ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: AUTOMATION, COMPUTERS, ELECTRONICS AND MECHATRONICS
Vol. 15 (42), No. 1, 2018 ISSN 1841-0626
***Note*:** The **"Automation, Computers, Electronics and Mechatronics Series"** emerged from **"Electrical Engineering Series" (ISSN 1223-530X)** in 2004.

Address for correspondence: Liana STĂNESCU
University of Craiova, Faculty of Automation, Computers and Electronics
Al.I. Cuza Street, No. 13, RO-200585, Craiova, Romania
Phone: +40-251-438198, Fax: +40-251-438198
Email: stanescu@software.ucv.ro

CONTENTS

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol. 16 (43), No. 1, 2019

Chatbots in E-learning: An Overview of the Literature

Stefania-Carmen Dobre
Computers and Information Technology Department, University of Craiova, Romania
cdobre@software.ucv.ro

Abstract :

A lot of the technology that we are using today involves Artificial Intelligence: there are also AI chatbots for human-computer interactions that can offer learning support, help students and improve learning. Chatbots have potential all along the learning journey, in many of the learning activities. They can be considered as a virtual 'teacher' used for reducing the workload of teachers, the trainers from the traditional learning. This paper describes a proposal for an educational chatbot and presents an overview about the AI features used for educational chatbots, an architecture model and a literature review about the chatbots used for improving the learning process.

Keywords: educational chatbot, artificial intelligence, learning

1. INTRODUCTION

A bot is a software that is designed to interact with humans using language-based interfaces to perform some automatic tasks. Chatbots are becoming more and more common nowadays, from personal chatbots, personal assistants (can assist in conducting business, meetings reminder, managing to-do lists), healthcare and business industries to educational chatbots (e.g. Botsify [Khan, A. (2019)]). Due to its flexibility, a chatbot can be considered as a trending system, that has been widely used in various fields. There are several companies that embedded chatbot technology into their system environment: Facebook (has implemented Facebook Messenger with the support of chatbot system), Microsoft (Cortana), Samsung (Bixby), Apple (Siri) and Google (Google Assistance).

In e-learning, chatbots offer a personalized experience for students, provide support to acquire knowledge being available 24/24 hours and enable the students to get everything instantly. They are very helpful for students' learning activities. Chatbots are an innovative approach to automate user personalization messages (Akma et al. (2018)). The purpose of this paper is to present an overview of the literature about the chatbots used in education, how AI can improve the learning process and a proposal for a chatbot for education, a smart student assistant. Our proposal for an AI chatbot comes to improve student interaction and collaboration and to act as a virtual teacher assistant in the innovative ed-tech world.

The rest of the paper is structured as follows: section 2 presents an overview of AI features used for educational chatbots, section 3 describes the general features of the chatbot system, an architecture model, few of the methods applied for chatbots development, section 4 includes some examples of chatbots used in education, section 5 contains few details about the proposed chatbot and section 6 highlights some conclusions.

2. AI AND CHATBOTS IN EDUCATION

Chatbots are often called artificially intelligent conversational tools. They are acting as a game changer in the innovative ed-tech world and are built to improve student interaction and collaboration (Singh (2018)).

In the education field, AI is present right across the learning journey (Clark (2018)) to support the students to achieve their learning goals. Chatbots can play a useful role for educational purposes, because they are an interactive mechanism for learning, compared to traditional learning systems as Kowalski et al. (2011) considered.

AI Chatbots can simulate a real human conversation, with real-time responses in natural language based on reinforced learning. They either use voice, text messages or both. There are already artificially intelligent conversational tools able to find things, create and curate content, allow natural language input and output, deliver personalized answers to questions, to enable online assessment and to have adaptive learning features for students' needs.

A chatbot is a useful tool used in education because it can have a lot of features to help teachers to deliver the learning content in a pleasant way or to provide information about the students to adapt the teaching

activity (for example a chatbot to answer questions will help the teacher to see what questions students ask, where students have problems, it can identify the student's weaknesses or the learning path). For students, the interaction with chatbots will bring them a lot of benefits: can offer them needed information, receive instantly answers to their questions, support them with learning/administrative topics and last, but not least, to offer a personalized learning experience.

In the last years, chatbots have started to be used in education due to the benefits they bring to both students and teachers. In what follows, there will be presented some of the ways that chatbots and artificial intelligence are influencing the education (Singh (2018), McElvaney (2018), Spilka (2017)).

From these ways can be mentioned:

Learning Through Chatbot: Artificial intelligence technology used for building chatbots can be used to teach the students a lecture by turning it in a series of messages to make it easier to be read and to look like a standardized chat conversation. The bot will present the next part of the lecture when the student will understand the concept. Unlike the traditional learning method, where a teacher cannot track the level of understanding of the lesson for each of the students present in the classroom, the chatbot can offer this. It will lead to a higher percentage of assimilation of the information presented and implicitly, to achieve the student's learning goals.

Enhanced Student Engagement: The instant messaging platforms and social media tools are used daily by nowadays students. So, the best ways to communicate with peers or to find information about school topics are these platforms or a virtual assistant that can easily provide to the students the needed information about the assignments, due dates or any other events. Thus, it can lead to increase the engagement of students in a subject and can enhance the learning process.

Use Bots as Trainers: Students need continuous support for learning, which makes necessary a resource that can answer any question as quickly as possible.

AI chatbots, are the best option, in this case, because, chatbots can be trained and they will continually learn from students' questions to enrich their knowledge database and to have answers to more and more questions.

Artificial intelligence chatbot can reinforce by learning while doing.

Efficient Teaching Assistant: In a modern learning environment, the repetitive tasks of the teachers can be replaced with a virtual teaching assistant. It should be able to answer the students' questions about the courses, lesson plans, assignments and deadlines. Chatbots deliver instant access to expert knowledge and advice all the time. Also, the bot needs to be knowledgeable enough to provide feedback to students and to analyse the students learning path and to recommend the learning content to them accordingly.

Instant Help to Students: Technology has enabled the students to get everything instantly. Sometimes, a lot of students need the same information, which can be a very time-consuming task for teachers. Chatbots can be used to convert this time-consuming task of replying to each query personally into an automatic one. This will save a great deal of time of the students and they do not need to wait to receive the information, they will instantly receive the chatbot response.

Concentrate on the Learner: To facilitate learning and to increase student engagement, take ownership of their activities, it is useful to use a chatbot. Students will access the information and the learning content via a chatbot and will decide which topic to cover in a learning session. They will perceive this activity as having a personal teacher, who will provide support for the desired lesson/subject.

Smart Feedbacks: In education, feedback is playing an important role: on one hand, students' feedback offers information about their gaps and thus, the teachers can improve their teaching activity; on the other hand, the teachers' feedback allows the students to identify the areas where they need to do some extra work. A smart option to provide feedback, is to use a chatbot, where students can ask several questions, and the feedback will be automatically sent to the teacher, to be analysed and to change/improve their activities accordingly.

Better Student Support: Chatbots can offer a huge value to the educational institution if they are used to keep students informed about faculty facilities, if students have access to all the necessary information about the courses or the modules. Chatbots can also act as campus guides and help the students as they arrive at the campus.

This can increase institutions trust, because it seems that they really care about the students and facilitate their access to the information.

These bots can collect a lot of data about students' perception about school/faculty services and thus the services can be improved.

Personalized Learning: One of the main benefits of e-learning is that the course can be structured to cater for learners with different needs and abilities. Sometimes, some students need to learn deep some modules, but others have other gaps and need an adaptive learning environment. A chatbot can build a comprehensive picture of the learner as well as devising a highly personalized learning pathway. Thus, they will continue to monitor the progress of the learner and can provide the course information only when it is needed.

3. CHATBOT FEATURES

3.1 Chatbot architecture

A chatbot is created to perform some tasks. Regardless of the responsibilities that the chatbot must have, the architecture should contain the following components marked in the figure 1 as component blocks. The chatbot will always communicate with a user and will perform the activities requested by him/her.

The overall conceptual architecture of chatbots is shown in Figure 1.

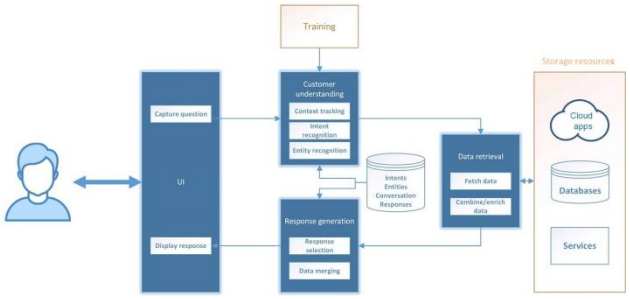

Fig.1 Chatbot architecture (De Scheerder (2018))

In fig.1, there can be identified the following functional components:

User interface (UI): The chatbot should have a responsive UI, good looking and attractive. It is the only visible component for the end-user, the way that the user interacts with chatbot functionalities.

User understanding: This block is responsible for few tasks: it tries to identify the intents (what the user is trying to do), to extract entities (the subjects that the user is talking about; these provide more information about the intents), it uses the conversation context to track what has been said. The bot should be able to learn from these actions. It is trained to match intents and entities and to acquire knowledge to become even better over time.

Data retrieval: If the bot understands what users want, it must find and retrieve the correct data in the storage resources. The resources can be retrieved using web-services or database calls or can be reached out from cloud.

Response generation: The chatbot must select from several good alternatives for responses to provide them to the user. The response(s) will be displayed to the user.

Conversation and context: This component can be programmed with conversation flows (the 'user understanding' module responsible for matching intents) matching the purpose of the bot. This way, it can track and update context and provide a meaningful and natural way to engage in a conversation with the user. Without this component, the bot would revert to zero-state after each question (De Scheerder (2018)).

3.2 Work Methods of Chatbots

3.2.1 Pattern Matches

Chatbots use pattern matching to classify the text and to give a suitable response to the users. For these patterns "Artificial Intelligence Markup Language" (AIML) is used as a standard structure model (Elupula (2019)).

Example for pattern matching:

<category>

<pattern>Who old are you </pattern> <template>I am 25</template>

</category>

<category>

<pattern>what are you called</pattern>

<template>

<srai>what is your name</srai>

</template>

</category>

The chatbot will give the answer for the user question because the pattern contains a part of the question. It will react only to anything related with the correlate patterns. But it can't go past the related pattern. To go beyond the associated pattern, there are used algorithms. The algorithms are utilized for text classification and can produce several pattern combinations to make a hierarchical structure.

3.2.2 Natural Language Understanding (NLU)

NLU (Natural Language Understanding) is an artificial intelligence technique used for matching the sentences from the users into intents, based on the intelligence behind the system, a supervised intent classification model created on a range of sentences as input and intentions as target. NLU has 3 specific concepts which will be described below.

Entities are represented by the chatbot purpose (e.g. student assistant chatbot, language learning chatbot).

Intents are the chatbot actions performed when the user says something. The bot should identify the user's intentions, to extract the main idea from these, because it is possible that users ask for the same thing, but with different inputs.

Context is a way for mapping user questions to intents, without being necessary to store conversation/questions history, each of the question will have assigned a flag (e.g. if a question is: "What's time in Romania", the flag can be 'Romanian time').

3.2.3. Natural Language Processing (NLP)

Natural Language Processing Chatbots are smart enough to find a way to convert the user's speech or text into structured data for choosing a relevant answer.

NLP can translate human language into information that contains text as well as patterns that can be useful in discovering applicable responses. Usually, the NLP chatbots are connected to a database, which is used to

sustain the chatbot for providing the appropriate responses to every user.

3.3 Elements of a chatbot

Chatbots must have the following essential components to be a conversation partner (Nieves, 2018):

- *Conversational artificial intelligence*, the basic source of chatbots, thanks to which all management and natural language processing (NLP) occur. The first chatbots focused on the interpretation and recognition of patterns and rules. The more advanced chatbots implement deep learning processes to analyze the human input, learn from conversations and generate as suitable a response as possible.
- *User experience* (UX), which allows a natural, intelligent and coherent conversation to be established.
- *User interface* (UI), whereby the user can see or hear the conversations with the chatbot.
- *Conversational design*, which allows an artificial interaction to be equipped with human logic.

4. CHATBOTS FOR IMPROVING THE LEARNING PROCESS

Depending on their nature, there can be distinguished two types of chatbots in education: with and without an educational intentionality (Garcia Brustenga et al. (2018)). Chatbots without educational intentionality are responsible for teaching tasks of an administrative nature (student guidance and personal assistance) and of a support nature (to answer FAQs). From the other category, are mentioned chatbots that are designed to foster teaching and learning directly: tutors, that provide support for the learning process (can adapt, select and sequence contents according to the student's needs and provide learning motivation), and exercises or practice programs for skills acquisition (chatbot gives immediate feedback to the student when the student answers questions).

Lately, the need for the use of chatbots in education implies their development according to the new technologies that respond better to the needs of students and teachers. In the following, some examples of chatbots used for educational purposes will be presented.

Otto chatbot was developed by Learning Pool company (Clark (2018)). It aims to enhance the student-content interaction and it is integrated in an LMS.

Ani (Garcia Brustenga et al. (2018)) is an educational chatbot that was designed for learning and to replace some tasks of teachers. The goal was to provide personalized tutoring and mentoring that students become more involved in the learning activity. It includes the ability to adapt to the user's needs using automatic learning algorithms, as well as elements of assessment, motivation and immediate feedback.

