Unofficial

Genelec Smart IP SIMPL+ Modules and Simpl Sharp Pro libraries

Developed by Niklas Olsson - JaDeVa AB

How to download the SIMPL+ modules and SIMPL Demo program

1. Press the green **Clone or download** button in the top right, and select **Download ZIP**
2. Open up the file and open the **SIMPL** directory.
3. Use the `GenelecSpeakerDemo_CP3_compiled.zip`

Summary

This repository contains

- SIMPL+ module
  - `Genelec_Speaker_x.x.x_SE` - Module to control Genelec Speakers
- SimplSharp (C#) solution containing projects for both SIMPL+ modules and Simpl Sharp Pro
  - `Specialelektronik.Products.Genelec.SmartIp`
    - This project contains the library for controlling Genelec Speakers utilizing the Smart IP protocol.
  - `Specialelektronik.Products.Genelec.Test`
    - This is a S#Pro Demo program. Load this program to your processor and use the Xpanel `Genelec_Speaker_Xpanel.vtp` to test the libraries.
  - `Genelec_Speaker_CSharp`
    - This project contains wrapper classes that is used for the SIMPL+ modules.
- SIMPL Windows Demo program
- VTPro-E project to be used with both Simpl Sharp Pro and SIMPL Windows demo programs.
- Documentation for all SIMPL+ modules, as well as this README as a PDF

This readme is mainly focused on information on how to use `Specialelektronik.Products.Genelec.SmartIp.GenelecSpeaker`.

Dependencies

You need to add the following reference to you project

```
SimplSharpNewtonsoft
```

Important notes and Known issues

- If you have Genelec's Smart IP Manager open at the same time as the processor is controlling the speaker things might not work as it should. Some things we've noticed
    - If the processor sets the speaker to standby, it wakes up again after a couple of seconds.
    - The speaker shows up as offline in Smart Ip Manager.

Getting started

- Clone the repository.
- Open the solution in the `SIMPL SHARP` directory.
- Build the solution.

Start up your S#Pro project and do the following:

- Add references to the dependencies shown in the Dependencies section of this readme.
- Add a reference to `Specialelektronik.Products.Genelec.SmartIp.dll` that you built in the steps above.
- The namespace for the library is `Specialelektronik.Products.Genelec.SmartIp`.

Genelec Speaker

Class name: `GenelecSpeaker`

This class integrates with Genelec speakers utilizing the Smart IP protocol. Testing has been performed on a Genelec 4430A.

Quickstart

```
// Instantiate the speaker
var device = new GenelecSpeaker("192.168.10.128");

// Subscribe to events
device.Events += new EventHandler<GenelecSpeakerEventArgs>(_device_Events);

// Start polling
device.StartPolling();

void _speaker_Events(object sender, GenelecSpeakerEventArgs e)
```

```
{
    switch (e.EventType)
    {
        case GenelecSpeakerEventArgs.eEventType.Responding:
            bool responding = e.BoolValue;
            break;
        case GenelecSpeakerEventArgs.eEventType.LevelDb:
            double levelDb = e.DoubleValue; // A value between -130.0 and 0.0
            break;
        case GenelecSpeakerEventArgs.eEventType.LevelPercent:
            double levelPercent = e.DoubleValue; // A value between 0.0 and 1.0
            break;
        case GenelecSpeakerEventArgs.eEventType.Mute:
            bool mute = e.BoolValue;
            break;
        case GenelecSpeakerEventArgs.eEventType.DeviceInfo:
            string model        = e.DeviceInfo.Model;
            string firmwareId   = e.DeviceInfo.FirmwareId;
            string build        = e.DeviceInfo.Build;
            string baseId       = e.DeviceInfo.BaseId;
            string hardwareId   = e.DeviceInfo.HardwareId;
            string category     = e.DeviceInfo.Category;
            string technology   = e.DeviceInfo.Technology;
            string apiVersion   = e.DeviceInfo.ApiVersion;
            break;
        case GenelecSpeakerEventArgs.eEventType.PowerState:
            eGenelecSpeakerPowerState powerState = e.PowerState;
            break;
        case GenelecSpeakerEventArgs.eEventType.Poe15W:
            bool poe15w = e.BoolValue;
            break;
        case GenelecSpeakerEventArgs.eEventType.AllocatedPower:
            double allocatedPower = e.DoubleValue;
            break;
        case GenelecSpeakerEventArgs.eEventType.Profile:
            int profile = e.IntValue; // A value between 0 and 5
            break;
    }
}
```

Constructors

- `GenelecSpeaker(string ip)` - Uses default port 9000, username *admin* and password *admin*.
- `GenelecSpeaker(string ip, string username, string password)` - Uses default port 9000.
- `GenelecSpeaker(string ip, int port, string username, string password)` - `ip` can be either the ip address or hostname of the device to control.

