Short descriptions of key program files of the JouKo-control device

The main program components of the device

main_loop.py – main program

- Main program, which starts automatically when the device starts up. This main program manages the measurement, break management and communications. This program calls other program components depending on how the functionality of the device is specified in json files (see descriptions of JSON files, eg settings in 'asetukset.json')
- The program is executed as different alternating sequences, as the functions are not desired to run in parallel. In this way, the optimum state of the measurement is achieved with a limited processor power while the measurement process is not performed simultaneously with other functions. The result is consistent and high-quality measurement as well as the highest possible sampling rate as the measurement process always uses all possible processor power.

GPIOhallinta.py - GPIO definitions and IO commands

- IO channel definitions and commands for reading and controlling IO. Reading and guiding GPIO is mainly defined in this program component. Only using SPI and UART via GPIO are their separate components (see mittausMCP3008.py ('measure') and Communication)

mittausMCP3008.py – Current and voltage measurements with the MCP3008 AD converter

- Reading the current and voltage measurements via the GPIO SPI bus to the MCP3008 AD converter. The program component defines measurement reading, measurement calculations, calibration the measurements, etc.

Communication

Different components are used in communication, depending on the device configuration. The communication program components for GPRS, LORA, BT and ETHERNET are described below.

One of the main priorities has been minimizing the accumulating costs of constant data transfer. This is also committed by minimizing the message size while communicating between the server and the the control device. In addition, the message size is limited by the allowed maximum size of the message when using Lora data transfer. When LoRa communication uses SF12 (Spreading factor), the maximum payload size of data to be sent in a message is 51 bytes.

When a good and fast data connection is available in LoRa communication, the message size may be 241 bytes in the "Lora EU" message. However, when a message is sent from the server to the LoRa control device, it can not be assumed that the high-speed data connection is in use, as this would cause the entire message to fail to be transmitted to the control device.

GPRScom.py - communication via GPRS radio

- GPIO UART communication with the SIM800F GPRS radio is defined in this program component. This component is only used if the SIM800F GPRS radio is in use.

LORAcom.py - communication via LoRa radio

- GPIO UART communication with the RM186 LoRa radio is defined in this program component. This component is only used if the RM186 LoRa radio is in use.

BTcom.py - communication via Bluetooth radio (not tested)

- The program for BT communication was not tested because no Laird RM186 chip with firmware for peripheral device was obtained.
- UART communication with BT radio via the GPIO is defined in this program component. This component is only used if the device is a control device with the role of BT master or, if the device is switching device that is working as a BT slave.

ETHERNETcom.py - Communicating through the device's built-in Internet connection

- The device could also be used on the device via a pre-established internet connection. This program component defines the communication of the device through the existing Internet connection. An Internet connection can be, for example, a WLAN or other connection to a device.
- Note! This component is intended for test use only. Continuous use of a fixed Internet connection would require a number of changes to your device to ensure its data security.

viestinpurku.py - server message management

- This program manages the use of protobuf messages used in communication between the control device and the server. The functions of the laiteviestit_pb2.py are used to encode and decode the protobuf messages.

laiteviestit_pb2.py - server message management with protobuf

- This program component encodes and decodes protobuf messages used in communication between the control device and the server.

katkoSQL.py – power interruption (break) management using SQL database

- This program component includes all the functions used to control the power interruption database.

valvoja.py - control device operating watchdog

JSON-file descriptions

The functional settings of the device are loaded from JSON files when initiating the main program. Settings divided into several different files so that update procedure can be limited to updating only the desired features. The settings can be activated or deactivated by setting the value as ‘true’ or ‘false’ or by changing the numeric values.

1 - asetukset.json - general settings for all devices: the durations of the measurement cycles and communication cycles, definition of the processor measurement reading speed

2 - kommunikaatio.json - communication settings for radio communications: defines the communication channel (GRPS, Lora or ethernet) and also settings such as addresses, device names, and communication keys.

3 - testiasetukset.json - test settings that useful for further development. Test settings can be easily activated or deactivated by setting the values ‘true’ or ‘false’ eg. speeding up the testing process by shortening delays.

4 - kalibrointi.json - calibrated values for the hardware to ensure accuracy of measurements. The automatic zero-level calibration of the current measurement also stores the values here. The automatic zero-current calibration can be activated here.

5 - paivityksenTila.json - defines the update settings and the version number of the current software.

Additionally json files exist for other special purposes such as calibration and operation controlling.

6 - toimintatila.json - Select the operating mode of the device for running calibration measurements: calibrate the voltage measurement (‘kalibroidaanJannite’), calibrate the current measurement (‘kalibroidaanVirta ‘) and verify the calibration (‘varmennetaanKalibrointi’).

7 - ohjelmanTila.json - storage that is used to tell about the correct operation of the device.