Duolingo (Sawers (2019)) is designed for language learning (many of the world's most common languages) using conversation through gamification techniques.

Botter (Garcia Brustenga et al. (2018)) is a physical robot able to provide student support, it works as cognitive technology for learning, for the promotion of student behavioural change. It can interact with students through few ways: using light signals, movements, sound messages for motivation and disappointment or to help students to monitor their learning progress.

Differ (McNeal (2016)) is a chat application for higher education. The aim is to create a space for students where they can ask everything. Differ can have communities that bring together students in similar situations, it publishes relevant messages, reminders for increasing commitment and involvement.

CourseQ (Garcia Brustenga et al. (2018)) was designed at Cornell University (USA), to help the students, college groups and teachers by providing them an easy platform to communicate. The chatbot's functions include obtaining information for faculty and students as well as giving reminders regarding submission dates, timetables, material and events.

Botsify (Khan, A. (2019) is an education chatbot that presents a specific topic to the students and after learning the topic, students take quizzes and submit the results to their teachers. It is used, mainly, to help teachers to easily monitor the students' performances.

Ivy (Garcia Brustenga et al. (2018)) is an artificially intelligent self-service chatbot, designed for colleges and universities. It enables financial services, career services, management of admissions, technological services such as email access, Wi-Fi connection and app installation.

Di Blas et al. (2019 a, b) propose iMOOC and iCHAT, two chatbots created as an innovative approach to adaptive learning and for the using of the conversational interfaces in education. They implemented adaptive learning via conversational empathic interfaces, to help the learner deal with complex materials.

Unsupervised machine learning techniques were used by Ndukwe et al. (2019) for the task of automated grading chatbot, able to ask students questions and require written responses. In contrast with this approach, a different communication from the questionnaire procedure, Vladova et al. (2019) proposed a solution, a chatbot that acts as a teacher with natural language capabilities and used an informal way of obtaining the information about students through the chat addresses.

5. TOWARDS AN INTELLIGENT CHATBOT FOR EDUCATION

Due to continuous improvements in technology, the educational field should also be up-to-date with current trends. That is why chatbots (educational virtual assistants) are used more and more to perform automatic tasks for teachers and to support the learners to acquire knowledge.

Chatbots are considered a "way to improve the learning process, by helping the learner to deal with a complex material, tailoring the learning experience to specific needs" (Di Blas et al. (2019 a)).

An educational chatbot should be powerful enough to provide immediate and customized instruction or feedback to the learner, a unique, highly focus learning path and individual learning program for each student based on data gathered throughout the training process.

A combination of different AI techniques such as Natural Language Processing, Machine Learning and Semantics Understanding can drive the desired results for the needed requirements. From the above, we can consider that an AI chatbot for educational environment is an intelligent tutoring system for students, that has adaptive and personalization capabilities. There are already many chatbots used in education with AI features (Di Blas et al. (2019 a, b), Vladova et al. (2019), Ndukwe et al. (2019)) created to assist the students.

In the following, will be presented an intelligent chatbot for education, some features and problems that will be solved by our system proposal.

Our AI chatbot proposal will include features from two main areas: Intelligent Tutoring Systems (ITSs), the way in which AI techniques will be applied to education and adaptive and personalized learning, the way in which AI techniques will identify what the learners' needs and how will be improved the learning experience.

The chatbot will be a "virtual teacher", to support one-to-one tutoring, with natural language understanding and natural language processing capabilities. The advantage of this chatbot is that the communication/responses in real time will be through text or voice as it is in the traditional classroom or in case of face-to-face training.

By using AI techniques, the system will be able to interpret complex language and to extract the intents from the questions, will reply with natural responses and due to that, the learner needs to have a learning progress, the bot must remember the context of the entire conversation. From the informal conversation with the students and from answers, the bot will know the starting point for learning a topic and will identify the cognitive students' skills for organizing the course information that will be provided to the learner. An important role in the learning process is represented by the assessments and the feedback related with the acquired knowledge. To complement the grading system mentioned in Ndukwe et al. (2019)), where short answer text provided by a student are matched with at least one correct answer, machine learning techniques for text processing will be used when the students answer will be a paragraph or a document. For collecting data used for personalization, all students'answers will be stored to get an overview about the learning style, how the learning content should be provided (text, video) and how the student can manage routine tasks easily (e.g. reading the course information and taking a quiz to check for knowledge).

6. CONCLUSIONS

In this paper, we have presented a literature review about the chatbots used in education, an overview of AI features used for educational chatbots, general features for chatbots and few details about a chatbot proposal.

The next step is to design and implement this chatbot and to conduct an experimental evaluation about the chatbot usage for learning.

REFERENCES

Akma, N., Hafiz, M., Zainal, A., Fairuz, M. and Adnan, Z. (2018). Review of Chatbots Design Techniques. *International Journal of Computer Applications*, 181, 7-10.

Anteunis, J. (2017). How to Build A Powerful Enterprise Chatbot? Retrieved from: https://cai.tools.sap/blog/enterprise-chatbot-methodology/. (last accessed on 1 august 2019)

Clark, D. (2018). The Fallacy of "Robot" Teachers. Donald Clark Plan B. Retrieved from: https://donaldclarkplanb.blogspot.com/search?q=10+uses+for+Chatbots+in+learning+(with+examples).

De Scheerder, M. (2018). Building a chatbot: a reference architecture. (2018). Retrieved from: https://www.loqutus.com/building-a-chatbot-a-reference-architecture/

Di Blas, N., Lodi, L., Paolini, P., Pernici, B., Raspa, N., Rooein, D. and Renzi, F. (2019 a). Sustainable Chatbots supporting Learning. *Proceedings of EdMedia + Innovate Learning*, 1376-1381

Di Blas, N., Lodi, L., Paolini, P., Pernici, B., Renzi, F. and Rooein, D. (2019 b). Data Driven Chatbots: A New Approach to Conversational Applications. Retrieved from http://ceur-ws.org/Vol-2400/paper-24.pdf

Elupula, V. (2019). How do chatbots work? An overview of the architecture of chatbot. Retrieved from: https://bigdata-madesimple.com/how-do-chatbots-work-an-overview-of-the-architecture-of-a-chatbot/. (last accessed on 1 august 2019)

Fryer, L., Nakao, K. and Andrew, T. (2019). Chatbot learning partners: Connecting learning experiences, interest and competence. *Computers in Human Behavior*, 93, 279-289.

Garcia Brustenga, G., Fuertes-Alpiste, M., Molas-Castells, N. (2018). Briefing paper: chatbots in education. *Barcelona: eLearn Center. Universitat Oberta de Catalunya*. Retrieved from: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/80185/6/BRIEFING_PAPER_CHATBOTS_EN.pdf

Khan, A. (2019). How Education Industry Is Being Improved by AI Chatbots? Retrieved from: https://medium.com/botsify/how-is-education-industry-being-improved-by-ai-chatbots-4a1be093cdae. (last accessed on 1 august 2019)

Kowalski, S., Hoffmann, R., Jain, R., Mumtaz, M. (2011). Using Conversational Agents to Help Teach Information Security Risk Analysis. *SOTICS 2011:*

The First International Conference on Social Eco-Informatics, pp 91-94. Retrieved from: https://www.thinkmind.org/download.php?articleid=sotics_2011_4_30_30106

McElvaney, P. (2018). 10 Secrets to Transforming L&D with A Chatbot. (2018). Retrieved from: https://elearningindustry.com/transforming-l-d-with-a-chatbot-10-secrets (last accessed on 1 august 2019)

McNeal, M. (2016). A Siri for Higher Ed Aims to Boost Student Engagement. https://www.edsurge.com/news/2016-12-07-a-siri-for-higher-ed-aims-to-boost-student-engagement. (last accessed on 1 august 2019)

Meyer von Wolff, R., Hobert, S., Schumann, M. (2019). How May I Help You? - State of the Art and Open Research Questions for Chatbots at the Digital Workplace. *Proceedings of the 52nd Hawaii International Conference on System Sciences 2019*.

Ndukwe, I.G., Daniel, B.K., Amadi, C.E. (2019). A Machine Learning Grading System Using Chatbots. *Artificial Intelligence in Education. AIED 2019. Lecture Notes in Computer Science*, volume 11626, pp 365-368.

Roos, S. (2018). Chatbots in education: A passing trend or a valuable pedagogical tool? Retrieved from: https://pdfs.semanticscholar.org/533e/bc0255c36749e1f6b8d3662464d6ee5d4f0.pdf

Sawers, P. (2019). How Duolingo is using AI to humanize virtual language lessons. Retrieved from: https://venturebeat.com/2019/07/05/how-duolingo-is-using-ai-to-humanize-virtual-language-lessons/. (last accessed on 1 august 2019)

Singh, R. (2018). AI and Chatbots in Education: What Does the Future Hold? (2018). Retrieved form: https://chatbotsmagazine.com/ai-and-chatbots-in-education-what-does-the-futurehold-9772f5c13960 (last accessed on 1 august 2019)

Sjöström, J., Aghaee, N., Dahlin, M., Ågerfalk, P. (2018). Designing Chatbots for Higher Education Practice. *International Conference on Information Systems Education and Research*, Proceedings 4

Spilka, D. (2017). 4 Ways for Using Chatbots for eLearning. Retrieved from: https://elearningindustry.com/chatbots-for-elearning-4-ways-using. (last accessed on 1 august 2019)

Vladova, G., Haase, J., Rüdian, L.S., Pinkwart, N. (2019). Educational Chatbot with Learning Avatar for Personalization. Retrieved from: https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1352&context=amcis2019

Winkler, R. and Söllner, M. (2018). Unleashing the Potential of Chatbots in Education: A State-Of-The-Art Analysis. *78th annual meeting of the academy of management*, at Chicago

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol. 16 (43), No. 1, 2019

Rust, a Memory-safe Alternative to C

Mădălin Mămuleanu*

**Automation and Electronics Department, University of Craiova, Craiova, Romania*

(e-mail: mamuleanu.madalin@gmail.com)

Abstract: Applications written in unsafe languages like C or C++ can introduce serious security issues due to memory errors such as buffer overflows, dangling pointers or reads of uninitialized data. In this paper, we are analyzing the most common security vulnerabilities found in software applications written in C or C++ and whether these issues could be avoided by using Rust, a modern programming language focused on safety.

Keywords: software security, memory safety, undefined behavior, embedded systems, C programming, Rust

1. INTRODUCTION

One of the oldest problems in computer security is memory corruption. Software written in C or C++ is prone to these type of issues. The lack of memory safety in these languages offers hackers the possibility to exploit memory bugs and alter the software's behavior. These bugs can be very hard to reproduce, debug and potentially expensive to correct. This war in memory safety is fought on two sides. One side is trying to find new ways to exploit software bugs in order to change the application's behaviour. And, on the other side, researchers and programmers are working to develop new protections and to write a safer code.

According to the Common Weakness Enumeration (CWE™) Top 25 Most Dangerous Software Errors, the most dangerous issue found in software is related to memory safety. C programming language facilitates writing high performance code through lightweight abstractions close to the hardware. However, low-level control undermines security. Support for manual memory management and unchecked memory access introduces opportunities for human errors. In some cases, these errors may escalate to catastrophic failures. The C programming language was created to provide proximity to the underlying hardware in order to write low-level code. C has control over the memory layout and minimal runtime support. However, some features of C, like pointers, pointer arithmetic, pointers to middle of objects, unchecked array indexing can cause simple programming error to corrupt the values of arbitrary memory locations. Memory corruption can cause a program to crash either immediately or in a non-deterministic manner, producing wrong results. This can be the root cause of multitude of security bugs. This article is structured as following:

In section 2 the common security bugs related to memory safety are analyzed.

Section 3 describes Rust's approach to memory safety and the code from section 2 is analyzed and section 3 presents conclusions from the presented work.

1. Spatial safety violations

Memory safety violation can be either a spatial safety violation or a temporal safety violation. A spatial safety violation occurs when the code accesses a memory location outside the bounds of the object associated with the pointer or the array. Any pointer that points outside of it's associated object may not be dereferenced. Dereferencing these pointers results in a spatial memory safety error and undefined behavior.

Example of spatial safety violation:

```
//...
char *p = malloc(24);
for(int i=0; i<27;i++)
{
   p[i] = i+0x41;
}
//...
```

2. Temporal safety violations

A temporal safety violation occurs when a pointer is used to access a memory location after the object has been deallocated. When an object is freed, the underlying memory is not longer associated to the object and the pointer is not longer valid. Dereferencing these type of pointers will result in a temporal memory safety error and undefined behavior.

Example of a temporal safety violation:

```
//...
char *p = malloc(24);
free(p);
for(int i=0; i<27;i++)
{
   p[i] = i+0x41;
}
//...
```

2. COMMON SECURITY BUGS

The serious security vulnerabilities facilitated by the lack of memory safety in C and C++ are well known.
This is the underlying root cause of many security vulnerabilities. The memory corruption often allows an attacker to insert malicious code into the system and taking control of the entire system.

