Phase 2 Report | Project in Embedded Systems 15hp 1TE721

A FERMENTATION TEMPERATURE MONITORING SYSTEM using Atmel AVR and an ARM single-board computer

Group 1: August Forsman
aufo8456@student.uu.se

Summary

Phase 2 of the project has mainly consisted of

1. Tuning the UART communication between the ATmega328p MCU and Raspberry Pi
2. Choosing and configuring a suitable database and entry formatting
3. Fail safe data collection
4. Testing the web application visualization with live data

The work is finished in time according to the project plan for Phase 2.

UART communication

The serial communication between the RPi and MCU is done through the Python module pySerial which is supplied the Baud rate and port. In order to synchronize the data transfer, the temperature is fetched by sending a byte instruction `write(b'R')` followed by a `readline.decode()` in order to read the returning bytes containing a string formatted floating point temperature. The scheduled transfer rate relies on the clock of the RPi and is set in a Python script using the `time` module.

Choice of database

The document database MongoDB is chosen as the back end storage of the temperature data. The Python module PyMongo integrates the connection with the web application and the database storage. Another feature is that it is easily converted to Pandas `DataFrames` for in order to statistically analyze time-series data.

Fail safe data collection

In order to be able to gather data as fail safe as possible, the data collection script is administered by a Systemd service. Systemd is a system and service manager suite used

by Arch Linux ARM (and a majority of other UNIX-like operating systems) in order to boot, mount file systems and manage daemons to name a few.

```
 # Systemd service for logging temperature
# Use the command
#       sudo loginctl enable-linger $USER
# in order to enable the systemd instance to
# run even if the user is not logged in.

[Unit]
# Human readable unit name
Description=Reads serially from /dev/ttyUSB* and puts in MongoDB

[Service]
# Command that executes script
ExecStart=/usr/bin/python /home/alarm/Project/serial_temp_to_db.py
# Write to print() to journal
Environment=PYTHONUNBUFFERED=1
# Able to notify
Type=notify
# If it crashes with a non-zero exit code, then restart
Restart=always

[Install]
# Start service at boot
WantedBy=default.target
```

Listing 1: The systemd service that manages data collection

The service in Listing 1 utilizes the operating systems logging and prints initialization procedures as well as exception during runtime.

Web app visualization

Plotly is an open source Python module which provides the Dash web application environment.

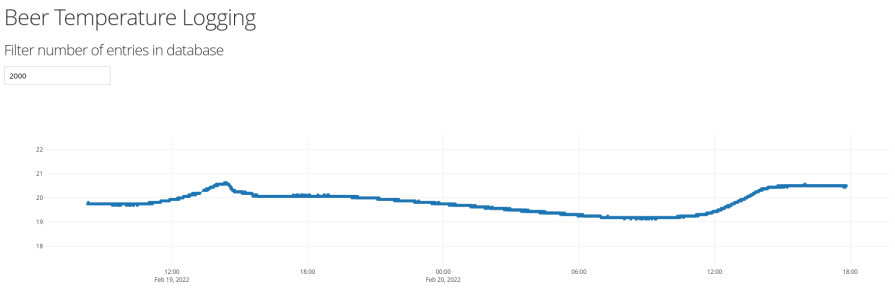

Figure 1: Temperature curve plotted in Plotly running in a Dash web application

As for now, the server hosted by the RPi is reached by a certain port accessible for units on the local network.

Conclusion and further work in the Final Phase

Further work regarding the web application is proper deployment through a service such as Flask. The full set of features that Dash has is still not known but it seems to cover some statistical features of interest. One idea of a proposed layout is a live graph containing the latest It has been discussed in the grading criteria that some kind of callback function should be applied in order to set a value that is passed to the MCU as a control parameter. However, it seems quite risky to alter the power circuits of a high voltage freezer in order to control the temperature and should be discussed with the project supervisor.