Methods

- `CustomGet(string url)` - Makes it possible to send custom commands to the device. This is a GET-request. Url example: `public/v1/audio/volume`. It returns with the response from the device.

- `CustomSet(string url, string body)` - Makes it possible to send custom commands to the device. This is a PUT-request. Url example: `public/v1/audio/volume`, body example: `{"level":-20.5}`. Returns true if the command was accepted.
- `Dispose()` - Used to clean up timers and connections. This must be called when your program stops.
- `PollDeviceInfo()` - This polls the device for information about the device such as Model and Firmware. When the device has responded, `Events` will be called with `eEventType.DeviceInfo`.
- `PollPowerAndAudio()` - This polls the device for the following properies: `PowerState`, `AllocatedPower`, `Poe15W`, `LevelDb`, `LevelPercent` and `Mute`. This is the same poll as `StartPolling()` does. When the device has responded, `Events` will be called with the new data.
- `SetProfile(int profile, bool loadOnStartup)` - Restore profile from flash and set it as an active profile. `profile` can be a value between `0` and `5`. If `loadOnStartup` is true, that profile will be loaded after a power reset.
- `StartPolling()` - Starts polling for Power and Audio. The poll rate can be set with PollRateMs, but defaults to 5000 ms. This polls the same as `PollPowerAndAudio()`.
- `StopPolling()` - Stops polling the device.

Properties

- `AllocatedPower` - Gets the power allocated by PoE PSE (switch). Use `PollPowerAndAudio()` or `StartPolling()` to update this property.
- `Debug` - Enables debugging messages to console.
- `DeviceInfo` - The last polled information about the device. Use `PollDeviceInfo()` to update this property.
  - `ApiVersion` - API version. Example: *v1*
  - `BaseId` - Platform software version number in format major.minor.rev. Example: *1.0.0*
  - `Build` - Committed GIT revision number. -modif means that uncommitted source code is used when creating firmware. Example: *c5ca14*
  - `Category` - Category. Example: *SAM_2WAY*
  - `ConfirmFirmwareUpdate` - New firmware is running and waiting for confirmation from user. Bootloader reverts backup firmware during next reboot if confirmation is not done.
  - `FirmwareId` - Firmware identification number in format model_base-major.minor.revbuild_date_and_time. Example: *44x0-1.1.11-202007021238*
  - `HardwareId` - Hardware version string.
  - `Model` - Device model name. Example: *4430*
  - `Technology` - Technology. Example: *SAM_IP*
  - `UpgradeId` - Compability information for upgrading firmware
- `Ip` - The IP address or Hostname of the device to control.
- `IsResponding` - Returns true if the device responded to the last command.
- `LevelDb` - Get or set the volume level in Db. Range `-130.0` to `0.0`
- `LevelPercent` - Get or set the volume level in percentage. Range `0.0` - `1.0`
- `Mute` - Get or set the mute state.
- `Password` - The password of the device. Default: `admin`
- `Poe15W` - Returns true if PoE PD (loudspeaker) limits current consumption to 15W. Returns `false` if full power is needed (30W). Use `PollPowerAndAudio()` or `StartPolling()` to update this property.
- `PollRateMs` - This sets how often the device will be polled when using StartPolling(). Default: `5000` (ms)
- `Port` - The port number to connect to. Default: `9000`

- `PowerState` - Get or set the power state. You can only set it to Active, Standby or Boot. Boot will reboot the speaker.
- `Username` - The username of the device. Default: `admin`

Events

- `Events` - This will trig when any of the properties change. The event args contains:
  - `EventType` - An enum telling you which property changed.
  - `BoolValue` - Contains the new value of the property for event types `Responding`, `Mute` and `Poe15W`.
  - `DeviceInfo` - Contains the new value of the property for event type `DeviceInfo`.
  - `DoubleValue` - Contains the new value of the property for event types `LevelDb`, `LevelPercent` and `AllocatedPower`.
  - `IntValue` - Contains the new value of the property for event type `Profile`.
  - `PowerState` - Contains the new value of the property for event type `PowerState`.

Release notes

1.0.0 (Initial version)

- Supports Genelec Speakers utilizing the Smart IP protocol (`GenelecSpeaker`)