§ PDI-HMI

This repository contains the PDI-HMI monitoring application.

The project's responsibilities are:

- Monitoring of preconfigured electric devices in a one-line schematic view.
- Configuration and detailed view of each configured device.
- Trending/historic view of selected devices and signals.
- System overview (summary of errors, warnings, alerts, etc.)

Other areas are:

- Customizability - Users can choose predefined themes
- Language support - The system's UI can be presented in different languages
- System can be accessed remotely.

Example devices that can be configured:

- Transformer
- Voltage Sensor
- Current Sensor

For details on configuration refer to Configuration section.

For details on the current status of the development refer to ZenHub associated with this project.

§ Project structure

Structurally code is divided into three main parts:

Core - Subproject that builds core library - a library that provides low-level functionality to Ui. Object representations, database, configuration files are handled here. C++ code.
Ui - Subproject that handles the user interface. It uses the core library to create views that allow interfacing with the application. This part contains code in both C++ and QML/JS.
Data-generator - Subproject not related to actual target code. It is an application that is used to read/write to the database used to store historic data. It's designed to be used only on the desktop platform.

§ root structure

├── Doxyfile - Configuration for automatic documentation generation
├── artifacts - Miscellaneous files (specification, other projects, legacy code)
├── doc - Documentation (yocto, database)
├── docs - Generated documentation (present only when docs are generated)
├── hmi-solution - Main project folder
└── tools - Various tools and scripts related to project ()

§ hmi-solution structure

├── _clang-format - C++ auto-formatting rules (plugged in QtCreator)
├── core - Core subproject folder. Contains source files along with configuration files
├── data-generator - Data-generator
├── format_all.sh - Script that is used to auto-format the whole project
├── hmi-solution.pro - Qt Creator top-level project file
├── hmi-solution.pro.user - Qt Creator per-user configuration
└── ui - Ui subproject folder. Contains source files

§ Tools and languages

Languages:

- C++ (compiler with C++17 support)
- QML and JS (ui)
- Python (tools)

Libraries:

- Qt framework 5.15.2
- Qt Quick 2.15 (along with other QML specific libraries like Controls, Layouts, etc.)

Tools:

- Qt Creator - main IDE
- PostgreSQL - database system used for trending

- TimescaleDB - extension for database system. Adds time-series functionality to the postgresql database.

This project uses Qt framework in version 5.15.2. Using a version lower than that may break the compilation.

§ Targets

- Raspberry Pi 4 based hardware platform
- Desktop

The main application target is raspberry pi 4 platform-based custom hardware. The code can also be compiled and tested on desktop environments, Windows, MacOS and Linux based OS were tested.

§ Raspberry Pi target

The final hardware target will be based on RPI 4 CM platform. Currently, during development, we're using an ordinary Raspberry PI 4 board.

Code can be compiled and developed for the target platform with the use of custom virtualbox virtual machine. For overview on how this is configured please refer to this file or this file. For details see this

§ Desktop target

Desktop is used only for development, debugging, and testing. It won't be the final target for this project. To compile for desktop. The first step is to configure the environment: The most straightforward path is to use qt online installer (maintenance tool) https://www.qt.io/download.

1. Install Qt Creator
    1. Make sure that the compiler for the platform is checked (mingw for windows, gcc/clang for others)
    2. Make sure that QVirtualKeyboard library is installed
    3. Make sure that Qt Charts library is installed
    4. Make sure that WebGL plugin is installed
    5. Make sure that Qt is installed in version **5.15.2** (newer versions should also be fine)
2. Open hmi-solution.pro file, configure the toolset (should be autodetected), and compile the project.

The selections should look something like this (don't mind the version of Qt here):

§ IDE

Qt Creator is the main IDE used for this project.

§ Qt Maintenance Tool

The Qt's environment comes with a handy package downloader/update-tracker called `Qt Maintenance Tool`. This tool is used to configure the development environment. It's done by selecting proper packages

and their versions and installing them.

§ Initial setup

Currently, the target uses Qt version build 5.15.2

The first step is to install proper IDE and packages: The most straightforward path is to use qt online installer (maintenance tool) https://www.qt.io/download.

§ Raspberry Pi target

What is crucial here: The versions of Qt (and its libraries) installed in the development machine **must** match those on the hardware target. (details of potential solutions are below)

§ VM

Code can be compiled and developed for the target platform with the use of a custom Virtualbox virtual machine.
For overview on how this is configured please refer to this file or this file. For details see this.

§ Native

It's also possible to compile and debug the code natively using Windows. A commercial Qt license is required for that.

Firstly, head to https://account.qt.io/downloads and log in to see the list of downloads

Select proper Qt version of the package 

The file can then be installed in Qt Maintenance Tool creator.

The package will contain all necessary libraries to build the project.

§ Desktop target

1. Install Qt Creator
    1. Make sure that the compiler for the platform is checked (mingw for windows, gcc/clang for others)
    2. Make sure that QVirtualKeyboard library is installed
    3. Make sure that Qt Charts library is installed
    4. Make sure that Qt is installed in version **5.15.2** (newer versions should also be fine)
2. Open hmi-solution.pro file, configure the toolset (should be autodetected), and compile the project.

§ Configuration

The project is configured using grid.yaml, and panelboard.yaml files. For simulation purposes telemetry.yaml file can be used. Default files for below configuration are stored here

§ grid.yaml

grid.yaml file is used to configure:

- Type (eg. *Transformer*), position (on the one-line view), and configuration (label, and type-specific options) of hardware devices.
- Theme
- Login screen widgets

For details see readme-grid.md

§ panelboard.yaml

panelboard.yaml file configures the connections between adjacent breakers.

For details see readme-panelboard.md

§ telemetry.yaml

Telemetry.yaml file can be used to simulate data inputs to the application.
This file specifies sensors along with their specific signal or option configuration.
Telemetry is read every 2 seconds and the device status is updated.

For details see readme-telemetry.md

§ logo

To change the default logo that is visible on the dashboard screen, add a file in the same location as the other configuration files are located. The filename has to be "logo.EXTENSION", where the possible extensions are: svg, png, jpg.

§ Language support

The default language of the interface is English. The user can select from a few predefined languages on the configuration page. For details on how to add more languages, or add translations to existing ones, see here

§ Logs

Logs are stored in the logs.txt file. They can be viewed in the application only when logged in as an EatonAdmin. The structure of the logs.txt file is:

- first row contains information about the offset of the first (oldest stored) log and the number of logs in the file. This information is updated each time a new log is added.
- the rest of the file is filled with logs. Each log consist of:
    1. timestamp
    2. type of log
    3. length of text
    4. text

- the logs.txt file has a maximum size that will be exceeded by no more than length of a single log. When this max size is reached new logs will start replacing the oldest ones.