EVALUATION OF SCHEDULING ARCHITECTURES FOR OSEK/VDX COMPLIANT HARD REAL-TIME OPERATING SYSTEMS

A Thesis Submitted to
the Graduate School of İzmir Institute of Technology
in Partial Fulfilment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Berkay SAYDAM

August, 2020

İZMİR

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, *Assoc. Prof. Dr. Tolga AYAV* who provided full support at all stages of my thesis process and contributed to my academic development.

I would like to thank *Prof. Dr. Cüneyt F. BAZLAMAÇCI* and *Assist. Prof. Dr. Hüseyin HIŞIL* for their participation as committee members.

I would like to express my endless thanks to my parents and my brother who helped me on this challenging path.

Special thanks to the faculty members who gave the chance of attending graduate programs in Izmir Institude of Technology Department of Computer Engineering and support throughout my thesis.

ABSTRACT

Evaluation of Scheduling Architectures for OSEK/VDX Compliant Hard Real-Time Operating Systems

Technological advancements are reflected to the vehicles as well, but it brings the challenge of adding new functionalities to vehicles without compromising safety. Tasks are used to provide functionalities which are used in car. These tasks have different characterictics. Safety and performance are two main criteria to determine characterictic of tasks. Characteristics of tasks can be classified according to their safety levels which are known as Automotive Safety Integrity Levels. Designing of hardware and software and also testing them is a long progress in automotive industry. Any changes on the design of hardware is quite costly when an ECU began to be used in field. According to my hypothesis, scheduling algorithms which are used by Central Processing Unit to determine sequences of task executions, should be well known. Besides, designing of hardware and software should be done according to these characteristics and algorithms. If not, tasks will cause serious problem like missing deadline for safety-critical component. In this thesis, the scheduling architectures are evaluated and they are determined which scheduling architectures should be used for which purpose. Besides, the advantages and disadvantages are explained.

ÖZET

OSEK/VDX Uyumlu Katı Gerçek-Zamanlı İşletim Sistemleri için Zamanlama Mimarilerinin Değerlendirilmesi

Teknolojik gelişmeler araçlara da yansıyor ancak güvenlikten ödün vermeden araçlara yeni işlevler ekleme zorluğunu da beraberinde getiriyor. Görevler, arabada kullanılan işlevleri sağlamak için kullanılır. Bu görevler farklı karakteristiklere sahiptir. Güvenlik ve performans, görevlerin karakteristiğini belirlemek için iki ana kriterdir. Görevlerin karakteristikleri, Otomotiv Güvenlik Bütünlük Seviyeleri olarak bilinen güvenlik seviyelerine göre sınıflandırılabilir. Donanım ve yazılım tasarımı ve ayrıca test edilmesi otomotiv endüstrisinde uzun bir ilerlemedir. Bir ECU sahada kullanılmaya başlandığında, donanım tasarımındaki herhangi bir değişiklik oldukça maliyetlidir. Benim hipotezime göre, Merkezi İşlem Birimi tarafından görev yürütme sıralarını belirlemek için kullanılan zamanlama algoritmaları iyi bilinmelidir. Ayrıca bu karakteristik ve algoritmalara göre donanım ve yazılım tasarımı yapılmalıdır. Aksi takdirde görevler, kritik bileşen için son tarihin kaçırılması gibi güvenlik açısından ciddi sorunlara neden olacaktır. Bu tezde, zamanlama mimarileri değerlendirilerek hangi zamanlama mimarilerinin hangi amaçla kullanılması gerektiği belirlenir. Ayrıca avantajları ve dezavantajları anlatılır.

I would like to dedicate this thesis to my parents *Arzu* and *Mustafa SAYDAM* and also my brother *Mehmet SAYDAM*.

