TECHNICAL AND HARDWARE IMPLEMENTATION OF A MODULAR TMS PROTOTYPE SYSTEM BY USING DATA MODELING, SENSOR COMPUTING AND WARNING SYSTEMS

SEBASTIAN SBIRNA[1], SIMONA SBIRNA[2]

[1]*Ørsted, Dept. of Digital Transformation, Nesa Allé 1, 2820 Gentofte, Denmark, sebas@orsted.com; seby.sbirna@gmail.com*

[2]*University of Craiova, Faculty of Science, Chemistry Department, 107i Bucharest Street, Craiova, Romania, simona.sbirna@edu.ucv.ro; simona.sbirna@gmail.com*

The great necessity of realizing appropriate monitoring of cooling and heating systems seems to be essential in the current context, as we approach the reality of interconnected smart cities and vehicles, which can regulate their energy consumption and, at the same time, maintain high standards on the equilibrium that they provide. This work outlines the essential technical considerations when modeling a working device prototype designed for monitoring of temperature inside vehicles and other areas. The main focus of this paper is to present and explain the algorithm that has been developed throughout three months of intensive prototyping of such a system. The scope of our final prototype includes an automatic alarm function for extreme cases, along with visual feedback in the form of messages on an LCD screen and color-coded LED signals. There is a specific focus regarding the collection and processing of accurate data for vehicles, with the consequence of releasing appropriate warnings to drivers or system owners. The main research focus is on explainability and reproducibility of our decisions behind the programming approach. Finally, we have demonstrated the means through which the mentioned solution is adaptable and cost-efficient, together with short considerations that would take the prototype into mass production.

Keywords: computerized monitoring, production engineering, sensor programming

INTRODUCTION

Owing to the constant expansion of technology and globalization, what was considered as an unreachable dream during the last century is now a common reality: easy delivery of food with refrigerated vehicles, directly to restaurants, shops or even right in front of your door.

Nevertheless, in order for the food to arrive to the customers in the proper conditions and in accordance with the respective regulations, these vehicles need to be properly equipped with temperature monitoring systems (TMS) that alert the clients of the system when there is a change in temperature that requires their attention. This research was conducted within the Department of Engineering and Science from Aalborg University Copenhagen, using helpful advice from Faculty of Science of the University of Craiova – Sbirna and Veng Søberg (2015).

Our research was done using Arduino as prototyping platform, Arduino being an open-source electronic prototyping platform allowing users to create interactive electronic objects (https://www.arduino.cc), which has been received with great interest from the programmers' community, new groups of interest appear very often, and there are a lot of projects available – Shakirovich Ismailov and Botirovich Jo'rayev (2022), Jadav *et al*. (2022), Shaheen *et al*. (2021), Roldán *et al*. (2015), Devika *et al*. (2014), Mellis *et al*. (2007).

The core concepts behind the sensor features will be discussed, along with the implementation of such features into the C++ programming language, custom-configured for Arduino projects – Hughes (2016).

https://doi.org/10.24264/icams-2022.III.17

PRODUCT DESCRIPTION

The system is divided into three main parts, namely: a sensor part, a monitoring part and a warning part. The two output parts will be placed near the driver and the input part will be placed in the storage compartment to measure the temperature. The system is a dependent system because it is optimized for cooling trucks, and the features that it has are designed to be used in such vehicles. It can be used independently in other places, but some of its features might be unnecessary and/or redundant, depending on the application.

The Sensor Part

The sensor part of the system (Figure 1) is where the temperature is measured with three sensors. They work individually and are not dependent on each other. The sensors are meant to be placed in three different locations in the cooling compartment of the truck, one in the front, one in the middle and one in the back. Thus, the sensors are separated from the monitoring and warning part of the system.

