C# Developer QuickStart

# Introduction

This document describes the controller, GUI and other code written in C# and using the .NET 3.5 framework. Before you start reading this you should preferably have read the conceptual introduction of the framework. The purpose of this document is to clarify the different C# projects and their use. Important concepts specific to a solution will be explained as well. For a more in depth explanation of the source code, head over to the code documentation.

When developing the framework, please make sure you have met all the requirements as described in the respective document.

Contents

[Introduction 1](#_Toc232967154)

[Opening the solution 3](#_Toc232967155)

[Solution Overview 4](#_Toc232967156)

[GUI & Controller / Engine 4](#_Toc232967157)

[Localization 4](#_Toc232967158)

[Tools 4](#_Toc232967159)

[Simulation 4](#_Toc232967160)

[Projects Overview 5](#_Toc232967161)

[Elab.Rtls.Engines.WsnEngine.GUI 6](#_Toc232967162)

[Requires 6](#_Toc232967163)

[Elab.Rtls.Engines.WsnEngine 6](#_Toc232967164)

[Requires 6](#_Toc232967165)

[Elab.Rtls.Engines.WsnEngine.EngineForm 6](#_Toc232967166)

[Requires 6](#_Toc232967167)

[Elab.Rtls.Engines.WsnEngine.Tools.SocketConnection 7](#_Toc232967168)

[Requires 7](#_Toc232967169)

[Elab.Rtls.Engines.WsnEngine.Tools.Logger 7](#_Toc232967170)

[Elab.Rtls.Engines.WsnEngine.Tools.DatabaseConnection 7](#_Toc232967171)

[Requires 7](#_Toc232967172)

[Elab.Rtls.Engines.WsnEngine.Simulation 7](#_Toc232967173)

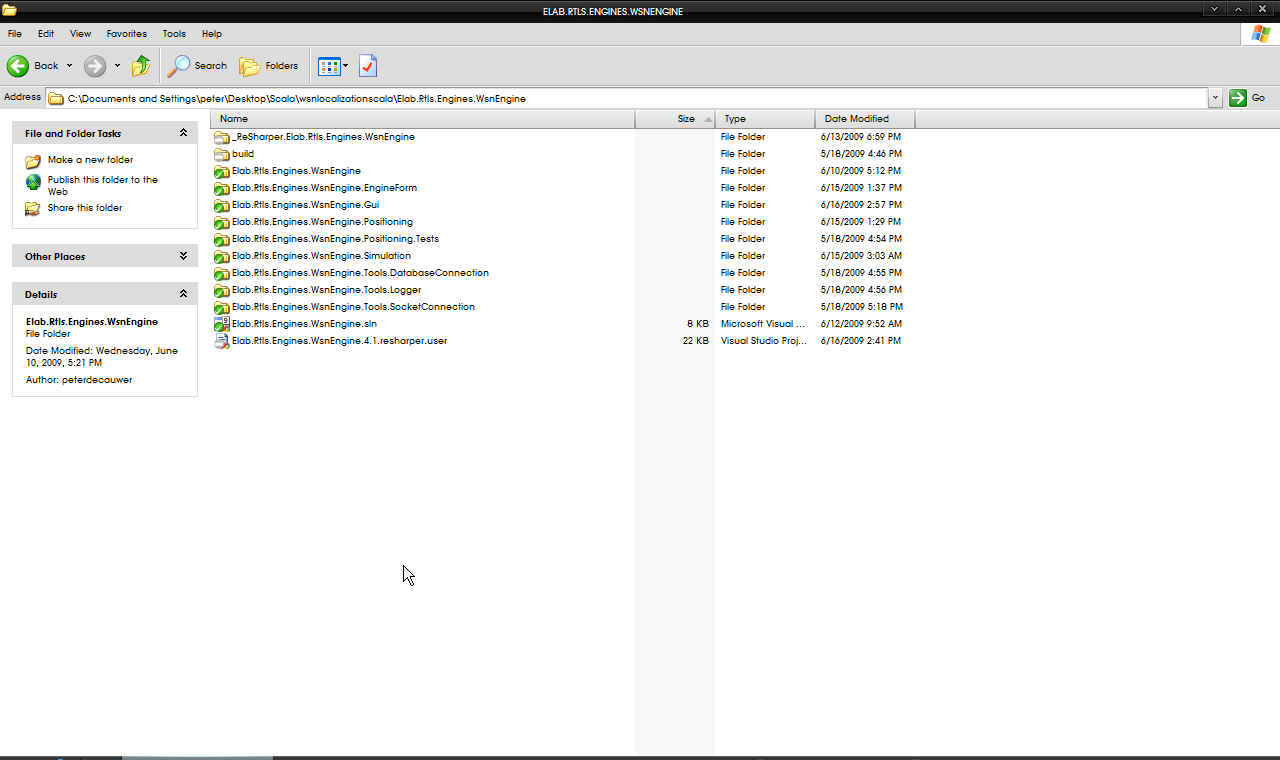
[Elab.Rtls.Engines.WsnEngine.Positioning 7](#_Toc232967174)

[Elab.Rtls.Engines.WsnEngine.Positioning.Tests 7](#_Toc232967175)

[External Dependencies 9](#_Toc232967176)

# Opening the solution

1. Browse to the directory containing the source code, this should be the directory /Final Product/Source
2. Double click the file: Elab.Rtls.Engines.WsnEngine.sln to open the Visual Studio Solution



# Solution Overview

When you have opened the solution, you can see the solution and the projects in the solution explorer. These projects can be divided into logical groups:

## GUI & Controller / Engine

The GUI, Controller / Engine form the part of the framework as described in the conceptual overview.