2.1 Integer overflow

In the context of programming, an integer is a variable capable of representing a number with no fractional part. Like all variables, integers are regions of memory. When we talk about integers, we usually represent them in decimal because this is the numbering systems that we are used to. Computers are not dealing with decimal. So, internally, an integer is stored in binary. Because it's necessary to store also negative numbers, there is a mechanism to represent them using binary. This is accomplished by using the most significant bit (MSB) to determine the sign. Because an integer is a fixed size, there is a maximum value it can store. When a value greater than the maximum value is written to this type of data, and integer overflow occurs. This overflow causes undefined behavior. An integer overflow cannot be detected after it happened, so there is no way for an application to tell if a result calculated previously is in fact reliable. This can become very dangerously if the calculated value has to do with the size of a buffer or an index of a buffer.
Example of a integer overflow bug in libssh2 library:

```
if(session->userauth_kybd_num_prompts) {
    session->userauth_kybd_prompts =
=LIBSSH2_CALLOC(session,sizeof(LIBSSH2_USER
AUTH_KBDINT_PROMPT) *              session-
>userauth_kybd_num_prompts);
// This may overflow


}
```

The bug fix provided by the libssh2 team:

```
   if(session->userauth_kybd_num_prompts >
100)
{
        _libssh2_error(session,
LIBSSH2_ERROR_OUT_OF_BOUNDARY,"Too many
replies for keyboard-interactive prompts");
     goto cleanup;
}
if(session->userauth_kybd_num_prompts)
{
    session->userauth_kybd_prompts =
LIBSSH2_CALLOC(session,
sizeof(LIBSSH2_USERAUTH_KBDINT_PROMPT) *
session->userauth_kybd_num_prompts); //
This will not overflow
                /* ... */
}
```

2.2 Dangling pointers

In computer programming, dangling pointers are pointers that are pointing to a memory address that has been deleted (or freed). Dangling pointers often arise during object destruction, when a reference of an object has been deleted or deallocated without modifying the value of the pointer, so that the pointer still points to the deallocated address. In some situations, the system may have reallocated the previously freed memory and unpredictable behavior may result because the memory contains completely different data.

Example of a program that creates a dangling pointer:

```
#include<stdio.h>
#include<stdlib.h>


int main()
{
    char **strPtr;
    char *str = strdup("Hello!");
    strPtr = &str;
    free(str);
    //strPtr now becomes a dangling pointer
    printf("%s", *strPtr);
    return 0;
}
```

2.3 Doubly freeing memory

Double free errors occur when free() is called more than once with the same memory address as an argument. Calling free() twice with the same memory address as an argument can lead to undefined behavior and could allow a malicious user to write values in arbitrary memory spaces.

Example of doubly freeing memory:

```
char* ptr = (char*)malloc (SIZE);
...
if (abrt) {
free(ptr);
}
...
free(ptr); // Double free!
```

2.4 Uninitialized Memory Access

This type of memory error will occur when an uninitialized variable is read in an application.

Example of uninitialized memory access:

```
char *p = (char*) malloc(256);
char c = p[0]; //  p was not initialized
void func()
{
    int a;
    int b = a * 7; // uninitialized read of
variable a
}
```

2.5 User-supplied format strings

When a user-supplied value is used as a format string in printf(), scanf() or a related function. Because of this, the attacked could execute code, read the stack and cause a segmentation fault in the running application, compromising the security or the stability of the system. Because printf has a variable number of arguments, it must use the format string to determine the number of arguments. In the code presented below, the attacker can pass the string "%p %p %p %p %p %p %p %p %p %p %p %p %p %p" and trick the printf function into thinking that it has 15 arguments. It will print the next 15 addresses on the stack.

Example of format string attack:

```
#include<stdio.h>
int main(int argc, char** argv)
{
    char buffer[100];
    strncpy(buffer, argv[1], 100);
    // Passing command line argument to
printf
    printf(buffer);
    return 0;
}
```

2.6. Race Condition

A race condition occurs when two or more threads access memory without synchronization and at least one of the is trying to write it. The first thread reads the variable and the second thread reads the same value of the variable. After that, both threads perform their operations on the value, and they race to see which thread can write last to the shared variable.

Example of a race condition bug:

```
#include <functional>
#include <iostream>
#include <thread>

struct Account{
  int balance{100};
};
void transferMoney(int amount, Account&
from, Account& to)

  {
   using namespace std::chrono_literals;
   if (from.balance >= amount){
from.balance -= amount;
std::this_thread::sleep_for(1ns);
to.balance += amount;
   }
}

int main(){
  Account account1;
  Account account2;
  std::thread thr1(transferMoney, 50,
std::ref(account1),std::ref(account2));
 std::thread thr2(transferMoney, 130,
std::ref(account2), std::ref(account1));
thr1.join();
thr2.join();
std::cout << "account1.balance: " <<
account1.balance << std::endl;
std::cout << "account2.balance: " <<
account2.balance << std::endl;
std::cout << std::endl;
}
```

3. THE RUST APPROACH TO SAFETY

Rust is a multi-paradigm system programming language focused on safety, especially safe concurrency. Rust is syntactically similar to C++, but is designed to provide better memory safety while maintaining high performance. [wiki] Rust was developed by Mozilla and it combines the speed and control of a lower level language with the tools, safety, and debugging provided from a high-level language.

3.1 Hello World in Rust

The simple program from below, similar to C, defines a main function that is the designated entry point for the program. The function is defined with the fn keyword followed by an optional set o parameters. This function consists of a call to the println! macro, which sends formatted text to the console.

```
fn main()
{
    println!( "Hello World.");
}
```

3.2 Embedded Rust

Embedded Rust is focused on safety and can replace code written in C. Like C language, Rust can be compiled for many targets including bare metal embedded systems. At this moment, the targets available for Embedded Rust are:

- Bare Cortex-M0, M0+, M1
- Bare Cortex-M4, M7
- Bare Cortex-M4F, M7F, FPU, hardfloat
- Bare Cortex-M3
- x86, x86-64
- ARM
- MIPS, MIPSEL
- RISC-V

3.4 Safety check for cleaner code

Unlike C, the Rust compiler enforces memory safety guarantees and issues like dangling pointers or using an object after it has been freed cannot occur in safe mode. But, in embedded software development it's important to manipulate addresses that could represent hardware registers or memory addresses. For this, Rust includes an unsafe keyword allows the programmer to do potentially memory-unsafe operations. In this section we'll show the Rust equivalent of the examples in section 2 and show what happens if we try to compile and run them.

Dereferencing a raw pointer using *unsafe* keyword:

```
fn main() {
    let a = 1;
    let rawp = &a as *const i32;
    unsafe {
println!("rawp is {}", *rawp); }
}
```

- Spatial safety violation:

```
fn main() {
    let mut vec = vec![0; 24];
    for i in 0..27 {
        vec[i] = i + 0x41;
    }
}
```

Result: **Runtime error**

thread 'main' panicked at 'index out of bounds: the len is 24 but the index is 24'

- Integer overflow:

```
fn main() {

    let a:u32 = 5436445;
    let b:u32 = 5436363;
    println!("{}", a*b);
}
```

Result: **Compilation error**

thread 'main' panicked at 'attempt to multiply with overflow'

- Dangling pointers:

```
fn main() {
    let r;
    {
        let vec = vec![1, 2, 3];
        r = &vec;
    }
    println!("{:?}", r);
}
```

Result: **Compilation error**
borrowed value does not live long enough, `vec` dropped here while still borrowed

- Race condition:

```
use std::thread;
use std::time::Duration;
```

```
struct Account {
    balance: i32,
}

impl Account {
    fn with_balance(balance: i32) -> Self {
        Self { balance }
    }
}

fn transfer_money(amount: i32, from: &mut
Account, to: &mut Account) {
    if from.balance >= amount {
        from.balance -= amount;
thread::sleep(Duration::from_nanos(1));
        to.balance += amount;
    }
}
fn main() {
    let mut account_1 =
Account::with_balance(100);
    let mut account_2 =
Account::with_balance(100);
crossbeam_utils::thread::scope(|s|
{
 s.spawn(|_| transfer_money(50, &mut
account_1, &mut account_2));
 s.spawn(|_| transfer_money(130, &mut
account_2, &mut account_1));
    }).unwrap();
    println!("account_1.balance:        {}",
account_1.balance);
    println!("account_2.balance: {}",
account_2.balance);
}
```

Result: **Compilation error**

cannot borrow `account_1` as mutable more than once at a time

cannot borrow `account_2` as mutable more than once at a time

4. CONCLUSIONS

The problem with memory safety in C and C++ is an old and very discussed topic. Despite the fact that the vulnerabilities introduced by the lack of memory safety in C and C++ are well known, critical memory bugs are found every year in software applications written in C or C++. All the code from section 2 written in C or C++ led to undefined behavior in these programming languages. Running the equivalent version of these examples in Rust led to runtime or compilation errors.
Because Rust is a programming language focused on safety and also because it provides the tools, safety, and debugging from a high-level language, it can be a perfect candidate for the main programming language in an embedded system.

REFERENCES

Laszlo Szekeres Mathias Payer Tao Wei Dawn Song Stony Brook SoK: Eternal War in Memory
Santosh Ganapati Nagarakatte (2012) PRACTICAL LOW-OVERHEAD ENFORCEMENT OF MEMORY SAFETY FOR C PROGRAMS University of Pennsylvania
https://www.autosar.org/fileadmin/user_upload/standards/adaptive/17-03/AUTOSAR_RS_CPP14Guidelines.pdf
https://www.cvedetails.com/cve/CVE-2019-3856/
https://cwe.mitre.org/top25/archive/2019/2019_cwe_top25.html
https://doc.rust-lang.org/book/ch16-00-concurrency.html
https://en.wikipedia.org/wiki/Rust_(programming_language)
https://forge.rust-lang.org/release/platform-support.html
https://www.libssh2.org/
https://nebelwelt.net/teaching/17-527-SoftSec/slides/02-memory_safety.pdf
https://nvd.nist.gov/general/visualizations/vulnerability-visualizations/cwe-over-time
https://securingtomorrow.mcafee.com/mcafee-labs
http://willcrichton.net/notes/rust-memory-safety/

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol. 16 (43), No. 1, 2019

The Transfer of an Artificial Neural Network from a Device to an Embedded Device

Mihai-BebeSimion

**Department of Automatic Control and Electronics, University of Craiova, Romania (e-mail: mihai.bebe.simion@gmail.com).*

Abstract: Designing a controller that can run on different systems is a challenge, due to the fact that each system has its own characteristics: operating system, power supply, clock cycles, etc. Due to an increase of computational power, artificial neural networks have come to aid in the field of control systems theory. An example of simple utilization of ANNs in control design will be proposed in this paper. More precisely, an ANN control approach is designed for a benchmark Quanser platform and the obtained results are compared with those of a simple PI controller. For practical reasons, the designed ANN control scheme is transferred to an embedded device - a dedicated microcontroller. Several simulations and experimental results are presented.

Keywords:Neural Network, Machine Learning, MATLAB - Simulink, Arduino, Quanser benchmark platform.

1. INTRODUCTION

Artificial Neural Networks are computing systems that are inspired by biological neural networks. As with all neural systems, these ANN learn by considering examples. There are different ANN topologies that are used to learn different thing.

The first artificial neural network was the perceptron. The perceptron was developed by Frank Rosenblatt's in 1958. It could learn to associate between a simple input image and a desired output, the output indicated if the object was present in the image. Its topology is made of a single input layer and a single output layer. The information passes from the input layer, then it is sum up, passed through an activation function and send to the output layer (Hetcht-Nielsen, 1989; Hertz et al., 1991; Saerens and Soquet, 1991; Warwick et. al., 1992).After the computation power increased, new types of artificial neural networks were developed: Feed Forward Neural Network, RBF neural networks, DFF neural network, Recurrent Neural Networks, Long / Short Term Memory, Gated Recurrent Unit, Auto Encoders, Deep Belief Network, Deep Convolutional Network, Generative Adversarial Network, etc.

The Feed Forward Neural Networks originates from the 1950(Fig. 1). The topology of this ANN is as follows: one input layer, one hidden layer and one output layer, all layers are fully connected. The information from the input layer passes to the hidden layer, at the hidden layer, the information from each input neuron is multiplied by a weight and it sums them up. The sum then next passes through an activation function and passed to the output. The same happens to the output layer, but instead of the input layer, the information from the hidden layer is used(Hetcht-Nielsen, 1989; Frean, 1990; Carelli et. al., 1995; James, 1995, 1999).

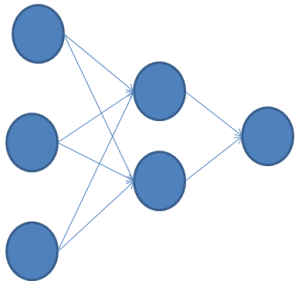

Fig. 1. Feed Forward Neural Networks topology.

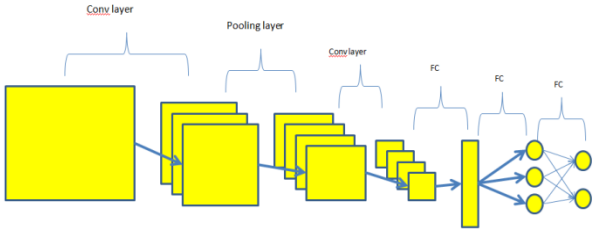

Fig. 2. Convolutional Neural Network topology.

A Convolution Neural Network (Fig. 2) is a class of deep NN, they are also known as space invariant artificial neural networks (SIANN). They are applied in image recognition, medical image analysis and natural language processing (Christopher and Bishop, 1995).

These networks apply a mathematical operation called convolution, hence the name "convolution neural network". The design of a "convolution neural network" consists of an input layer, multiple hidden layers and an

output layer. The hidden layers consist of a series of convolution layers that convolve to a dot product. The activation function is a RELU (rectifier) layer which is followed by other convolution layers (pooling, normalization or fully connected). The final layer involves backpropagation in order to more accurately weight the end product.