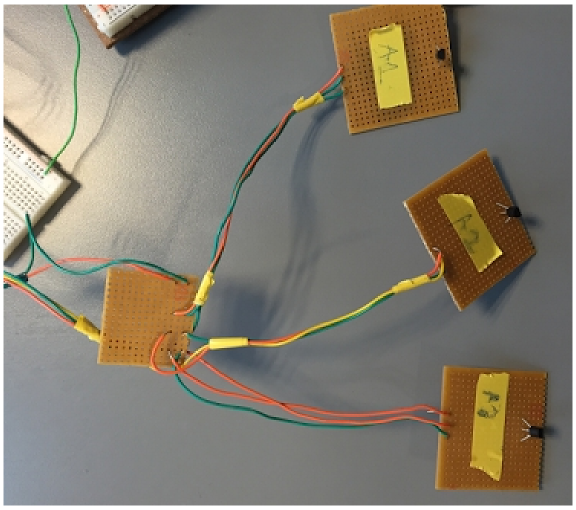

Figure 1. The sensor part of the system

The Monitoring Part

The monitoring part of the system (Figure 2) is its central part, where the current temperature in the cooling compartment is displayed. The wanted temperature range can be adjusted via buttons placed nearby, and, also, pressing a button can stop the sound alarm. If the alarm is stopped, it will start again after some time if the temperature is still outside the acceptable range, and after being stopped two times while the temperature is still outside the acceptable range, it will not be possible to be stopped a third time before it is adjusted correctly.

The display shows the temperature from each of the three sensors individually and also tells the driver if the temperature is “hot”, “warm”, “good”, “cool” or “cold”.

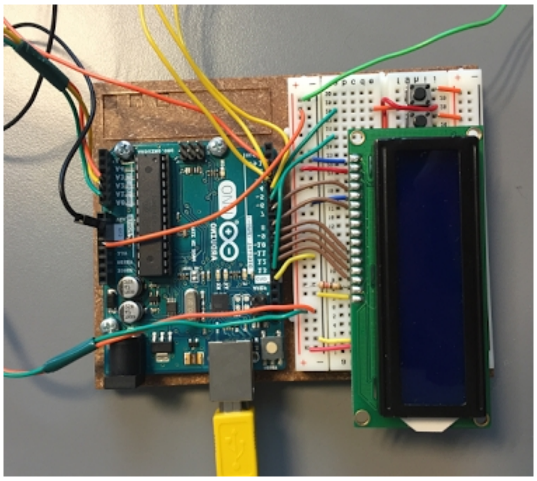

Figure 2. The monitoring part of the system

The Warning Part

The warning part of the system (Figure 3) alerts the driver that something is wrong with the temperature in the cooling compartment of the vehicle, without disturbing the driver more than necessary. The warning board announces the driver both visually and audibly to make sure that the driver notices the warning. It consists of a small speaker (Piezo) and three LEDs. They indicate if the temperature is high, good or low. The red LED is lighting when the temperature goes above a superior given temperature limit, the blue one when the temperature goes beneath an inferior given temperature limit and the green one when the temperature is within the limit range. The speaker informs by audio signals whenever the temperature is reaching an extreme level ("hot" or "cold").

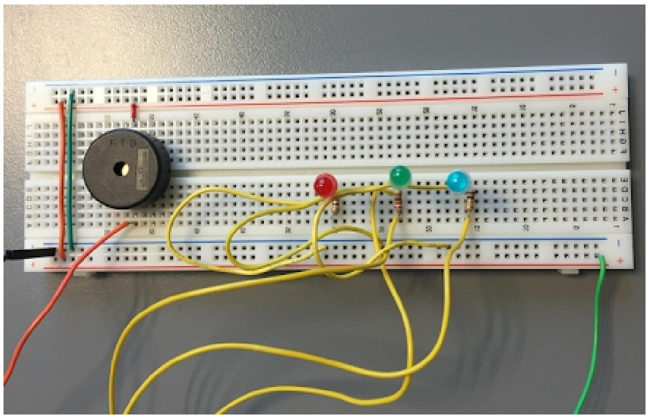

Figure 3. The warning part of the system

https://doi.org/10.24264/icams-2022.III.17

TEMPERATURE RANGES

The main goal of the sensor is to provide both temperature monitoring and warning functionalities to the clients, so one of our first questions in the development process was: "when will the sensor show that the temperature is good, and what will it do otherwise?"

We decided to set the initial "goodTemperature" at 5.0°C, with a no-warning range (a "good" range of values) between ± 0.9°C of the "goodTemperature" ("good" state); a mild-warning range between 1.00 and 1.9°C above or below the "goodTemperature" ("warm" or "cool" state); and a strong-warning range from 2.0°C above or below "goodTemperature" onwards ("hot" or "cold" state).