TABLE OF CONTENTS

LIST OF FIGURES vii

LIST OF TABLES viii

ABBREVIATIONS ix

CHAPTER 1. INTRODUCTION 1

CHAPTER 2. SAFETY IN AUTOMOTIVE INDUSTRY 4

2.1 Safety Standards 4

2.2 V-model 5

2.3 Automotive Safety Integrity Levels 6

2.4 MISRA C 7

CHAPTER 3. REAL-TIME OPERATING SYSTEMS 8

CHAPTER 4. SCHEDULING ALGORITHMS 13

4.1 Event Triggered Scheduling 13

4.1.1 Rate Monotonic 14

4.1.2 Earliest Deadline First 15

4.1.3 Comparison between RM and EDF 16

4.2 Time Triggered Scheduling 16

CHAPTER 5. HARD REAL-TIME OPERATING SYSTEMS 19

CHAPTER 6. IMPLEMENTATION AND EVALUATION 23

6.1 The Board Specifications 23

6.2 Implementation 26

6.3 Evaluation 30

CHAPTER 7. CONCLUSION 43

REFERENCES 44

LIST OF FIGURES

Figure **Page**

Figure 2.1 Derivation of ISO 26262 5

Figure 2.2 V-model 5

Figure 2.3 ASILs 6

Figure 3.1 Spectrum of usage fields 8

Figure 3.2 Behavior of real-time systems 9

Figure 3.3 Task State Machine 10

Figure 3.4 Classification of Scheduling Algorithms 12

Figure 4.1 The formula of utilization calculation 14

Figure 4.2 The calculation steps of RM feasibility 15

Figure 4.3 Representation of TT Scheduling 17

Figure 5.1 The founders of OSEK/VDX 21

Figure 6.1 Pin diagram of Arduino UNO board 24

Figure 6.2 The connection of board 25

Figure 6.3 Erika downloading package content 26

Figure 6.4 Configuration for Arduino UNO 27

Figure 6.5 Successful build console output 27

Figure 6.6 Creating upload target 28

Figure 6.7 Successful flashing console output 29

Figure 6.8 The utilization calculation of the scenario 30

Figure 6.9 OIL File Configuration for RM 30

Figure 6.10 The UART output of RM 31

Figure 6.11 The visualization of RM output 31

Figure 6.12 OIL File Configuration for EDF 32

Figure 6.13 The UART output of EDF 33
Figure 6.14 The visualization of EDF output 34
Figure 6.15 The UART output of TT Scheduling 35
Figure 6.16 The visualization of TT Scheduling output 35

LIST OF TABLES

Table **Page**

Table 3.1 Comparison between hard and soft real-time systems 9

Table 4.1 Comparison between RM and EDF 16

Table 5.1 The classification of popular hard real-time operating systems 19

Table 6.1 The scenario parameters to be used 29

Table 6.2 The testing outputs for different CPU Utilization with more tasks 36

ABBREVIATIONS

ECU	Electronic Control Unit
ISO	International Standardization Organization
IEC	International Electrotechnical Commission
E/E	Electric/Electronic
ASIL	Automotive Safety Integrity Level
MISRA	Motor Industry Software Reliability Association
CPU	Central Processing Unit
I/O	Input/Output
RR	Round Robin
EDF	Earliest Deadline First
RT-POSIX	Real Time-Portable Operating System Interface
OSEK	Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen
VDX	Vehicle Distributed eXecutive
OS	Operating System
API	Application Programming Interface
RTOS	Real-Time Operating System
AUTOSAR	AUTomotive Open System ARchitecture
OIL	OSEK Implementation Language
XML	Extensible Markup Language
PWM	Pulse-Width Modulation
USB	Universal Serial Bus
ICSP	In-Circuit Serial Programming
LED	Light Emitting Diode
RM	Rate Monotonic
FP	Fixed Priority
UART	Universal Asynchronous Receiver Transmitter
WCET	Worst Case Execution Time
TT	Time Trigger

CHAPTER 1

INTRODUCTION

The development of technology creates different needs in the sectors. These developments help to create smart cars in the automotive industry. Enhancement of vehicle performance is important to adding new functionalities without compromising safety. Electronic Control Unit (ECU) is an embedded system that controls some electrical systems in a car. Microcontrollers which are used to provide the functionalities, are determined in the first step of ECU design. This decision effects the remaining steps of the project development. Designing of hardware and software and also testing them is a long progress in automotive industry. Any changes on the design of hardware is quite costly when an ECU began to be used in field. For these reasons, the selection of microcontroller effects cost and also desired functionality which should be provided to customer.