## Localization

This division should contain all the logic for the localization

## Tools

These tools are in place to speed up the development

## Simulation

Though very closely related to a unit test, this division serves the purpose to process captured data and analyze it with the different algorithms from the localization part

# Projects Overview

This section explains the functionality of every project on a high level.

The GUI is constructed in the Elab.Rtls.Engines.WsnEngine.GUI project.

The Scala Engine / Controller consists out of 2 projects:

* Elab.Rtls.Engines.WsnEngine
* Elab.Rtls.Engines.WsnEngine.EngineForm

The attentive reader may notice that there are three more logical subdivisions; their respective projects are listed here:

* Tools
  + Elab.Rtls.Engines.WsnEngine.Tools.SocketConnection
  + Elab.Rtls.Engines.WsnEngine.Tools.Logger
  + Elab.Rtls.Engines.WsnEngine.Tools.DatabaseConnection
* Simulation
  + Elab.Rtls.Engines.WsnEngine.Simulation
* Localization
  + Elab.Rtls.Engines.WsnEngine.Positioning
  + Elab.Rtls.Engines.WsnEngine.Positioning.Tests

## Elab.Rtls.Engines.WsnEngine.GUI

The purpose of this project should be pretty clear, this is the GUI! The function of the GUI is to monitor the data coming from the WSN and to set the state of the nodes.

### Requires

* Elab.Rtls.Engines.WsnEngine.Tools.SocketConnection
* ZedGraph

## Elab.Rtls.Engines.WsnEngine

This is the core of the system, what keeps it running. The functionality can further be divided into two parts

* Controller
* Engine

As should normally be clear from the conceptual overview: the controller is the core of our framework and the engine is what interfaces the controller with the Scala interface (interface is becoming a buzzword these days) It actually translates the commands and fields accounted for in the Scala interface to functions & fields more native to the controller

### Requires

* Castle.ActiveRecord
* Elab.Rtls.Engines.WsnEngine
* Elab.Toolkit.Core
* Elab.Toolkit.Rtls
* Scala.Core
* NHibernate

## Elab.Rtls.Engines.WsnEngine.EngineForm

This is the controller’s GUI. Here the user can select a localization algorithm, RSS filter and method of calibration. This project also forms the WCF host for the engine.

We can conclude that this project houses the human (GUI) and Scala interface. This project does not contain any logic of the controller, but is merely a wrapper around it with which the user can interact.

### Requires

* Castle.ActiveRecord
* Elab.Toolkit.Core
* Elab.Toolkit.Rtls
* Elab.Toolkit.Imaging
* Scala.Core
* Elab.Rtls.Engines.WsnEngine.Tools.SocketConnection
* All the tools of this solution

## Elab.Rtls.Engines.WsnEngine.Tools.SocketConnection

This project adds an abstraction to the use of sockets within the .NET framework. With this tool it is possible to set up a TCP connection as a client or server with a single function call!

### Requires

Elab.Rtls.Engines.WsnEngine.Tools.Logger

## Elab.Rtls.Engines.WsnEngine.Tools.Logger

This tool has some logging functions. Fairly limited

## Elab.Rtls.Engines.WsnEngine.Tools.DatabaseConnection

As with the socketconnection tool, this form an abstraction upon the ADO.NET functionality. If you seek to expand database functionality you should put it in this project. By using this project you can perform a query with a single function call.

### Requires

Elab.Rtls.Engines.WsnEngine.Tools.Logger

## Elab.Rtls.Engines.WsnEngine.Simulation

This projects does not make part of the normal way of execution. From that point of view this project is separated from the others.

The purpose of this project is to run the localization algorithm without having to setup a fully working WSN and framework. It processes raw RSS from the localization table and uses this as input for the localization algorithms. The output of these algorithms is written to a log file.

## Elab.Rtls.Engines.WsnEngine.Positioning

This projects houses all the positioning algorithms.

## Elab.Rtls.Engines.WsnEngine.Positioning.Tests

This projects does not make part of the normal way of execution. From that point of view this project is separated from the others.

With this project you can perform unit tests on some of the algorithms in Elab.Rtls.Engines.WsnEngine.Positioning

# External Dependencies

This section describes all the external projects that are required by the solution

* Scala.Core: The core of Scala, describes the interfaces used in the engine
* ZedGraph: A popular graphing framework for .NET
  + <http://zedgraph.org/wiki/index.php?title=Main_Page>
* Castle.ActiveRecord & NHibernate: A popular ORM for .NET
  + <http://en.wikipedia.org/wiki/Object-relational_mapping>
  + <http://www.castleproject.org/activerecord/index.html>
* Elab.Toolkit: houses some basic functions commonly used. Developed at Elab by Pieter Spinneweyn & Olivier Coudeville.