Genesis of the project
I considered purchasing the Acqua Micronova IoT interface (i.e. the Wifi interface sold by the various pellet machine manufacturers) to be able to monitor and control the stove remotely.

I carried out some research to evaluate the product and found that an essential part was missing for me, namely the management of the historical signals made available by the boiler.

Hence the decision to create a monitoring and control system [called **Supervisor**] for Micronova controllers.

For the realization of the project I took inspiration from the great work already done by others, specifically I cite the following sources

- www.ridiculouslab.com
  https://www.ridiculouslab.com/arguments/iot/stufa/micronova.php
- Philibert Cheminot https://github.com/philibertc/micronova_controller

Through the analysis of the signal and user history I was able to optimally configure (in my opinion and according to my needs) my boiler [I have a Klover Ecompcat 190 connected directly to the collectors] which currently manages to work in modulation continuously (without bringing the water temperature too high and without dropping too much with that of the fumes with a live flame even in modulation) guaranteeing

- high thermal comfort
- flue gas and water working temperatures within the set ranges
- low consumption

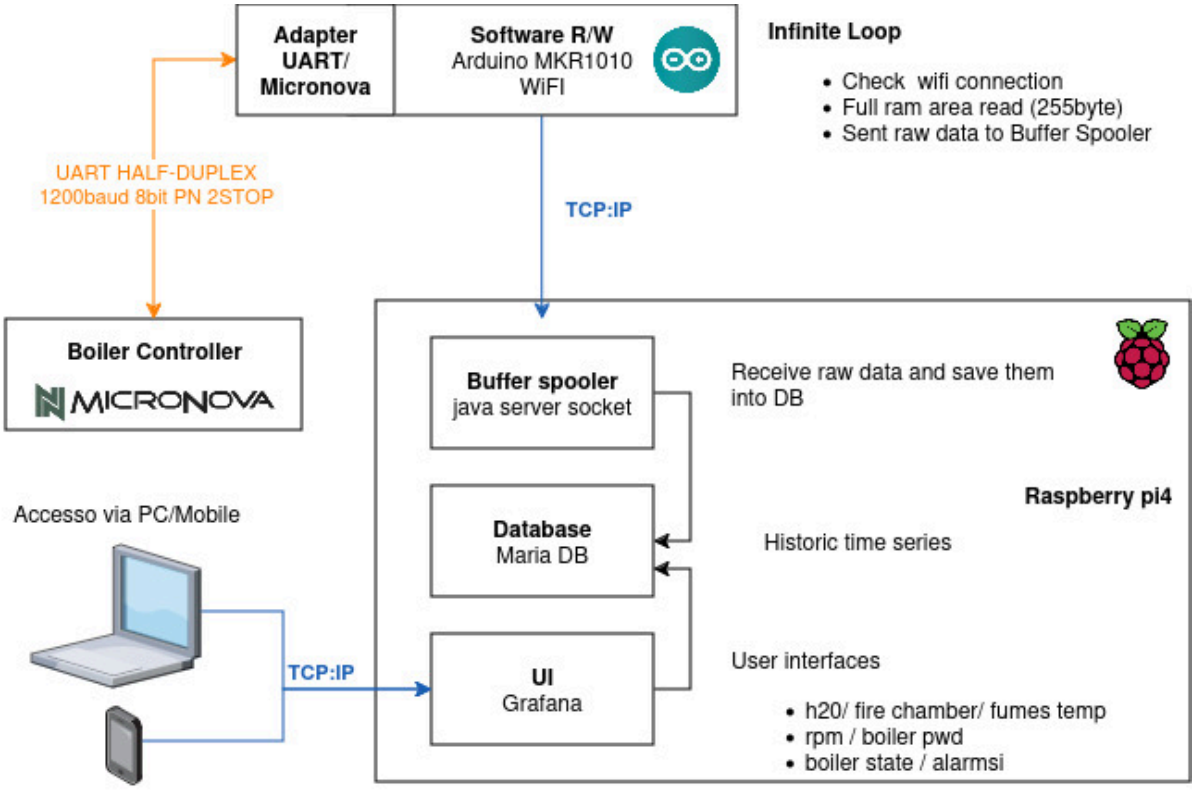
Some considerations

- I undertook this project for an exclusively functional aspect, that is, to have the data to carry out analyzes (initially to understand the response of the boiler to the various parameterisations) supported by objective data. If the evaluation is strictly economic and the need is to have a remote control system, it is better to opt for a commercial product.
- I intend to develop further functions, among these the automation of the setup of some work parameters (in addition to the thermostat relay, currently activated by a room thermostat) taking into consideration
  - external temperature
  - internal temperature
  - pellet type
  - internal temperature trend

Architecture
The system, at a macroscopic level, is modularized by identifying the Supervisor component on one side, and the Micronova controller component on the other,

The Supervisor consists of the following modules

- data reading/writing system from Micronova controller
    - UART/MICRONOVA adapter
    - Software R/W
- buffer spooler interfaccia seriale / database
- database for historicizing the data collected
- module for displaying historicized data on the user interface