These ranges of values will be extremely important for our further computations within the sensor code, for they constitute a means to analyzing, giving meaning and working easier with the raw sensor values.

CODE SUBROUTINES

Each individual subroutine of the code for this sensor has to be presented, along with its role in the greater whole of the program – Pan and Makinwa (2022). The functions have been ordered in accordance to their order in the execution flow and will be presented in what follows.

• Subroutine no. 1 – void setup()

The first function to be executed, setup() mainly takes care of the pin mode configurations and displays a welcome message for the client on the LCD, after which it shows the hard-coded "goodTemperature" on the screen and lets the user decide if this temperature value needs adjustment or not.

• Subroutine no. 2 – void setGoodTemperature()

This function will take care of appropriately updating the "goodTemperature" according to the needs of the clients, through the pressing of two buttons (left for decrease, right for increase) which increase or decrease the value by 1.0°C each time one of the buttons is pressed (the buttons can also be long-pressed for a continuous increase or for a continuous decrease of the "goodTemperature" value, until the button is released).

• Subroutine no. 3 – void setGoodTemperatureLCD()

This function updates the value of "goodTemperature" on the LCD screen every time a button is pressed and from setGoodTemperature().

• Subroutine no. 4 – void loop()

This is the function which will be executed over and over again by the Arduino, as soon as setup() and its relevant subfunctions end, and until the program is stopped or reset, so all functions which will be called from loop() or from its subfunctions will also be executed endlessly, allowing us to do most of our mathematical calculations within this function; while the translation of the results into something more easy to work with, as well as the update of the warning devices, will be done in subfunctions of loop().

• Subroutine no. 5 – int checkTemperatureState(float temperature)

The function checkTemperatureState() is the one which assigns an integer number between -2 and 2 (hence, a temperature state) to each of the measured temperature values. The states are saved in a vector named, for good practice, "temperatureState" – Pan and Makinwa (2022) and only after these assignments will the function setState() be executed.

• Subroutine no. 6 – void setState(int i, int currentState)

This function is part of the program because of a problem that needed to be solved, the warning sound being supposed to turn on just after one or more thermosensors have measured a “hot” or “cold” value, and is supposed to turn off only after no thermosensors have measured a “hot” or “cold” value in the last cycle (Figure 4).

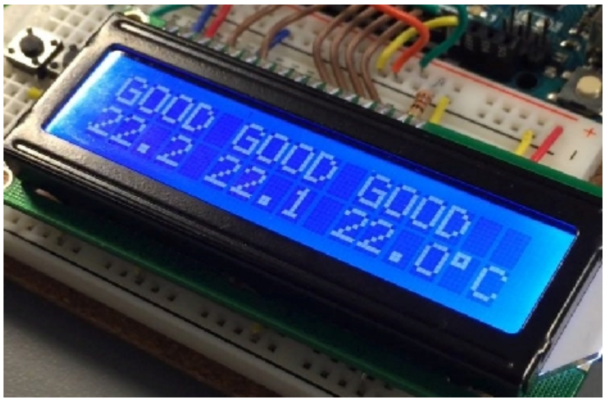

Figure 4. Displaying temperature on the monitoring part of the system

• Subroutine no. 7 – void updateLCD()

This function first calls updateLCDMessages() and then updates the second (bottom) row of the LCD with the current temperatures read from each of the three sensors. Because of size limitations, the sign “°C” is only shown once on the LCD, at the bottom right corner, to inform the user that the temperatures are in degrees Celsius (Figure 4).

• Subroutine no. 8 – void updateLCDMessages(int tmpState)

This function takes care of updating the first (top) row of the LCD with the temperature states corresponding to the values read from each of the three sensors. The states will be shown in capital letters, with one whitespace in between each of the states.

• Subroutine no. 9 – void updateLEDs()

This function updates the LEDs of the prototype each time it is executed, by first turning off all the LEDs and then turning back on only those that correspond to the currently registered temperature states. Because the time during which all the LEDs are off is extremely short (one or a few milliseconds), the eye cannot notice this change.