A generative adversarial network is a class of machine learning systems that generates training data with the same statistics as the training set. It is composed of two networks, one "generative network" that generates candidates while the "discriminative network" evaluates them (Juergen, 2015).The objective of "generative network" is to increase the error rate of the "discriminative network" (its purpose is to fool the discriminator network by generating new data that the discriminator network thinks it is not synthesized).

In control system theory, artificial neural networks can have a big impact due to their nonlinearity behaviour and the capability to replicate any control law.One example in which an ANN is used in the control system theory is to choose the Kp, Ki and Kd gains of a PID controller for a known installation.

This paper covers the transfer of an ANN which was trained on a device with specific characteristics to embedded device. The ANN was trained in MatLab – Simulink with data from a PI control law that controls a Quanser benchmark platform (SRV02 motor system).

2. PI CONTROL LAW IN MATLAB – SIMULINK

The integral component, expressed by the integration time constant Ti determines an order proportional to the integral of the error of the system in question, a stationary regime is possible only if this error is null. The existence of a such a component I in a control law indicates that the accuracy of the system in stationary regime (if such a regime can be obtained) is infinite. In a stationary regime, most of the time the component I determines the increasing oscillation of the response, reducing the stability of the system[Hassan et. al (2018)].

2.1 Hardware

The Pant is a Quanser Rotary Motion Servo Plant: SRV02 (Fig. 3). It consists of a DC motor that is placed in a solid aluminium frame and equipped with a planetary gearbox. The motor has its own internal gearbox that drives external gears. The SRV02 units are equipped with a potentiometer sensor and an encoder, in order to obtain digital position measurement.

Table 1. Electrical and mechanical characteristics

Symbol	Description	Value
Vnom	Motor nominal input voltage	6.0 V
Rm	Motor armature resistance	2.6 Ω
Lm	Motor armature inductance	0.18 mH
kt	Motor torque constant	7.68E-03N·m
km	Back-emf constant	7.68E-03V/(rad/s)
ηm	Motor efficiency	0.69
Jm rotor	Motor shaft moment of inertia	3.90E-7kg·m^2

Fig. 3. SRV02 system.

Fig. 4. Universal Power Module UPM 1503.

Each SRV02 system requires a power amplification module. The power amplification module used is the Universal Power Module UPM 1503 (Fig. 4). The power module consists of a regulated dual output VDC power supply and a linear power operational amplifier. The power section is powered through a differential power supply with the following characteristics: Maximum power output 45W, Maximum current output 3 A, maximum output voltage 15 V, power bandwidth 60 KHz, small signal bandwidth 700 KHz, slew rate 9 V/μsec. All of these are connected together through a Quanser Q4 acquisition board (Fig. 5).

The Q4 acquisition board is a powerful measurement and control board. Many devices with digital and analog sensors can easily connect to the Q4. This board is ideal for control systems and measurement applications.

Characteristics:

- High speed sampling rate 350kHz;
- 4 encoder inputs;
- 4 D/A voltage outputs;
- 4 analog inputs;

- Can be integrated with MATLAB/Simulink via QunaserWinCon solution.

Fig. 5. Universal Power Module UPM 1503.

2.2 PI design in Matlab-Simulink

The Simulink diagram it is composed from the following block: Pulse Generator, Gains, Scopes, Quanser Q4 Encoder Input, Quanser Q4 Analog Output, Summation blocks and Scopes.

The Pulse Generator block generates an input between 0 and 50degrees at regular intervals, this input is then passed through a Gain block that converts the input into radians. The Quanser Q4 Encoder Input reads the impulses given by the SRV02encoder;these impulses are converted to radians. The error between the reference and the encoder input is computed and passed through a PI controller. The PI controller was implemented using the following equation. The Kr and Ti gains are empirically determined.

$$H(s) = K_R\left[1 + \frac{1}{T_i s}\right] \quad (1)$$

$$K_i = \frac{K_R}{T_i} \quad (2)$$

Using the QunaserWinCon solution, the Simulink application was able to obtain the following results (Fig. 7), when a step response of 50 degrees was applied.

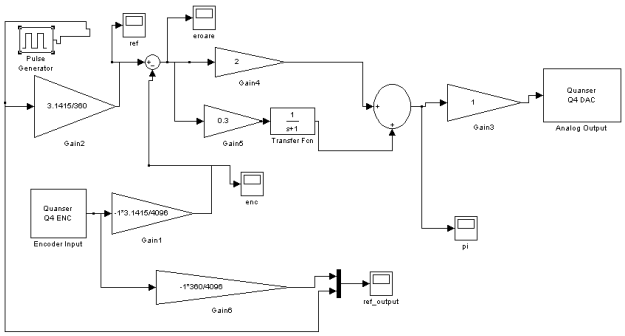

Fig. 6. PIcontroller in MATLAB - Simulink.

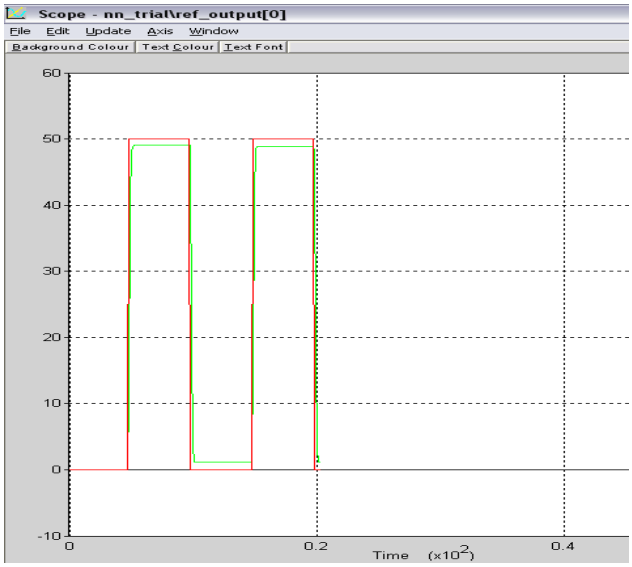

Fig. 7. The Step Response of a PI controller with an input of 50 deg.(y axis: deg., x axis: sec.).

3. ANN TRAINING IN MATLAB

MATLAB Neural Network Toolbox provides a framework for designing and implementing neural networks. MATLAB Neural Network Toolbox contains different Neural Network topologies such as: convolution neural networks, long short-term memory (LSTM) and generative adversarial networks (GANs).

3.1 Generating training data

Using the PI controller previously designed with an input of -180 to 180 deg., the following outputs were generated using the Scopes blocks implemented in the design: the reference, the error, the encoder input and the output provided by the PI controller. In order to reduce the unnecessary training data provided by the scope blocks, only the error output (Fig. 8) and the PI controller output (Fig. 9) were kept for training.

3.2 Training the Artificial Neural Network

Using the MATLAB Neural Network Toolbox, a simple Feed Forward Neural Network with one input neuron in the input layer, this neuron represents the error, five neurons in the hidden layer and one output neuron in the output layer, this neuron represents the output of the PI controller (Fig. 10).

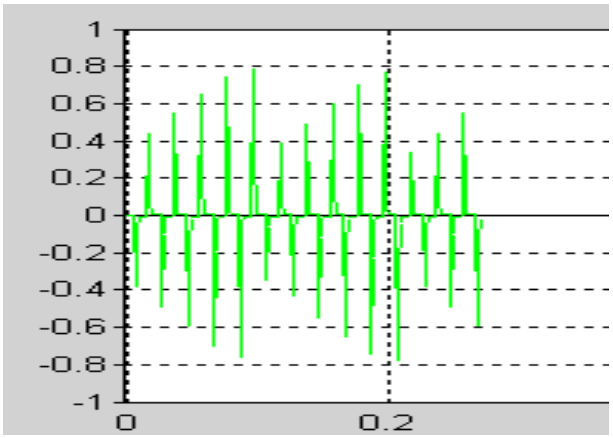

Fig. 8. Training data (error output)(y axis: rad., x axis: sec.)

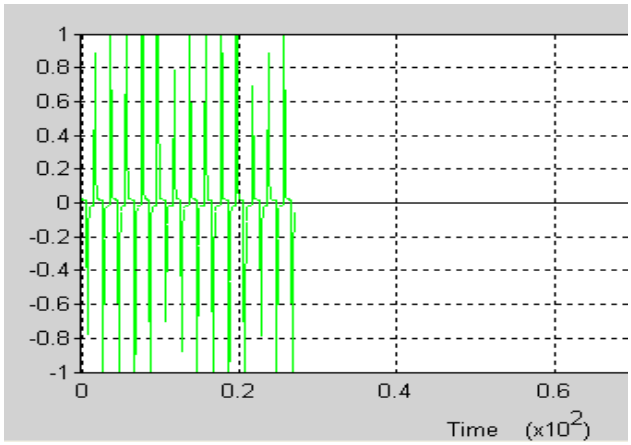

Fig. 9. Training data (PI controller output)(y axis: rad., x axis: sec.)

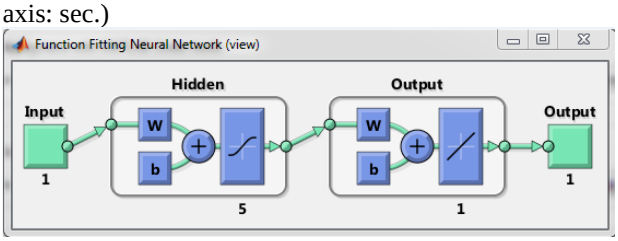

Fig. 10. The topology of the Feed Forward Neural Network

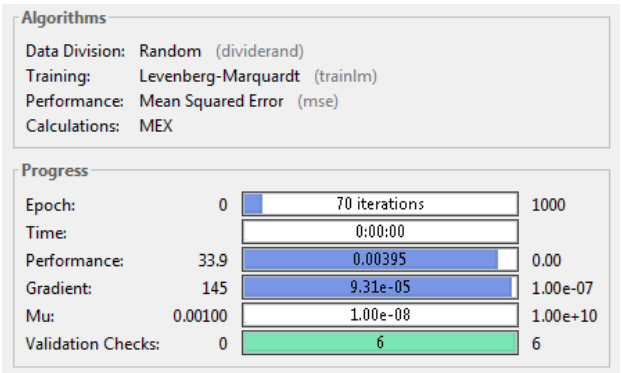

Fig. 11. Algorithm and Progress

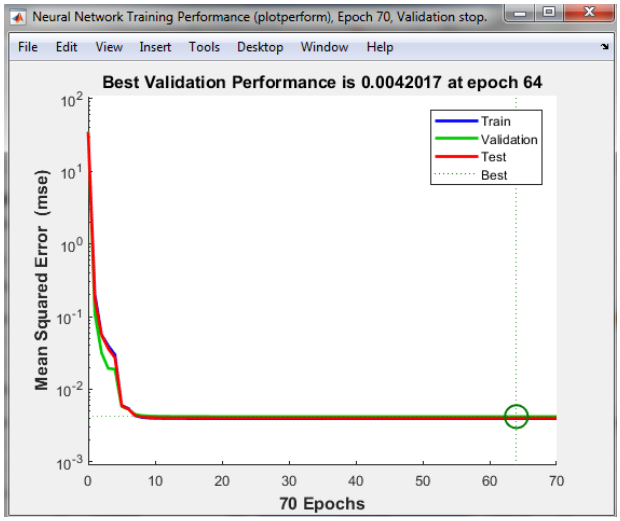

Fig. 12. Performance

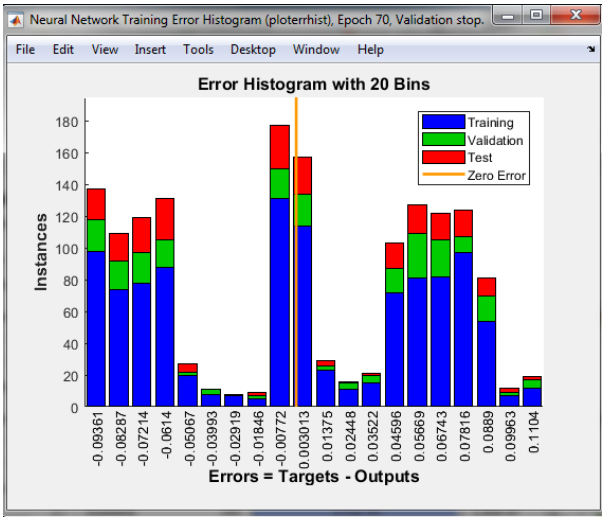

Fig. 13. Error Histogram

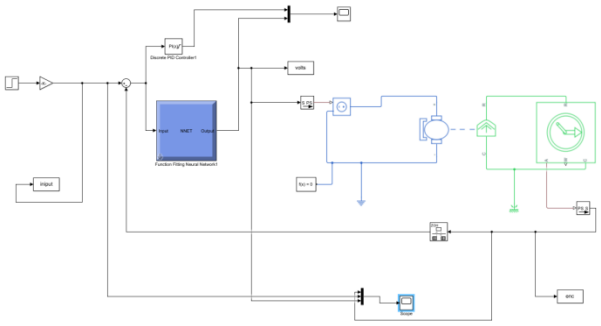

Fig. 14. The ANN in a simulated environment

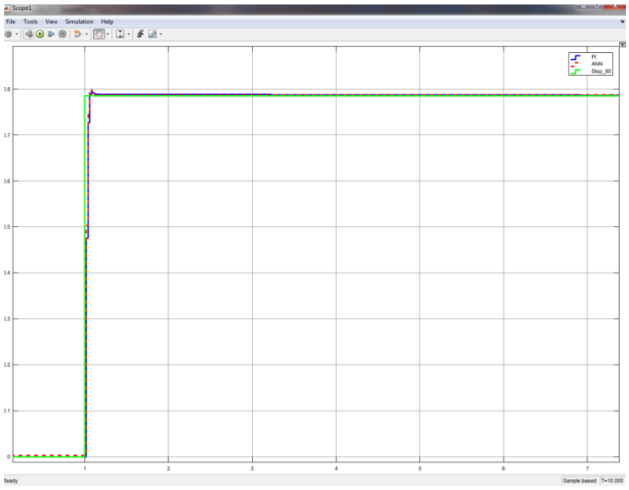

Fig. 15. The step response of an ANN and PI in a simulated environment

The data for training the network was divided randomly, the training was performed with Levenberg - Marquardt algorithm (Fig. 11).