In automotive systems, tasks are used to provide functionalities which are used in car. These tasks have different characterictics. Safety and performance are two main criteria to determine characterictic of tasks. Their characteristics can be classified according to their safety levels which are known as Automotive Safety Integrity Levels (ASILs). ASIL-D tasks require strict response time when ASIL-A tasks require multiple process to obtain performance. According to my hypothesis, scheduling algorithms which are used by Central Processing Unit (CPU) to determine sequences of task executions, should be well known. Besides, designing of hardware and software should be done according to these characteristics and algorithms. If not, tasks will cause serious problem like missing deadline for safety-critical component. It also occurs cost because of multiple hardware and software design cycle.

In this thesis, the scheduling architectures are evaluated and they are determined which scheduling architectures should be used for which purpose. Besides, the advantages and disadvantages are explained. Thus, a software architect, who is responsible for designing ECU, will know which scheduling architectures should be used purposeful and also he/she is able to make a decision for correct microcontroller according to their criteria and design ideas which are given in conclusion section. As a

benefit to the automotive industry, this study will be answer to whether an ECU should contain more than one microcontroller and also these microcontrollers should be chosen according to what purpose when it includes.

In analysing phase, necessary environment was created to show pros and cons of scheduling algorithms. Erika Operating System (Erika OS) [18] was selected because it is free and OSEK/VDX (Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen/Vehicle Distributed eXecutive) certified Operating System. Erika OS was built and embedded to Arduino Uno board. Rate Monotonic (RM), which is Fixed Priority (FP) Scheduling Algorithm, was first scheduling algorithm to analyse. Resulting behaviour of this scenario was visualised. There was missing deadline but it had the lowest priority. This is a disadvantage for RM, however it missed deadline for the lowest priority task. The task can be ASIL-A like a task to showing door status with opens interior light. The latency of this task can not cause catastrophic consequences in this system. This missing deadline can be eliminated with decreasing CPU Utilization. This analysis shows us that RM, which is a FP scheduling algorithm, can be used to create a safe system by renouncing performance.

The next scheduling algorithm is Earliest Deadline First (EDF) which is a Dynamic Priority Scheduling Algorithm. It needs less effort than FP, because it automatically schedules the tasks according to their deadlines at runtime. This time, same scenario was run for EDF. There was no missing deadline. This shows us that EDF can provide CPU Utilization higher than FP. However this does not show that EDF has no disadvantages. EDF guarantees the fulfillment deadline for each task, but it is not possible to provide a minimum response time for any given task. The highest priority task always has a minimum response time in RM, but it is not possible to guarantee for EDF. If the system becomes overloaded because of adding a new task or a wrong Worst Case Execution Time (WCET) estimation, domino effect can be occurred. Domino effect is that all other tasks miss their deadline after one task missed its deadline with executing more than its declared runtime. This is possibility of missing deadline of ASIL-D task like a task which opens airbags at crash time. This condition causes catastrophic consequences in this system. Therefore, EDF can be used for tasks which need high performance and low safety.

These were pros and cons of online scheduling algorithms. Automotive industry does not have only two parameters as performance and safety. Determinism and predictability are also other important parameters for developers working in automotive industry. Therefore, Time Triggered (TT) scheduling architecture is important in this industry. OSEK Implementation Language (OIL) file is used commonly in automotive industry to adjust fundamental parameters for Operating System (OS). This architecture has schedule table which includes offline defined tasks and their triggering times. This schedule table was prepared in OIL file and each task calls at a certain time. If needs a comparison between event triggered scheduling architecture and time triggered scheduling architecture, event triggered is flexible and is characterized by high resource utilization. However, it is not reliable from the determinism and predictability point of view which are important aspects in hard real-time systems. Besides, Event triggered has higher runtime overhead than time trigger. Time triggered has a periodic world. It is not flexible for sporadic tasks. It provides determinism and predictability but with lower rate of resource utilization.

Safety is very important topic in automotive industry. Some standards are created to provide secure development by automotive companies. These standards are discussed in chapter 2. Time is significant parameter to provide safe transportation at vehicles. Real-time systems are widely used because of providing to guarantee that have response within a specified timing constraint. There is detailed information about these systems in chapter 3. Each real-time system has CPU to manage tasks in systems. CPU should make decisions to determine tasks according to sequence. Therefore, it needs an algorithm. They are called scheduling algorithms. They are represented in chapter 4. Real-time systems can be classified according to the hazards they cause when the system response occurs too late. These are soft real-time systems, firm real-time systems and hard real-time systems. Real-time systems use special operating systems according to needs. Existing operating systems, which are used for hard real-time systems, are discussed in chapter 5. Scheduling algorithms, which are described in chapter 4, were implemented with hard real-time operating system which is described in chapter 5. Implementation steps, findings and evaluations are added to chapter 6. Results were deduced according to chapter 6 and it discussed in conclusion chapter.