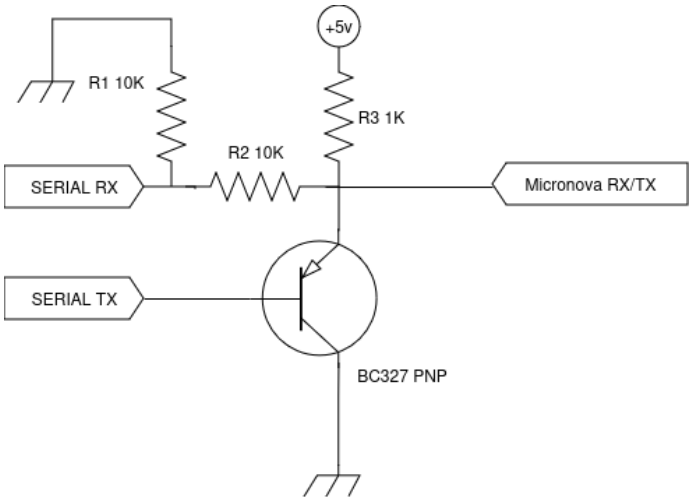


UART/Micronova adapter
Going into the merits of the data reading/writing system, it is necessary to introduce the communication specification of the Micronova controller

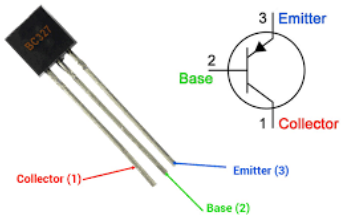
- bus UART
- Half-Duplex
- 1200 baud
- Length 8 bits
- Parity No
- Bit stop 7
- Logic levels 0-5V

the implementation of a UART/Micronova adapter is therefore required to adapt the voltage levels 0-3.3V to 0-5V

The electronic implementation is shown in the following diagram



The resistors are 1/4W.
The pinout of the BC327 transistor is shown below



Using the R1/R2 voltage divider, the value of the signal on the RX pin of the Supervisor is brought to 3.3V.

The BC327 PNP transistor is driven by the TX pin of the controller and biased via resistor R3.

Some considerations

- The above configuration is only applicable in scenarios similar to that of the project, i.e., where one component queries and the other responds. It is not applicable in scenarios in which you intend to perform RX and TX simultaneously.
- Echo effect: the messages sent by the supervisor are received by the same, therefore at a software level it will be necessary to remove the self-sent contents from the serial stream

Software R/W
The Micronova controller exposes two 255-byte memory banks

- RAM
- EEPROM

The reading process implemented in this revision of the system involves reading all RAM locations.

It is possible to read one byte at a time by sending the 0X00 command followed by the memory address (from 0x00 to 0xFF).

Therefore, to read the sixteenth byte into memory, the datagram 0x00 0x0A must be sent

The Micronova controller responds with a datagram structured like this

- Checksum calculated as follows
    - 'Requested address' && 0xFF
    - for example, given the address 0x01, the checksum will be 0x01+0xFF=0x00
- Memory location value

The reading process of the Micronova controller was implemented on an Arduino MKR1010 WiFI programmable board in order to take advantage of the latter's Wifi connectivity (in other words to avoid placing the PC near the boiler and laying a serial cable - it goes without saying that, for those who wish, it is possible to use the PC serial interface directly). The loop implemented on the board involves the following steps

- data reading from Micronova controller (and TX echo cancellation)
- wifi connectivity check
- opening client socket towards buffer spooler
- sending datagram
- socket closure

Some considerations

- in light of the fact that the meaning is not known for all memory locations, we chose to arbitrarily read all the available memory locations in such a way as to historicize the data in any case, to then enrich the analysis interfaces in the face of future decodings
- the read-only process takes approximately 25" (to avoid Micronova supervisor/controller communication errors) an arbitrary delay of 100msec has been introduced between sending the request and reading the response stream. .

Buffer spooler seriale / db
During the read cycle the data is saved in a datagram (a 255 element byte array). At the end of the reading cycle, the datagram is sent on the TCP:IP bus waiting for connections. The datagram is then managed by an internal process that inserts it into the database,

Relative to this architectural choice, the following objection could arise: why was it not chosen to save the datagram directly in the database?

The reasons that led to the choice to put a software module with the buffer function are identified in the desire to separate the reading process of the Micronova controller from the saving time in the database, known as commit time, which is unpredictable.

We have identified a scanning time of approximately 25", to which by adding the tcp:ip socket opening time and an arbitrary delay (further 35") we obtain a record every 1'. (reading more frequently does not bring any benefit given the inertia of the system and the timing of any alarms).

Database structure
The database used is MariaDB 10.11 (Mysql Fork). A simple table was created, called stove, with only two fields

- date and time record (primary key, therefore indexed)
- blob(255), which is the raw byte array read by the scanning process

Through the use of functions for string manipulation and conversion between types, it is possible to extrapolate the specific data of a portion of memory to present the extracted values in the desired format

User interface
Grafana is a software for displaying data based on time series, it provides the tools to create graphs and visualizations drawing from configurable data sources, in our case, the stove table referred to in the previous point.
The interfaces are made available in the intranet in which the supervisor is configured, for remote access it is possible to use various strategies (access via VPN from remote to local, publication on public IP or via dynamic DNS of the app,...) .

For greater convenience it is necessary to set the A **time zone UTC (Coordinated Universal Time)**
) per

- operating system
- MariaDB
- Grafana

On the Grafana dashboards it is possible to set the CET time zone during the configuration phase **Central Europe Time - Rome GMT +01** for display only)

Developments underway

- **UART/Micronova adapter**

    - Powering via voltage regulator, directly from +20V made available by Micronova controller on the connector
    - Status LED
        - Wifi (connected / not connected)
        - Response within timeout from Micronova controller
        - Sending data to Buffer Spooler
    - SD card to upload to configuration files
        - SSID & Password Wifi
        - IP:Port Buffer Spooler
- Raspberry pre-configured .iso image