3.3 Testing the ANN on a simulated environment

After the neural network had been trained, in order to prove that acts like a PI controller (Fig. 12, Fig. 13), a simulated environment was created in Simulink. The PI controller and the ANN receives the same reference, but only the ANN can control the DC motor. The DC motor has the same electrical and mechanical characteristics as the as the physical plant (Fig. 14). A step function with an amplitude of 90 degrees (0.785 rad.) was applied to the ANN input. The ANN has the same result as the PI controller (Fig. 15).

4. IMPLEMENTING THE ANN ON AN ARDUINO BOARD

The next step after the implementation of an ANN in MATLAB – Simulink is to downgrade the hardware that the ANN resides and implement a PI controller.

4.1 Arduino Hardware

For this step an Arduino MEGA 2560 R3 (ATmega2560 + ATmega16u2)(Fig.16)was chosen with the following characteristics:

- Operating voltage: 5V;
- Power supply Jack: 7V - 12V;

Fig. 16. Arduino MEGA 2560 R3

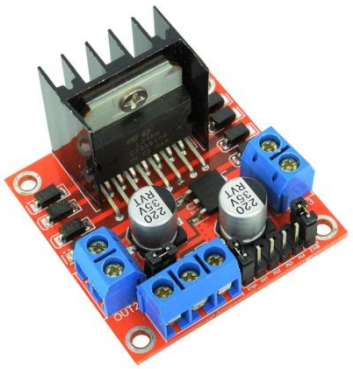

Fig. 17. L298N Dual Motor Driver

- I / O pins: 54;
- PWM pins: 15 (from I / O);
- Analog pins: 16;
- 4 x UART;
- Flash memory: 256KB, of which 8KB are occupied by the bootloader;
- Operating frequency: 16MHz.

The connection between the Arduino board and the DC motor is made through a L298N Dual Motor Driver (Fig.17) with the following characteristics:

- Motor voltage: 5V - 35V;
- Logic circuit voltage: 5V;
- Motor current: 2A (MAX);
- Logic current: 36mA;
- Maximum PWM frequency: 40kHz.

4.2 Software implementation of an ANN

The software is made from several functions:

- An SPI communication function;
- A bridge control function;
- A encoder function;
- An activation function (tansig);
- ANN function.

The ANN follows the same topology as the one from Simulink, one neuron in the input layer, 5 neurons in the hidden layer and one neuron in the output layer. Every neuron in the hidden and output layer has a weight and a bias. The input layer takes the difference between the reference and the encoder input, all of them converted into radians. The output layer is limited between -1 and 1 due to the Activation function. The weights and the biases were taken from the Simulink model.

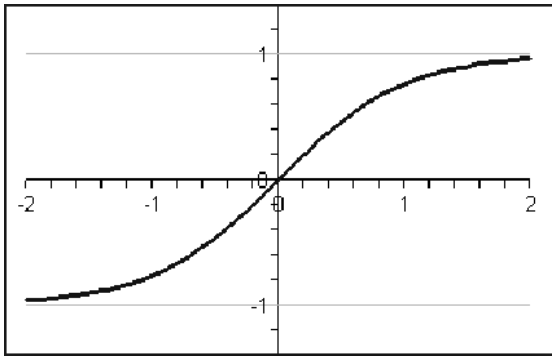

Fig. 18. Tan-Sigmoid Activation Function

Table 2. Hidden layer weights and biases

Weights	Biases
6.9408688360690060648	-6.4818336945281203043
5.5903787863739715647	-3.8951912402557224979
-0.7900194017891473130	-0.00720784506970014327
7.1922321473961501326	4.7170750208624925293
-5.5550886593272990055	-5.1628042671707987665

Table 3. Output layer weights and bias

Weights	Bias
0.039011975883084318295	-0.010528533666601457
0.0275710674929329283	
-1.3611649682569975095	
0.015152627955149008607	
-0.05181393509394715407	

The Activation function used is a Tan-Sigmoid Activation Function (Fig.18). This function is capable of having values between -1 and 1.

4.3 Software implementation of a PI controller

The PI controller program contains the same functions as the ANN one, except that the ANN is replaced with a PI controller.

The PI is computed using the following recursive formula:

$$b_0 = K_R * \left(1 + \frac{T}{2 * T_i}\right) \quad (3)$$

$$b_1 = K_R * \left(-1 + \frac{T}{2 * T_i}\right) \quad (4)$$

$$U_k = U_{k1} + e_k * b_0 + e_{k1} * b_1 \quad (5)$$

Where Uk is the current output, Uk1 is the previous output, ek the current error and ek1 the previous error.

Using the new gains, the response of the PI controller when a step function was applied can be observed in Fig. 19.

5. EXPERIMENTAL RESULTS

For the PI controller implemented in Arduino, in order for the output to follow the input, the Kr and Ki needed to have their value modified.

Table 4. PI controller gains used on the embeddeddevice

	Simulink	Arduino
Kr	2	100
Ki	0.3	30

For Arduino, the Kr and Ki were also empirically determined, which took more time than implementing the ANN. For the ANN just knowing the weights, biases, activation function and the topology we get the following result (Fig.20). There is an overshoot but without prior knowledge of the installation, computing the Kr and Ki gains is difficult.

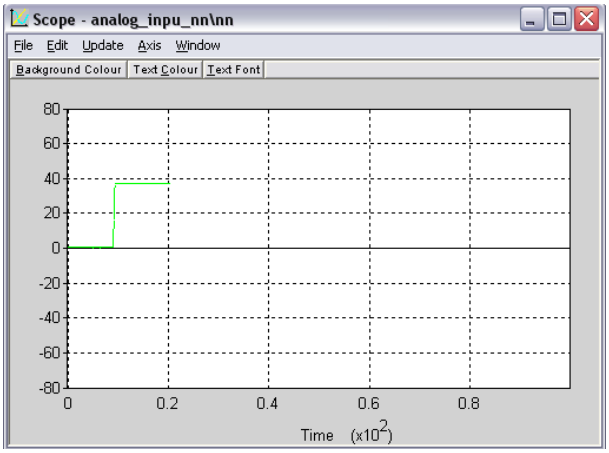

Fig. 19. The step response of and PI controller on an Arduino board (y axis: deg., x axis: sec.)

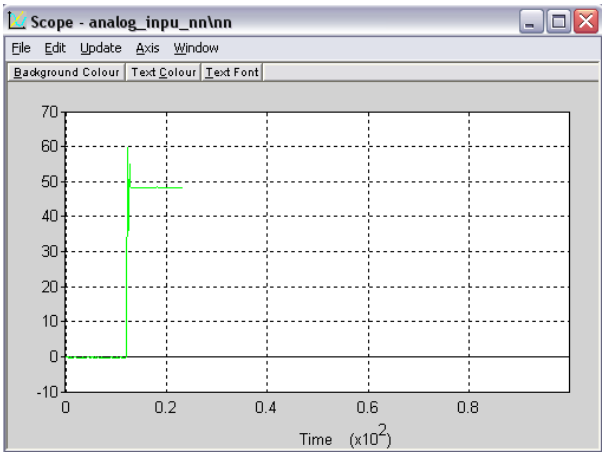

Fig. 20. The step response of an ANN on an Arduino board (y axis: deg., x axis: sec.)

6. CONCLUSIONS

The results proves that the PI control law can be replicated on an ANN. The ANN can be easily transferred to an embedded device with lower computation power and can give almost the same results without any modifications to the ANN. The Kr and Ki gains need to be modified in order to work on the Arduino board.

REFERENCES

A. K. Hassan, M. S. Saraya, M. S. Elksasy, and F. F. Areed, "*Brushless DC motor speed control using PID controller, fuzzy controller, and neuro fuzzy controller*", International Journal of Computer Applications, vol. 180, no. 30, pp. 47–52, 2018

Christopher M. Bishop, Oxford press, 1995, *Neural Networks for Pattern Recognition*, He also has a more recent book called Pattern Recognition and Machine Learning (Springer, 2006)

James A Anderson, *An Introduction To Neural Networks*, MIT Press, 1995.Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks (Reed, Marks, MIT Press, 1999)

J. Hertz, A. Krogh, R. Palmer, *Introduction to the theory of Neural Computation*, Addison Wesley, 1991

Juergen Schmidhuber(2015) (Cited: 2,196), *Deep learning in neural networks*

K. Warwick, G. W. Irwin, K. J. Hunt, *Neural Networks for Control and Systems*, U.K., Stevenage: Peregrinus, 1992.

M. Frean, *The Upstart algorithm: a method for constructing and training feed-forward networks*, Neural Computation, vol. 2, pp. 198-209, 1990.

M. Saerens, A. Soquet, *Neural-controller based on back-propagation algorithm*, *Proc. Inst. Elect. Eng.*, vol. 138, pp. 55-62, 1991.

R. Carelli, E. F. Camacho, D. Patiño (Patino), *A neural network based feedforward adaptive controller for robots*, *IEEE Trans. Syst. Man Cybern.*, vol. 25, pp. 1281-1288, Sept./Oct. 1995.

R. Hetcht-Nielsen, *Theory of the backpropagation neural networks*, Proc. Inter. Joint Conf Neural Networks, vol. I, pp. 593-611, 1989-June.

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol. 16 (43), No. 1, 2019

Instrumentation and processing application in automotive using MATLAB Simulink

Cătălin-Andrei Gheorghe*, Oana Mihaela Ciucă**

**Automation and Electronics Department, University of Craiova*
Craiova, Romania (e-mail: catalin96s@yahoo.com).
*** Automation and Electronics Department, University of Craiova*
Craiova, Romania (e-mail: ciucaoana96@gmail.com).

Abstract: This paper presents a method to analyze and process signals in automotive, designing an application in MATLAB Simulink, run on Arduino based microcontroller. The model based design allows implementing and testing more efficient, using the available blocks in Simulink libraries. The signals received from sensors can be analyzed in real time using support packages from MathWorks and the resulted models can be tested in model-in-loop (MIL) strategy.

Keywords: automotive, model based design, Simulink, data acquisition, model-in-loop.

1. INTRODUCTION

Every year, automotive systems become more and more complex. They are increasing in difficulty and cost to be designed successfully. Every customer wants to add new options and accessories to the future car model. Body electronics are very affected of this direction, a good example being the electrical trunk design.

According to R. Charette (2009), the first production car to incorporate embedded software was the 1977 General Motors Oldsmobile Toronado which had an electronic control unit (ECU) that managed electronic spark timing.

During that time, the trunk was mostly opened and closed manually. The few cars which had the electrical trunk available had a simple way of implementing it and it was presented to customer as on option.

Nowadays, this concept has become a trend. This involves meeting market and legislative requirements, which actually leads to the need of creating a control system designed to combine the input for several sensors and follow the complex requirements, keeping the deviations and errors as small as possible (Z. Zhung et al., 2019).

Traditional design methodologies in automotive involves creating and implementing the algorithms in C. Design is done for every module integrated in project as a text description of algorithms.(C. Andrici et al., 2014)

In case some hardware parts are missing, the engineer must wait for them to arrive, in order to test his implementation and see the system behavior.

This issue which can cost both time and money, can be avoided using an algorithm development as a model based design in Simulink. This decouples the software creation from hardware and also allows simulation of any kind of signal. (S. Wakitani et al., 2017)

For example, instead of using the same hardware ignition signal from car, in order to test the trunk functionality, the ignition can be simulated as input, following the exact curve and values as the real one, using MATLAB Simulink. Another advantage is represented by the fact that once the design has been checked and reviewed, the code can be quickly and efficiently generated via automatic code generation. The current paper contains an example of analyzing and designing the control of an electrical trunk in automotive using Simulink and Arduino based microcontroller.

With the latest release of Simulink Support Package for Arduino Hardware, this application can be directly built up in Simulink with extra blocks from other provided libraries.

Of course, this is only an example. There are many possibilities in which the model based design using Simulink can be used in automotive applications, most of them being already used by embedded programming companies.

As the car market and customer expectations expands, companies are successfully using the design approach, including Caterpillar, General Motors, Toyota,

Continental, Daimler, Jaguar and others (J. Friedman, 2006).