CHAPTER 2

SAFETY IN AUTOMOTIVE INDUSTRY

In this chapter, the importance of safety in automotive industry is highlighted and the existing standards on safety are presented and reviewed. Derivation of ISO 26262 from IEC 61508 will be shown. Then, there will be detailed expressing about V-model and ASILs which are found in ISO 26262 standards. The last section will present knowledge about MISRA to develop safe software.

In dictionary, safety means the condition of being protected from danger, risk, or injury. A safe system is one which does not cause harm to people. There is no system which is completely safe, so that safe systems attempt to reduce potential risk to an acceptable level.

2.1 Safety Standards

The major communities in automotive industry created some standards to provide protection for human. One of them is International Standardization Organization (ISO) 26262 [1]. It is actually derived from International Electrotechnical Commission (IEC) 61508 [2]. IEC 61508 is a basic functional safety standard applicable to all kinds of industry. Also, ISO 26262 is an adaptation of IEC 61508 for Automotive Electric/Electronic (E/E) Systems. ISO 26262 restricts processes at entire product development lifecycle of automotive E/E products within the framework of safety.

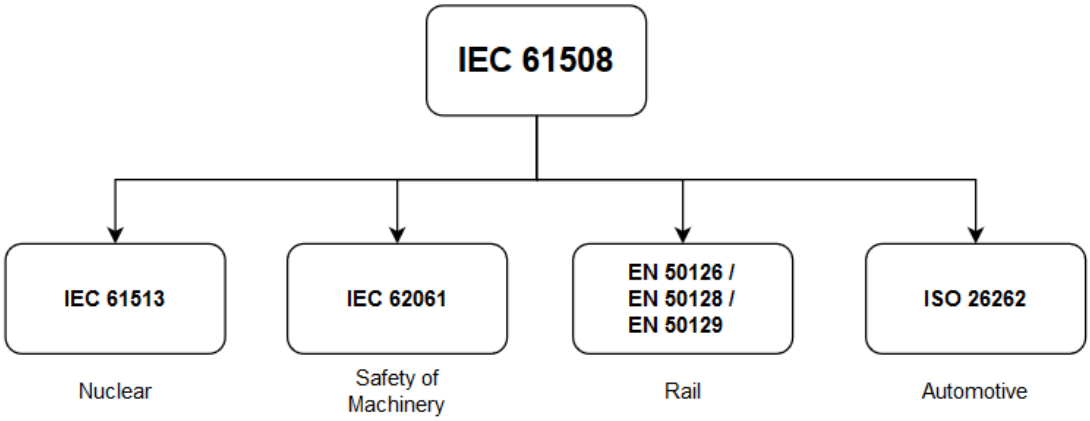

Figure 2.1: Derivation of ISO26262

2.2 V-model

ISO 26262 uses V-model framework to organise product development at the software level [3]. It is model-based software design, because model-based design and ISO 26262 complement each other. This model is called V-model because phases verify each other like in the form of letter V. It is represented in Figure 2.2.

