System on Chip Architecture
PIO Project Report

Grottesi Lorenzo - Bianco Nicolò
Department of Control and Computer Science
Politecnico di Torino

CONTENTS

LIST OF FIGURES

LIST OF TABLES

System on Chip Architecture PIO Project Report

***Abstract*—The industrial environment became more and more smart in the last decades. In particular some concepts were created to identify this growth. An example could be either the Internet of Things or Industry 4.0, these names are used to address the smartness reached by the devices composing the manufacturing chain. The solution discussed in this paper is thought to be placed in this context of general growth. In particular the Programmable Industrial Oven is thought to improve the reliability of food industries chains, but its applications could space in other industrial domains.**

I. Introduction

The Programmable Industrial Oven is a smart device that allows to automatize the baking process, with the least human interaction. It is based on a STM32 microcontroller that maintains a fixed temperature in the oven. The MCU works using an array of sensors for measuring the internal temperature. The control of the heating actuation is demanded to a PWM timer, that is driven by a software PID. An electrical fan is also driven by another PWM timer, which value is settled by the user interface. The operator can see the oven state on the terminal, change the temperature and activate the ventilation.

II. Background

The board used to implement the system is the STM32F429-Discovery. This board is based on the STM32F429 MCU, which supports different kind of peripherals.

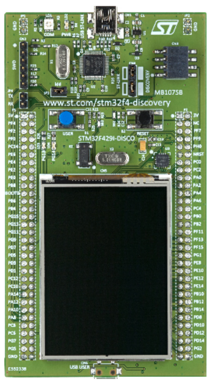

Fig. 1. The STM32F429 Board

In order to improve the controllability of the oven, a PID controller is used to drive the PWM actuation. This kind of controller bases on a negative feedback network, computing the difference between the sensed valued and the target. This difference is then processed by a proportional, integrative and derivative blocks. The three block outputs are summed up and the results is used to feed the actuation. In order to work correctly, the controller requires three parameters: Kp, Kt, Kd. The reliability of the controller depends mostly on these three.

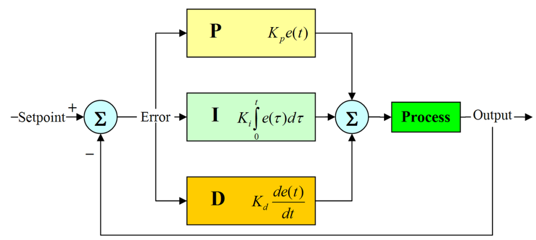

Fig. 2. PID functional scheme

III. Proposed Solutions

The solution for the Programmable Industrial Oven has been to use the STM32F429-Discovery board working with 4 temperature sensors. The sensors are BMP180, all connected on the I2C bus. The actuators for the fan and the heat mechanism are connected on two dedicated interfaces supporting PWM outputs. Given the high number of peripherals in the related

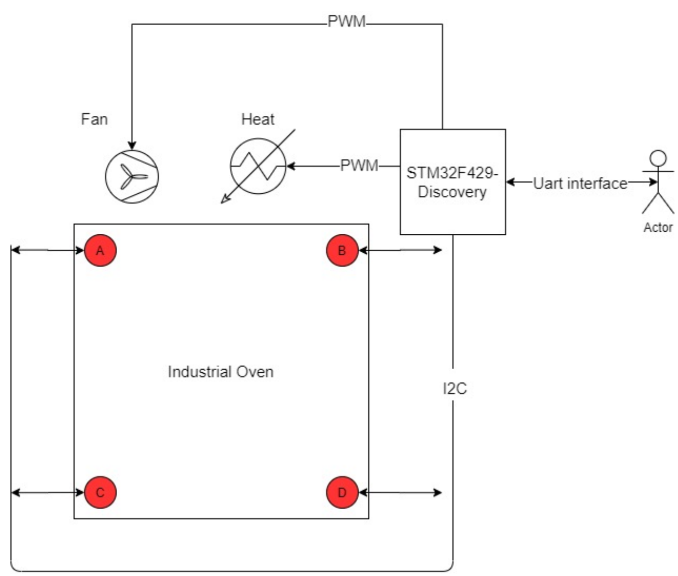

Fig. 3. Conceptual diagram

MCU, an event driven program has been written to create the system. The board has been simulated using Renode platform, which is also able to emulate the UART interface and the sensors behaviour. Going more in details, the peripherals used are:

- UART5, user interface transmission and receive.
- TIMER 2, PWM used for thermal actuation.
- TIMER 3, PWM used for fan speed actuation.
- TIMER 6, used to periodically refresh the user interface.
- I2C3, communications with temperature sensors.

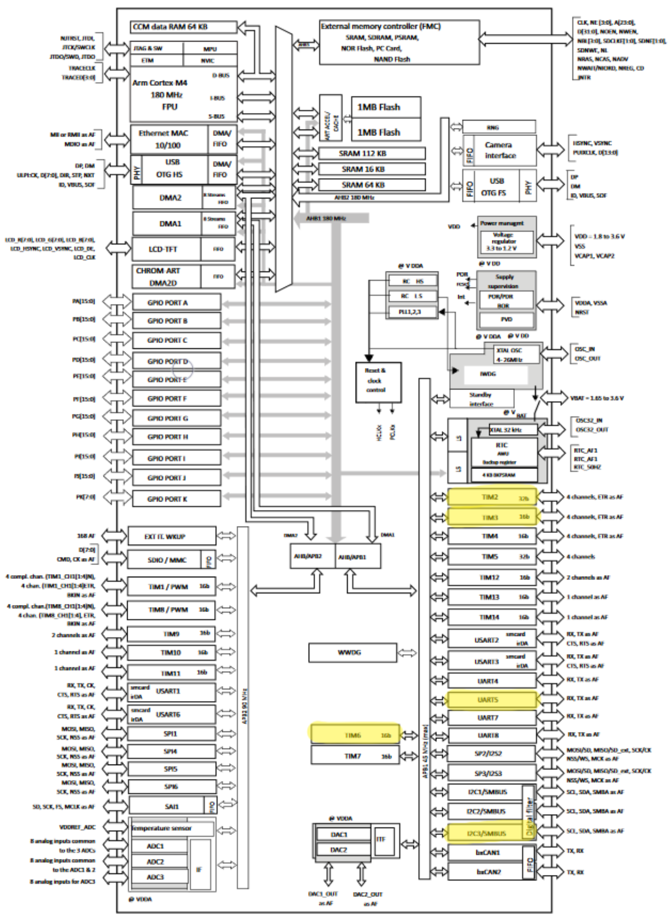

Fig. 4. Solution level architecture with all various functional blocks and peripherals. The highlighted are the used ones.

The overall software is based on three routines:

- Main routine.
- TIMER 6 interrupt routine.
- UART5 receive interrupt routine.

The flow of each routine is reported in figure 5.

The main routine initializes all peripherals and enters in an infinite loop. Here the software reads all the 4 sensors connected on I2C3 and computes the average temperature. The PID control is then applied considering the actual temperature and the projected value inserted by the user. The actuations are then updated. The TIMER 6 is initialized with an APB frequency of 6,25 MHz. In order to have a good refresh rate the prescaler register was set to 12500. This way a clock tick of the TIMER 6 is equal to 2ms. The refresh rate has been empirically chosen to 100ms, so the Timer count has been set to 250. Connecting the TIMER 6 on NVIC the routine is executed every 100ms stopping the normal flow of the program. The timer interrupt routine simply prints on the UART5 a specific string depending on the current value of a variable, called "screen". Input of the user are handled by the UART interrupt routine, here according to screen value the variables are updated.