2. RELATED STUDIES

At Caterpillar, as in most automotive companies, the level of system complexity was out pacing the ability of mechanical control systems, resulting in increased demand for control software. They recognized that it needed to provide a mechanism to allow its controls groups that traditionally focused on mechanical systems design a means to develop innovative algorithms in software. As a result of their adoption of Model-Based Design, Caterpillar was able to reduce the man hours to develop and implement a standard project by a factor of 2-4. Caterpillar also found that the total project time was reduced by a factor of 2. Simply put, with half the staff, Caterpillar was able to complete their projects twice as quickly (J. Friedman, 2006).

E. Rapos (2014) observed that Simulink provides several added benefits for automotive development which make it an interesting technology to study. The versatility of the Simulink environment provides real-time simulation capabilities which allow for more reliable and accurate testing early in the development process.

T. Farkas et al. (2009) observed that the increased amount of software in automotive embedded systems has challenged its C code development to successfully manage software design, reuse, flexibility and efficient implementation. Model based methods help to address such challenges with more abstract specification, code generation and simulation to determine if software design will meet requirements. Therefore, migration concepts and adequate domain-specific methods with adoption of modeling languages and their tools in established embedded coding environments are needed.

3. OVERVIEW

The content of this paper presents the implementation of automatic control of a car trunk and all the necessary steps to analyse and process the signals from the system.

The acquisition of signals will be performed using the Atmega 328 microcontroller, and the interpretation, processing and generation of decisions based on inputs will be performed using MATLAB Simulink. Communication between Simulink and Atmega 328 is done through a platform offered by MathWorks, using serial communication as a basis.

The analog input signals are sampled and quantified by the analog-to-digital converter built in the microcontroller, and the result is transmitted cyclically, via SPI communication, to the PC unit. Therefore, evolutions of the input signals can be followed in real time, specific models can be created and the results of the experimental running on the microcontroller can be evaluated with precision.

The main hardware components used for electrical trunk control are:
• 12V DC motor control with built in hall effect encoder. The motor is used to open and close the electrical trunk. The Hall Effect encoder allows the system to track the motor position, leading to a precise actuation time needed to fully close or open the trunk.
• Current sensor ACS724 used to measure the current consumption of DC motor. Therefore, the obstacle can be detected by having a current consumption monitoring.
• Motor driver Pololu BD65496MUV which allows controlling of the 12V DC motor in both directions of rotation and also protects the hardware for overcurrent fault.

The algorithms will be simulated in real time, using MATLAB, having as inputs both physical signals and simulated signals. The results are plotted in the form of graphs, thus allowing the evaluation of the behavior over the entire execution period.

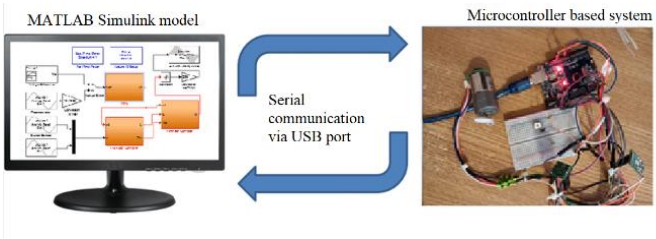

Fig. 1. Functional overview

4. ELECTRICAL TRUNK CONTROL

4.1 The acquisition and processing of the DC motor current consumption

The main advantage of using Simulink is the possibility to acquire in real time the signals read through the microcontroller and its ports. Therefore, to detect an obstacle, the application monitors the current consumed by the motor.

According to the laws of physics, if the load of a DC motor increases, the current consumed by it increases linearly, giving rise to an obstacle detection algorithm.

The current sensor ACS724 provides an output voltage corresponding to the intensity of the electric current flowing through it. The accuracy of the sensor used in the application is approximately 186mV/A. When the input current is zero, the sensor measures 2.5V. Based on the direction of current, the voltage is increased or decreased.

In the below graphic, it can be seen the shape of the input signal (plotted with the Scope block from Simulink) when the motor is running normally, without any external load:

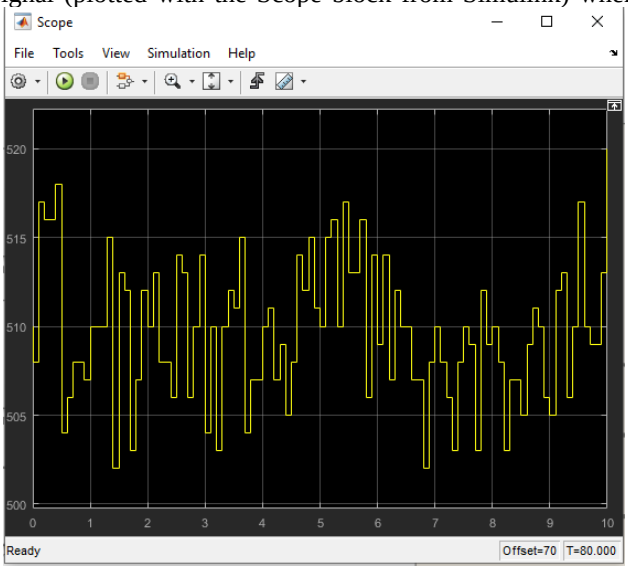

Fig. 2. Voltage generated by the current sensor in normal operation

The analog-to-digital converter used has a resolution of 10 bits. Since the maximum voltage received from sensor is 5V, the error of the ADC is 4.88mV. To have a more precise measurement, resolution of the ADC should be bigger.

Firstly, the average of the samples must be calculated in a time interval. The time interval was chosen as 3 seconds.

From Fig. 2, the average in 3 seconds is approximately 508 ADC units (it's calculated using MATLAB user defined function, with respect to the sample rate).

This is equivalent to 508*500/1023 = 2482mV ± 4.88mV. Considering that accuracy of the sensor is 186mV/A, 0.18mV from the sensor output is equivalent to 1mA.

After transformation from sensor datasheet, the current consumption of the motor is (2500 – 2482)/0.18 = 100mA ± 27mA.

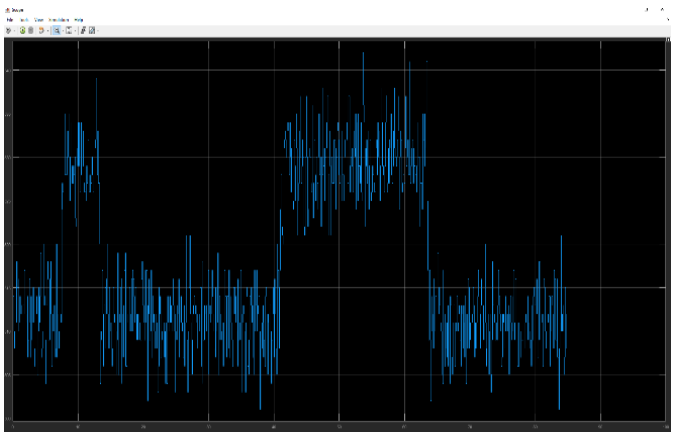

Fig. 3. The voltage generated by the load change of the DC motor in ADC units

As can be seen in Fig. 3, when the motor is blocked, encountering an obstacle, the current consumed by it increases, and in proportion to the current intensity also increases the voltage generated at the sensor output. Using the same logic as above, the current consumption of the motor calculated in an interval of 3 seconds is 500mA ± 27mA. Fluctuations and signal instability are caused by several factors: ADC conversion error, low motor current consumption (approximately 0.5A when completely blocked, powered at 12V), current conversion error of sensor.

However, based on this acquisition, a threshold for which the motor is considered blocked can be set. The threshold is defined as 450mA. If the current is greater than 450mA for a configurable time, then the motor is considered blocked or stuck and the trunk stops actuating, until a new command is received.

Using this system, the chance of having an accident by hitting the trunk while closing or opening is reduced significantly.

4.2 Acquisition and processing of the signal generated by the encoder

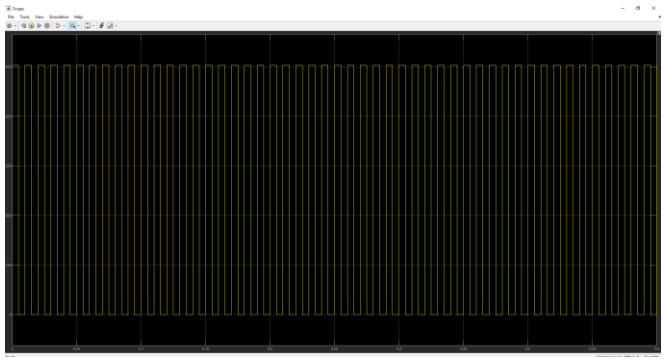

Fig. 4. Voltage signal generated by the encoder in ADC units, when the motor is supplied with 6 V

To determine the position of the motor, the application uses the pulses generated by the encoder during the rotation of the motor.

The signal received from encoder is represented by voltage in range of 0 – 2.5V. When the motor is rotating in one direction, the hall effect of the sensor generates a constant number of impulses with both logical levels high and low.

For a complete rotation there are 32 impulses received, according to datasheet. Is is very important to make sure that no hall impulse is lost, otherwise the position of the motor will not be precise and will lead to a wrong actuation

time of the trunk. To fulfill this requirement, the input signal must be read on event, using hardware interrupt.

Instead of using the polling concept to read the microcontroller input pin cyclically, an external interrupt will be set to trigger the read on the pin in case the voltage is greater than threshold.

This method is not dependent to the sample rate of the analog-to-digital conversion and also decreases the runtime and RAM consumption of the microcontroller. The interrupt is configured from Simulink using a special block from hardware support package library (Fig. 5).

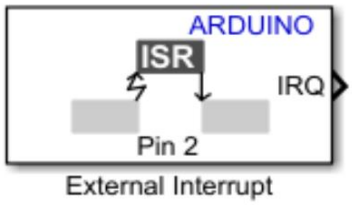

Fig. 5. External interrupt block in Simulink

4.3 Principle of operations. Simulink implementation.

Initially, the DC motor that controls the opening and closing of the trunk is stopped.

If the trunk button has been pressed and the trunk is open, it activates the rotation of the motor in the closing direction. As the motor rotates, it acquires and counts the number of pulses generated by the encoder. When the number of generated pulses has reached the corresponding value of open position, it stops the engine. Simultaneously, the application monitors the current consumed by the motor.

If it exceeds the current limit for normal actuation, meaning that an obstacle has been encountered on the route, motors is stopped and the fault is marked by flashing an LED and waits until a new ON ignition signal is received.

If the motor is stopped without any error and the trunk is closed, the application will activate the rotation of the engine in the opening direction. As the motor rotates, it acquires and counts the pulses generated by the encoder.

When the number of generated pulses has reached the corresponding value of closed position, it stops the motor. Simultaneously, the application monitors the current consumption of the motor.

Again as before case, if it exceeds the current limit for normal actuation, meaning that an obstacle has been detected, motor is stopped and the fault is marked by flashing an LED until a new ON ignition signal is received.

All threshold values used in the application are parameters, so they can be configured while in RUN mode. These are included in simulation by executing an external MATLAB file in the current workspace.

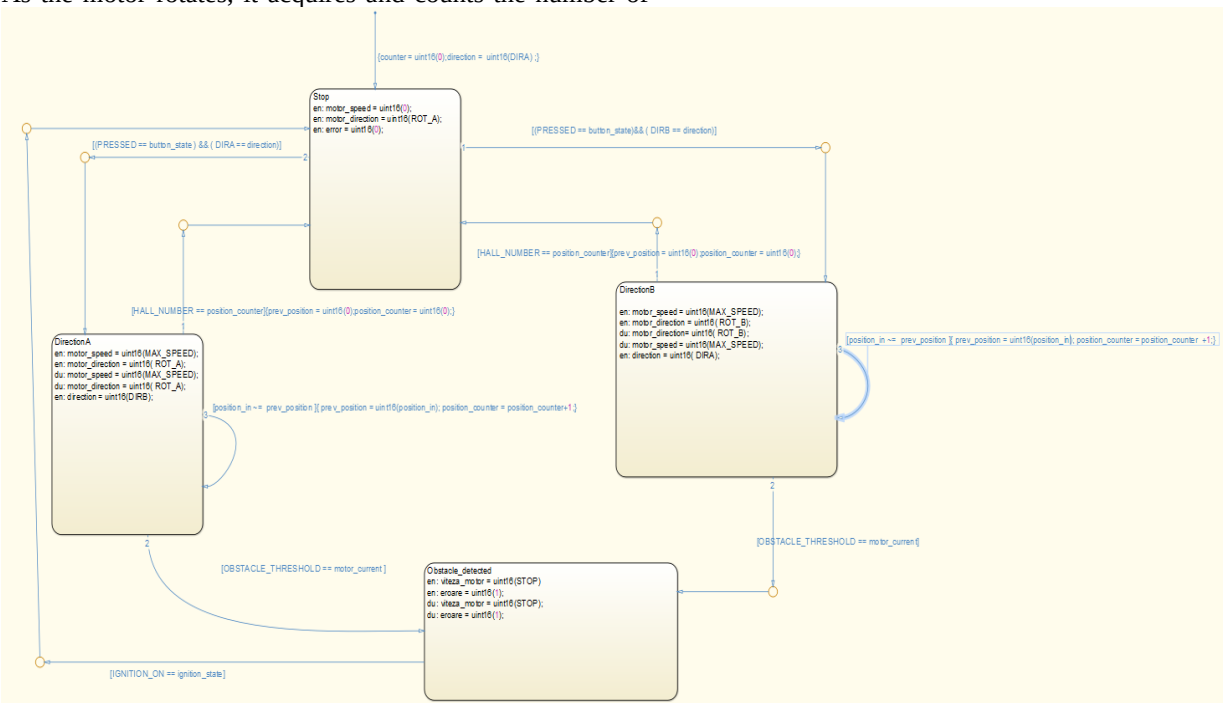

Fig. 6. Stateflow chart for trunk contro

A status diagram from Simulink Chart is used to create the trunk control algorithm. There are 4 system states required: Stop, DirectionA, DirectionB and Obstacle_Detected.