• Subroutine no. 10 – void updatePiezo()

By long-pressing both buttons of the equipment, the user may switch off the purposefully-done annoying sound of the Piezo. However, due to forgetfulness or lack of responsibility, a user may simply want to turn off the sound warning as soon as it starts, without also taking the necessary action of adjusting the temperature accordingly.

• Subroutine no. 11 – void piezoWarning()

This function may or may not be executed, depending on some criteria, and is responsible for: switching the sound warning on or off; deciding how high a pitch is given as output; as well as resetting the remaining number of times that a user can switch off the alarm, the alarm sound starting the first time a “cold” or “hot” temperature state is detected, and will stop completely only when all the calculated temperature states are neither “hot” nor “cold”.

CONCLUSIONS

The paper has presented most of the programming paradigms behind our chosen approach to achieve certain functionality by using specific components. Through many different incremental updates and improvements across the allowed timespan of our research, a reliable, consistent and easy-to-use temperature monitoring system has been developed. A great advantage of our system is that it is designed to be modular, so that if, in the future, users would need to add extra monitoring services, or perhaps adjust its appearance, it would be possible with very little obstructing factors, as while it is still in its prototype phase, it could easily be implemented by solving the necessary manufacturing hindrance factors (such as replacing connections through breadboards with soldered components or an open-circuited design of the system with a laser-cut acrylic glass casing).

An addition to the program of the system that would be very representative for our product would be the implementation of wireless communication functionality between different modules of the system, so each would gain a lower level of interdependency between components, the risk of errors would be minimized and our product would blend into most of the refrigerated vehicles available, regardless of their internal design.

It is worth mentioning the fact that, in the final product version of the program, new libraries and functionality might be implemented for many different areas.

Some of the improvements could be: data logging, wireless communication between sensors and micro-controller, more warning modules and/or states, the addition of moisture reading and warning, the ability of clients to change the mild and strong warning threshold values, showing of the readings and warning states in Fahrenheit grades and many more.

These changes can only be taken into account if the cost of manufacturing and services will increase by a reasonable amount, since one of the main reasons that a client should choose our solution over the competition’s is the low cost and high reliability of the product and of its services.

REFERENCES

Devika, S.V., Khamuruddeen, Sk., Khamurunnisa, Sk., Thota, J. and Shaik, K. (2014), “Arduino Based Automatic Plant Watering System”, *International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE)*, 4(10), pp. 449-456.

Hughes, J.M. (2016), *Arduino in a Nutshell: A Desktop Quick Reference*, O'Reilly Media, Inc., Sebastopol, California, USA.

Jadav, V., Machhi, N., Biawat, D., Das, R. and Mali, J. (2022), “Review for Arduino Based Portable Ventilator for COVID-19”, *International Journal of Creative Research Thoughts (IJCRT)*, 10(1), pp. d214-d217.

Mellis, D.A., Banzi, M., Cuartielles, D. and Igoe, T. (2007), “Arduino: An Open Electronics Prototyping Platform”, *Proc. Chi*, pp. 1-11.

Pan, S. and Makinwa, K.A.A. (2022), *Introduction. In: Resistor-based Temperature Sensors in CMOS Technology. Analog Circuits and Signal Processing*, Springer Nature Switzerland AG, Cham, https://doi.org/10.1007/978-3-030-95284-6.

Roldán, J.J., Joossen, G., Sanz, D., del Cerro, J. and Barrientos, A. (2015), “Mini-UAV Based Sensory System for Measuring Environmental Variables in Greenhouses”, *Sensors*, 15, pp. 3334-3350, https://doi.org/10.3390/s150203334.

Sbirna, S. and Veng Søberg, P. (2015), “P1 Project: MOE 1, Group 4, Semester Project”, Aalborg University, Copenhagen, Denmark.

Shaheen, Y.S.A., Alkafrawi, H.M.I., Al Aga, T.R.S., Elkafrawi, I.M. and Omar Imaeeg, M.A. (2021), “Arduino Mega Based Smart Traffic Control System”, *Asian Journal of Advanced Research and Reports*, 15(12), pp. 43-52, https://doi.org/10.9734/ajarr/2021/v15i1230449.

Shakirovich Ismailov, A. and Botirovich Jo‘rayev, Z. (2022), “Study of Arduino Microcontroller Board”, *Science and Education Scientific Journal*, 3(3), pp. 172-179.