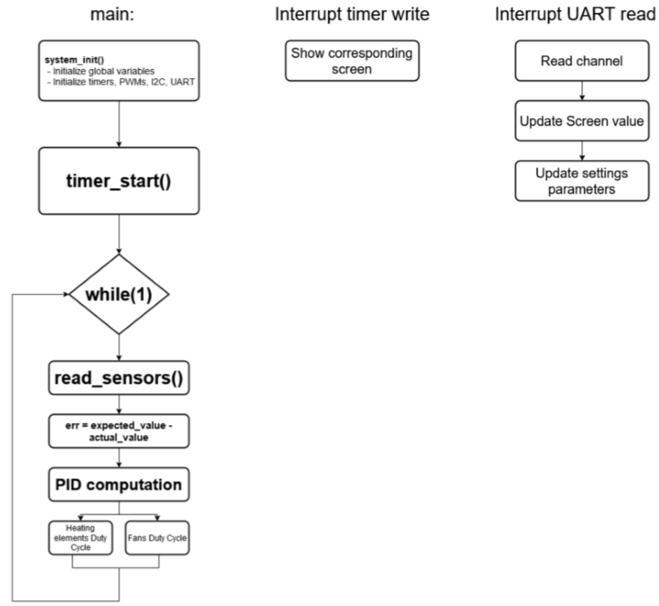

Fig. 5. Program flowchart

IV. Risks assessment

In the Specification phase the PIO platform was thought to be deployed on real hardware, but due to the current situation the system can only be simulated. This introduced some risks in the project that were listed in the Proposal document.

When the design phase was reached, the simulator boundaries were known, and so some criticalities were identified. In particular:

- **Development tools**: instead of the suggested PlatformIO framework the combination of CubeMX and Keil was prefered due to past experiences.
- **PID configuration**: since no real sensors and actuators were used no real data were studied for the selection of the PID parameters.
- **PWM**: there's no way to check whether the output signal of the PWM is the desired one so the firmware just outputs the expected value.

V. Possible applications

The described system could be used in several environment. The main place where the application can find a place is the food industry, where a precise temperature must be maintained for a specific time. Looking over the industries, the system can find place in research application, which require a specific temperature grow. So this application can be used for biomedical application, or either for testing purposes (burn-in tests, endurance test).

VI. Results and discussion

The firmware test has been performed using the Renode simulator coupled with PuTTY. In order to simulate the system in a correct manner the temperature has been simulated using the artificial environments provided from Renode. Thus, is possible to connect the emulated sensors to the related mechanical quantity. In particular four temperature environment has been created and connected to each BMP180 sensor in order to correctly simulate the physical environment. All the inputs and status has been controlled and observed using the terminal. A bottom-up approach has been followed for the overall system test. In particular, the three modules were tested in an independent manner. Then, the test was performed improving the dependency of the software modules. The overall result behaves as expected, implementing the functionalities reported in the specifications.

VII. Conclusions and Recommendations

The proposed platform can fit in many different application field due to its general function. The firmware has been tested by simulations but lacks the hardware test. Is important to underline that all the results achieved are purely digital, the possibilities of misbehaviour mast be kept in consideration when passing to the physical application. Main misbehavior to keep in consideration could be the non-idealities of the acquisition system:

- Conditioning noise
- Aliasing when sampling
- Quantiazation noise

Appendix A
What Goes in the Appendices

Usually not needed The appendix is for material that readers only need to know if they are studying the report in depth. Relevant charts, big tables of data, large maps, graphs, etc. that were part of the research, but would distract the flow of the report should be given in the Appendices.

Appendix B
Formatting the Appendices

Each appendix needs to be given a letter (A, B, C, etc.) and a title. LaTeXwill do the lettering automatically.

References

[1] Renode documentation, https://renode.readthedocs.io/en/latest/

[2] STM32F429-DISCOVERY, https://www.st.com/en/evaluation-tools/32f429idiscovery.html#documentation

[3] Programming manual, https://www.st.com/resource/en/programming_manual/dm00046982-stm32-cortexm4-mcus-and-mpus-programming-manual-stmicro pdf

[4] STM32CubeMX https://www.st.com/resource/en/user_manual/dm00104712-stm32cubemx-for-stm32-configuration-and-initialization-c-code-genera pdf