Default state is Stop.

The system outputs in this state are motor_speed equal to 0, motor_direction equal to ROT_A, and error (variable representing the result of obstacle detection) equal to 0. If the trunk button has been pressed and the motor has previously rotated in the opening direction (DIRA) then the system will switch to DirectionB state. In this state, rotation will be done in closing direction.

The corresponding outputs to this state are motor_speed equal to MAX_SPEED, motor_direction equal to ROT_B. The future direction will also be stored as direction A (after opening, the next time the button is pressed the trunk will close).

The first condition evaluated to get out of the current state is whether the number of pulses given by the encoder has reached the threshold value at which the trunk is considered closed. Thus the variable counter_position is incremented as long as it is not equal to HALL_NUMBER and only if the previous position is different from the current position. If before equality condition has been met, then the system status becomes Stop.

If the button has been pressed and the previous direction of rotation of the engine has been in the direction of closing (DIRB), a condition verified by evaluating the direction variable, then the next state of the system will be DirectionA.

In the A-direction state, the system outputs are motor_speed equal to MAX_ SPEED, motor_direction equal to ROTA. The next direction will also be stored as direction B (after closing, the next time when button is pressed the trunk will open).

The first condition assessed to get out of the current state is whether the number of pulses given by the encoder has reached the threshold value at which the trunk is considered open. Thus the variable counter_position is incremented as long as it is not equal to HALL_NUMBER and only if the previous position is different from the current position. If before defined equality condition has been met, then the system status becomes Stop.

If the system is in one of the Direction_A or Direction_B states, and the current consumed by the motor reaches the limit of the obstacle detection threshold, the system switches to Obstacle_Detected state. In this state, the system outputs are motor_speed equal to STOP, meaning that the motor does not rotate anymore and the error becomes present (1). The transition from this state is reached only if the ignition signal (ignition_state) is equal to IGNITION_ON.

In the end, the system will return to Stop state and the algorithm will be restarted from the beginning.

5. CONCLUSIONS

This paper demonstrates a way to implement and analyze an existing system on the car market: the control of the electric trunk of a car.

Using MATLAB Simulink, acquisition of input signals was performed, the model initially defined only conceptually was tested and validated, running the software on Atmega 328 microcontroller.

This method of programming, analyzing and processing signals within an embedded automotive system has a number of advantages, such as:

• Sampling and visualizing their evolution in real time, observing the influence of perturbations on the system;
• Creating and running models previously developed theoretically;
• Accurate reproduction of the real environment in which the embedded system will be used in reality;
• The evaluation of the system results is performed in real time, having the possibility of a real-time monitoring;

REFERENCES

Andrici, C., & Ipatiov, A. (2014). *Automotive multi-project architecture with model based development: An inside look at multi project handling within steering projects*. 2014 18th International Conference on System Theory, Control and Computing.

Farkas, T., Neumann, C., & Hinnerichs, A. (2009). *An Integrative Approach for Embedded Software Design with UML and Simulink*. 2009 33rd Annual IEEE International Computer Software and Applications Conference.

Friedman, J. (2006). *MATLAB/Simulink for Automotive Systems Design*, IEEE 2006 Design, Automation and Test in Europe - Munich, Germany.

Rapos, E. J. (2014). *Co-evolution of Model-Based Tests for Industrial Automotive Software*. 2014 IEEE International Conference on Software Maintenance and Evolution.

Wakitani, S., & Yamamoto, T. (2017). *Practice of model-based development for automotive engineers*. 2017 IEEE Frontiers in Education Conference (FIE).

Zhung, Z.-Y., Chen, K.-C., Yu, Y.-H., & Kwok, N. (2019). *Chip-based Anti-collision System for Car Door Opening*. 2019 4th International Conference on Intelligent Transportation Engineering (ICITE), pp. 322-326.

MATLAB and Simulink for Automotive: https://www.mathworks.com/

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol.16 (43) No. 1, 2019

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol.16 (43) No. 1, 2019

Operating system for real time applications in automotive

Oana Mihaela Ciucă*, Gheorghe Cătălin Andrei**
**Department of Automatic Control and Electronics, University of Craiova*
Craiova, Romania (e-mail: ciucaoana96@gmail.com).
*** Department of Automatic Control and Electronics, University of Craiova*
Craiova, Romania (e-mail: catalin96s@yahoo.com).

Abstract: This paper presents the configuration of a real-time operating system in applications with an impact on the safety and security of the user. It allows the possibility to use a processor type STM32F4001 as a control system in the automotive area for applications that require real-time control. Moreover, with the current advancement of today's STM32 processors, combined with a lower cost compared to current industrial control systems, the Nucleo board can be an excellent choice in reducing costs without compromising the quality or ability to achieve an automation of an embedded application system.

Keywords: real-time operating system, real-time control, embedded application system, automotive, Nucleo board, STM32

1. INTRODUCTION

Nowadays, as science evolves more and more, multitasking and the ability to perform many tasks quickly are very important to keep up with the fast work environment, in addition to work efficiency increasing.

In the automotive industry, most processes require an extremely fast execution time, so the system response ensures the safety and integrity of the user. A single-core processor can perform a single operation at a time. Sequential run of tasks, regardless of the duration or priority of each activity, could have a devastating effect. For example, starting and controlling the wipers for a car would mean that all other functions assisted by the machine's process computer (such as ABS, automatic braking system, light control, etc.) would be stopped until they were stopped, and the processor could control a single load, which is obviously impossible.(Walker, Richard, C. 2009)

A possible solution would be to use more multi-core processors to perform different tasks in parallel, but with the development of technology, thousands of tasks should be performed in parallel to perform all existing functionalities on top machines, considering the safety, user interface or comfort. This solution is proving to be extremely expensive and difficult to implement as the size, weight and cost of cars should increase significantly. In addition, in recent years there has been a decrease in the raw material needed to make microprocessors, which further hinders this solution.

Another simpler, cheaper, and easier to implement solution is a strict and careful organization and monitoring of the tasks that have to be performed, a list of priorities for each process executed so each task can be executed without blocking the execution of the others.

State of the Art

Furth et al. (1991) observed that applications and systems in real-time fields cover a diverse range. In the broad sense, real-time systems include systems for flight simulation, process control, medical laboratory automation, switching processing, on-line transaction processing, and even the multimedia systems.

In 1993 T. Soneoka, A. Oizumi and K. Suda observed that in real-time multi-tasking processes, the main requirement is to be able to process as many requests as possible, with strict restrictions on response time. Thus, switching systems can handle even hundreds of users. The number of tasks for these systems reaches the order of tens of thousands per second. In order to reduce the average response time and rise performance for this kind of systems, a preemptive priority method is adopted at the failure level and at the clock level, while a non-preemptive priority method is used for control at the base level, where call processing itself and off -hook supervision take place. The last method said that the priority is given to tasks with the shortest expected remaining processing time. T.W. Kuo and C.H. Li (1999) presented a two-level hierarchical scheduling scheme for an open system architecture with a fixed-priority scheduler. They show that the schedulability of any real-time application which adopt the stack resource policy cand be validated independently of other application. They developed a global resource synchronization mechanism where tasks in different applications can share global non-preemptable resources.

Later, J. Li, G, Zheng, H. Zhang and G. Shy proposed different task switching strategies corresponding to the

assigned tasks in the processing nodes of a real time processes and a task scheduling algorithm for heterogeneous real-time system. They demonstrated that using this algorithm we can avoid thrashing of scheduling system, missing deadlines of suspended tasks due to long waiting time.

Based on these topics, the article is structured as follows: Section 1 briefly presents relevant related studies and a short introduction. Section 2 provides a short overview of the system architecture. Section 3 presents the design of the multicore system using the configured operating system. Section 4 presents the conclusions of the article.

2. OVERVIEW

This paper presents the configuration of an operating system for the Nucleo 64 STM32F401 microcontroller aimed at controlling the exterior lights of a car, more precisely the adaptation of the position of the headlights according to the turning angle, hazard lights and turn signal lights. It is desired to monitor the signals and schedule the execution according to priorities.

The steering wheel position is constantly monitored with the help of a potentiometer, the output signal purchased from it is acquired as an input of an ADC of the Nucleo board and converted into a digital signal, then this information is processed and transformed into a value of the PWM duty cycle applied to the servomotor used for headlamp control. Signal acquisition and motor control are done on separate tasks that share a common resource that is written or read. The hazard lights or turn signal lights can also be activated by pressing the corresponding buttons.

The planner developed within the project uses a system for prioritizing tasks, periodically checking if a higher priority task has occurred.

Each task has assigned a certain priority and a unique state in which it can be at a given time. A system outage checks at a predetermined time if the current task has been executed or if a higher priority task has occurred and needs to be started. Less priority tasks are placed in a queue.

By running the processes one by one, suspending them, and resuming their execution, the so-called virtualization of the processor can be achieved.

With this project, we can demonstrate the possibility of using an STM32F4001 processor as a control system in the automotive area for applications that require real-time control.

Moreover, with the current advancement of today's STM32 processors, combined with a lower cost compared to current industrial control systems, the Nucleo board would be an excellent choice in reducing costs without compromising the quality or ability to achieve an automation of an embedded application system.

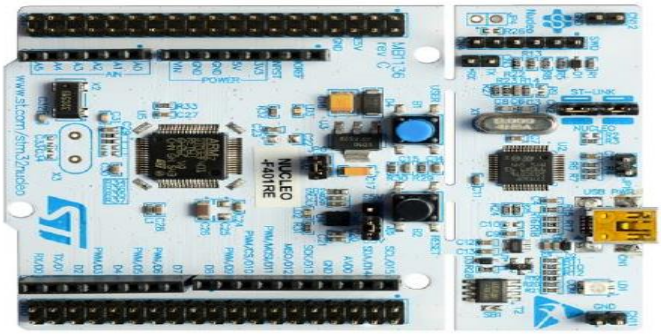

Fig 1. Nucleo 64 STM32 F401 controller board (ro.farnell.com)

The FreeRTOS operating system was chosen because it allows the configuration of an embedded system in the desired parameters, presenting a user-friendly interface. It also provides all the tools needed to set up a functional and efficient planner that meets the requirements.

3. DESIGN OF MULTICORE FREERTOS

Separate tasks that will monitor and control both the lights and the position of the headlights will be created.

A real-time application that uses an RTOS can be structured as a set of independent tasks. Each task is performed in its own context, without accidental dependence on other tasks in the system or on the RTOS scheduler itself. Only one task in the application can be executed at any one time, and the real-time RTOS scheduler is responsible for deciding which task it should be. Therefore, the RTOS scheduler can repeatedly start and stop each activity (change each activity in and out) as the application runs.

For this application, four tasks are sufficient to implement the acquisition of data on the position of the steering wheel, for processing and sending the command to the servomotor, for the control of emergency lights and for the control of turn signal lights.

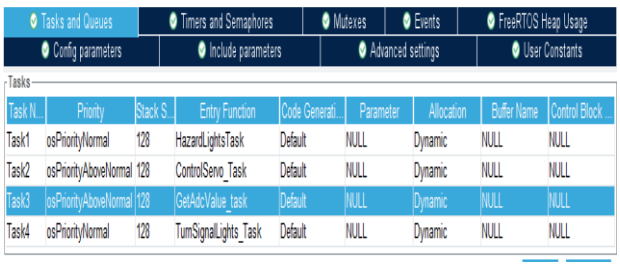

Fig. 2. Tasks configuration

In order not to load the system, the control of the lights and the ON / OFF period is done by using interrupts. For the same reason, pressing the buttons also generates an external interrupt. These interrupts are unmaskable and cannot be interrupted by any other interrupt.

3.1 Timer configuration

The input frequency for the internal clock is 8MHz and for setting the timers it was desired to set a frequency of 50MHz. STM32 CubeIDE provides general purpose

timers. One general purpose timer corresponding to the APB2 prequalifier was used to control the ON / OFF period of the lights. The calculation formula for the period is presented below:

$$T = \frac{1}{APB\_TIM\_CLK\ in\ MHz} \times (Prescaler_{Value} + 1) \times (Period\_Value + 1)$$

where:

APB_TIM_CLK = 8MHz; PRESCALER_Value = 50000 -1; PERIOD_Value = 10000-1.

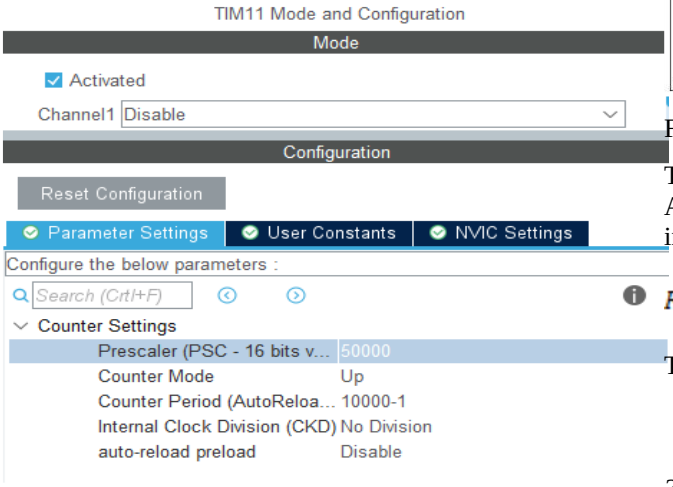

Fig. 3 Timer configuration

3.2 PWM configuration

The control of the servomotor was performed using a PWM applied on one of the analog pins which is connected to the data pin of the servomotor. The cannel was configured as "PWM Generation".

