otra tarea, si no que ambas se hacen paralelamente.



Esta ejecución en paralelo se realiza en medida de lo posible. Hay que tener en cuenta que en cada paso no se realiza solo la ejecución de una tarea, si no que las tareas se van ejecutando en bloques dependiendo de la capacidad del dispositivo así como del tamaño de las funciones a tratar.

Una vez terminada la función del hilo este se cierra automáticamente sin necesidad de "matar" el hilo por parte del usuario.

A continuación se mostrará un ejemplo de uso de esta función:

Creamos dos funciones. En **uno()** realizamos una espera activa la cual no hace nada, tan solo es para que espere.

La función **dos()** simplemente imprime por pantalla un mensaje.

```
function uno()
        nxa.LogInfo("funcion 1")
        for i=0, 5000000,1 do
        end
        nxa.LogInfo("funcion 1.1")
end
```

```
function dos()
        nxa.LogInfo("funcion 2")
end
```

Una vez creadas dichas funciones pasamos a ejecutar ambas en hilos independientes, para ello escribimos el siguiente código en la herramienta "Execute LUA script":

```
nxa.StartApp(uno())
nxa.StartApp(dos())
```

Primero creará un hilo con la función **uno()** y empezará a ejecutarse, al entrar en la espera activa (la cual no hace nada) y ser un hilo se ejecutara también la función **dos()** antes de la finalización de esta.

El resultado sería el siguiente:

"funcion 1"

"funcion 2"

"funcion 1.1"

En lugar de este sencillo ejemplo podemos utilizar estos hilos para realizar una espera activa a algún tipo de variable como puede ser un sensor de luminosidad.

Uno de los hilos está continuamente comprobando el nivel de luz que nos proporciona el sensor, cuando el nivel de luz sea lo suficientemente bajo procederemos a ejecutar cualquier acción (endencer una luz artifical).

Al estar en otro hilo, la espera activa no interfiere en el comportamiento del resto del sistema, pudiendo realizar cualquier otra tarea que sea necesaria.

Dentro de estos hilos hay que tener cuidado con el tipo de funciones que se utilizan. Se debe tener en cuenta que algunas funciones son críticas por lo que no pueden realizarse dentro de estos hilos. Estas funciones son:

nxa.LogInfo

nxa.LogWarning
nxa.LogError
nxa.IsInitialized
nxa.IsRunning
nxa.IsSimulation
nxa.IsActiveServer
nxa.IsMainServer
nxa.IsBackupServer
nxa.GetValue | nxa.Value
nxa.SetValue
nxa.SetItemData
nxa.WriteValue
nxa.ReadValue
nxa.IsValidValue

nxa.WorkspaceName
nxa.RootPath
nxa.WorkspacePath
nxa.ScriptFilesPath
nxa.DataFilesPath
nxa.LogFilesPath
nxa.ProjectFilesPath
nxa.EventFilesPath
nxa.ConfigFilesPath
nxa.GetItemID
nxa.Sleep
nxa.Now
nxa.TimeOnly
nxa.DateOnly
nxa.Year
nxa.Month
nxa.Day
nxa.Hour
nxa.Minute
nxa.Second
nxa.DayOfWeek
nxa.MakeDate
nxa.MakeTimeOnly
nxa.AddDate
nxa.DiffDate
nxa.DateToString
nxa.SetDate
nxa.SetTimeOnly
nxa.GetVar
nxa.SetVar
nxa.ClearVar
nxa.AddVarTask

nxa.SourceVar
nxa.GetPropertyValue | nxa.PropertyValue | nxa.Property
nxa.SetPropertyValue | nxa.SetProperty
nxa.GetLastErrorText
nxa.GetLastErrorCode
nxa.LowByte
nxa.HiByte
nxa.LowWord
nxa.HiWord
nxa.IsBitSet

nxa.SetBit
nxa.ResetBit
nxa.SetLowByte
nxa.SetHiByte
nxa.SetLowWord
nxa.SetHiWord
nxa.And
nxa.Or
nxa.Xor
nxa.Tor
nxa.Not