PWM mode allows the generation of a signal with a frequency determined by the value of the TIMx_ARR register, and an operating cycle determined by the value of the TIMx_CCRx register. In PWM mode, TIMx_CNT and TIMx_CCRx are always compared to determine if TIMx_CCRx and TIMx_CNT or TIMx_CNT & TIMx_CCRx are used (depending on the direction of the counter).

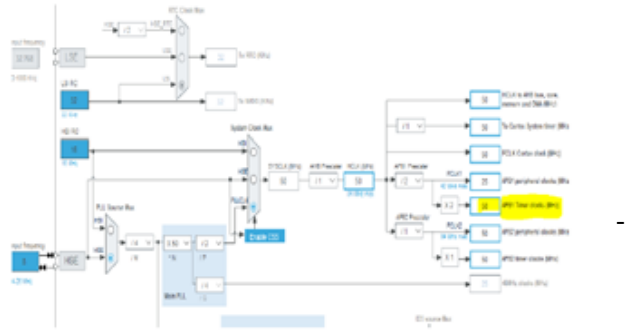

Fig. 4 PWM frequency configuration

Receiving the clock frequency from an internal source, the output pin that is configured in PWM mode is set to HIGH and a counter starts until the value loaded in the CCRx register is reached. At that time, the output pin is set to LOW. Thus, the value of the filling factor is actually loaded in the CRRx register.

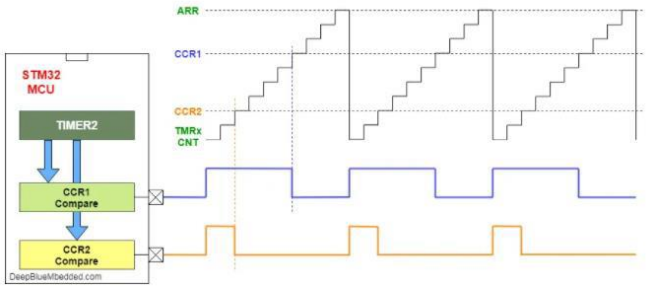

Fig. 5 PWM Output Channels (M. Chaled, 2019)

The PWM period (1 / F_{PWM}) is calculated based on the ARR register value, the prescaler value and the F_{Clk} internal clock frequency.

$$FPWM = FCLK \frac{FCLK}{(ARR + 1) \times (PSC + 1)}$$

The duty cycle is calculated as follows:

$$\text{DutyCyclePWM}[\%] = \frac{CCRx}{ARRx}[\%]$$

3.3 ADC configuration

The potentiometer used to acquire the steering wheel position data at a certain time provides an analog signal output that must be converted to a digital signal in order to be further processed. For this, the ADC module was used on one of the analog pins.

The configuration of the ADC module for conversions can be done in 3 different ways depending on the types of applications and requirements.

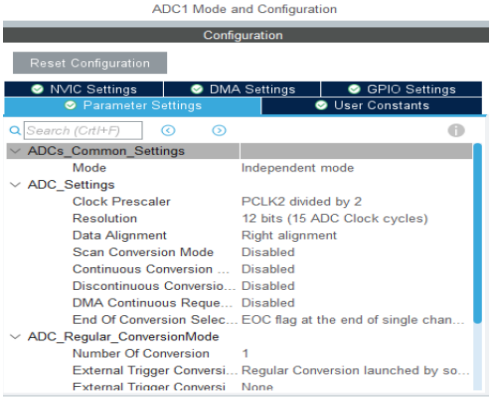

Fig. 6 ADC Configuration

- *Pooling mode* - an ADC conversion is started, and the processor is stopped to wait for the ADC conversion to complete. Only after the ADC conversion is complete can the CPU resume executing the master code.
- *Interrupt mode* - the ADC can be triggered to start a conversion and the CPU continues to execute the main code routine. When the conversion is complete, the ADC triggers an interrupt, and the processor is notified so that

it can switch the context to the SRI handler and save the ADC conversion results.

The risk introduced by using this method is that for many conversions, the CPU load can increase too much.

- *DMA* - The DMA unit can directly transfer the ADC result from the peripheral to the memory, all this being done without any intervention of the processor. After DMA transfers the data to a 1kb buffer, it can notify the processor to process the resulting data. A DMA module is an integrated device that performs fast data transfer between various physical components such as between memory and peripherals or between memories and even between peripherals without the involvement of the processor.
In this way the transfer can be executed in parallel with the execution of tasks.

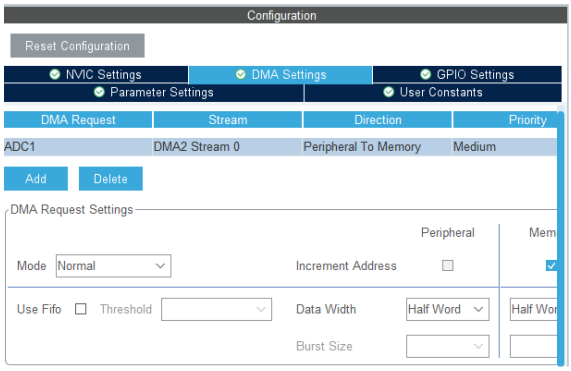

Fig. 7 DMA configuration

3.4 NVIC configuration (Nested Vector Interrupt Controller)

Nested Vector Interrupt Controller offers several features for efficient exception handling. When an interrupt is being scaled and a new request occurs, it is checked whether the one that interrupts it has a higher priority, and if this is met, the new

exception can interrupt the current one. This is called nested exception handling.

The exception handler resumes execution after the higher priority exception is handled. To complete the requirements, NVIC was configured as in the figure 8(a., b.).

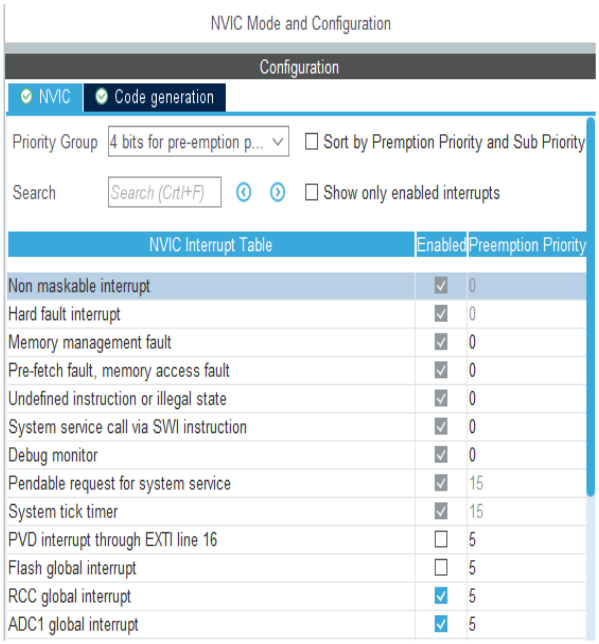

Fig. 8 a) NVIC configuration

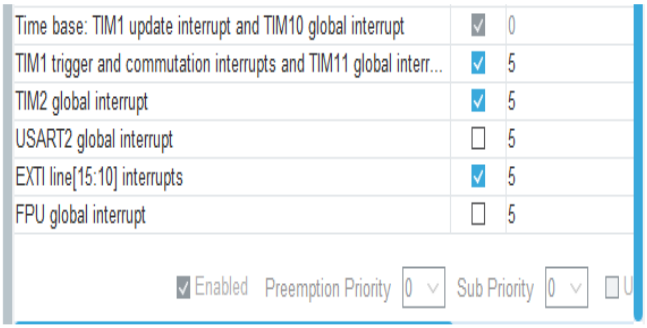

Fig. 8 b) NVIC configuration (Timers and interrupts)

3.5 Binary semaphores

The resources access was based on binary semaphores or mutexes. They are used to resolve the mutual exclusion of a shared resource or code sequence, to ensure that a single process accesses its critical region at a time. Binary semaphores are types of data that can have two values, true or false.

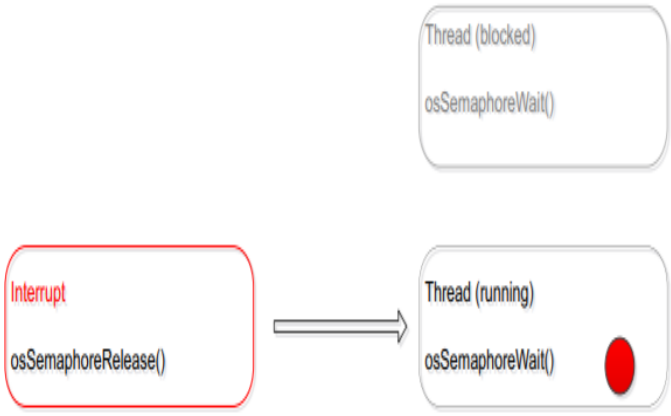

Fig. 9 Interrupt used to reach the semaphore (STMicroelectronics, 2019)

Emergency lights and turn signal lights use the same common resource represented by a common LED. Simultaneous ignition is not possible. For this reason, the use of binary semaphores was chosen to block simultaneous access to this resource. The same was used to read and write the common variable used by ADC to control the servomotor position. In this way a resource can be used at a certain point in time by a single task, avoiding overwriting the information. When one task needs the resource, it has to check if the semaphore is not already in use. If the semaphore is free, then the task can lock the needed resource.

The scheduler of the real-time system can decide the priority of the task to the shared resources.

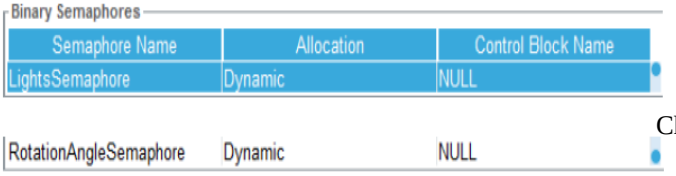

Fig. 10 Binary semaphore

3.6 External interrupts

The start of emergency lights or turn signal lights was achieved by pressing two buttons, by activating the external interrupts corresponding to each button, connected to the digital pins.

A function associated with each interrupt has been implemented, so when the button is pressed, a flag is set or reset.

```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_PIN)
{
    if((GPIO_PIN == GPIO_PIN_13)&&(button_pressed == 0))
    {
        avarie_ON = 1;
        button_pressed = 1;
    }
else if((GPIO_PIN == GPIO_PIN_13 )&&(button_pressed == 1))
    {
        avarie_ON = 0;
        button_pressed = 0;
    }
}
```

Fig. 11 Callback function associated to the external interrupt

4. CONCLUSIONS

The aim of the paper is to implement and configure a real-time operating system for embedded applications, demonstrating its usefulness for implementing systems that require a fast response time. This goal has been largely achieved by effectively managing three of the most common applications used for automobiles. The hardware support used effectively meets the requirements and manages to meet, in a minimal way, the constraints related to the speed of acquisition or response. The existence of a single core makes the controller used a perfect model of the development board that manages to meet both the requirements related to cost and efficiency. The paper presents many possibilities for development, offering the possibility to add more applications or to integrate new modules. There are also a variety of development directions, including scheduler optimization and support for a wide range of processors.

REFERENCES

Chaled, M., (2019). *STM32 PWM Example – STM32 Timer PWM Mode & LABs*: https://deepbluembedded.com/stm32-pwm-example-timer-pwm-mode-tutorial/

Furht, B., Grostick, D., Gluch, D., Rabbat, G., Parker, J., McRoberts, M. (1991). *Real-Time UNIX Systems Design and Application Guide*, Springer US, US.

Kuo, T.W., Li, C.H. (1999) *A fixed-priority-driven open environment for real-time applications*, Proc. 20th IEEE Real-Time Systems Symposium (Cat. No.99CB37054).

Li, J., Zheng, G., Zhang, H., & Shi, G. (2019). Task Scheduling Algorithm for Heterogeneous Real-time Systems Based on Deadline Constraints, *2019 IEEE 9th Int. Conf. on Electronics Information and Emergency Communication (ICEIEC)*.

NUCLEO-F411RE Development Board, *STM32 Nucleo-64, STM32F411RE MCU*: https://ro.farnell.com/stmicroelectronics/nucleo-f411re/dev-board-cortex-m4-mcu/dp/2433469

Soneoka, T., Oizumi, A., Suda, K. (1993). Highly multi-tasking real-time systems and their ev.aluation, 1993 *Proc. Real-Time Systems Symposium*, pp 249-252.

STMicroelectronics, (2019). *UM1722 User Manual Developing applications on STM32Cube with RTOS* : https://www.st.com/resource/en/user_manual/dm00105262-developing-applications-on-stm32cube-with-rtos-stmicroelectronics.pdf

Walker, Richard, C. (2009) *Automated devices to control equipment and machines with remote control and accountability worldwide*, USA: https://www.ic.gc.ca/opic-cipo/cpd/eng/patent/2335155/summary.html?pedisable=true

ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol.16 (43) No. 1, 2019

AUTHOR INDEX