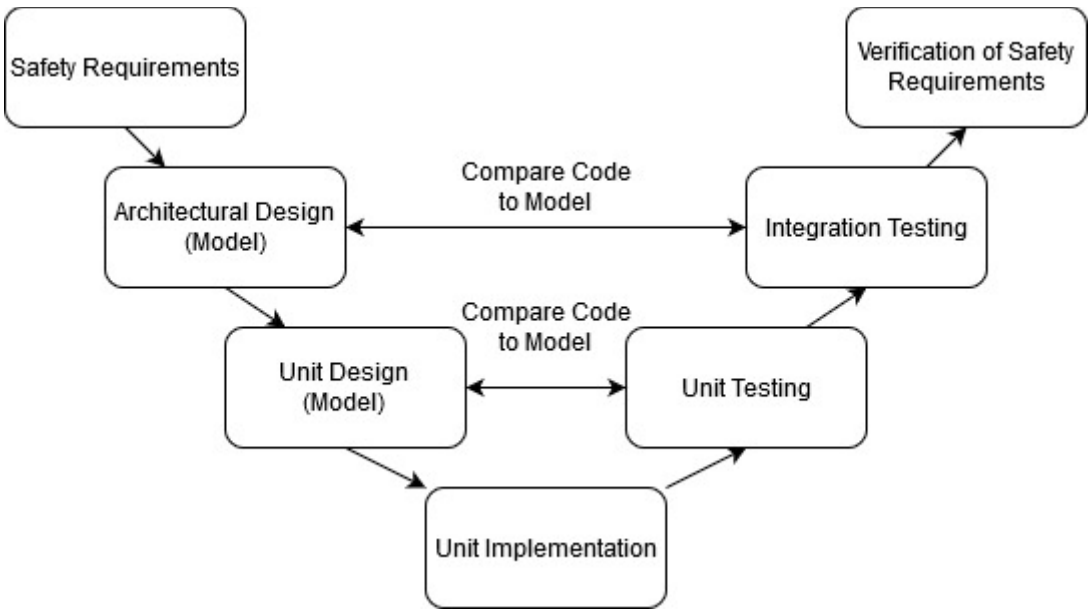

Figure 2.2: V-model

This framework has some phases. First phase has many safety requirements which are determined to overcome unsafe or ineffective handling of hardware or software faults. In next phase, the top architecture, which ensure safety requirements that are determined in the previous phase, is designed for each component. Other phase, software unit subsystems are designed in a similar way. They are passed implementation and doing unit testing phase, respectively. After the unit testing, integration test is applied on whole system to make sure integrated units behavior. In last phase, system are tested with actual environment to verify safety requirements.

2.3 Automotive Safety Integrity Levels

In configuration of the ISO 26262 requirements, Automotive Safety Integrity Levels are used to specify degree of the development [4]. ASILs are based on the probability and acceptability of harm. There are four levels in standard which are A, B, C and D. ASIL-A represents the lowest degree of automotive hazard, while ASIL-D represents the highest degree. For instance, components like rear lights require an ASIL-A grade while systems like airbags, anti-lock brakes require ASIL-D grade.

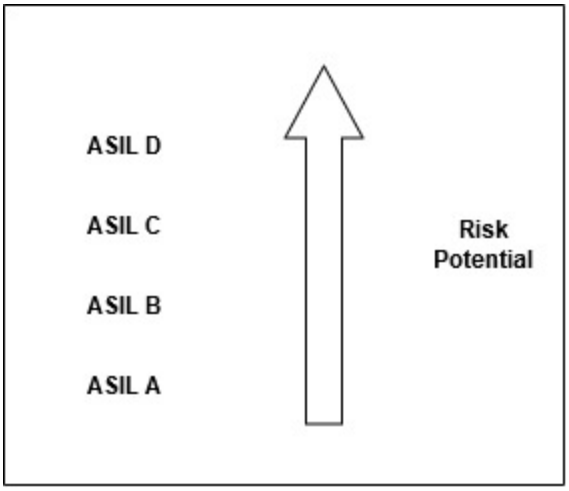

Figure 2.3: ASILs

Failure of airbags have a higher probability of harm according to failure of rear lights. ASIL degree for each electronic component is determined according to severity, exposure and controllability variables. Classifications of ASIL help to achieve the highest safety in vehicles.

2.4 MISRA C

Truly experienced programmer can get rid of making mistakes, easily. However this situation is not valid for inexperienced programmer. On the other hand, Easy access to hardware, low memory requirements and efficient run-time performance are the reasons that doing popular usage of C programming language in the embedded systems. However C have some problems such as highly limited run-time checking, a syntax that is prone to stupid mistakes that are technically legal.

For these reasons, Motor Industry Software Reliability Association (MISRA) established a set of software development guidelines for the use of C programming language in safety-critical systems [5]. This is called MISRA C. It aims to ensure code safety, security, portability and reliability in embedded systems which are used in automotive industry.

Task timing safety must be validated to build ISO 26262 compliant ECU schedules. Timing is a critical performance factor for the reliability and safety in vehicle systems. Timing analysis getting more and more difficult as functionality increases with the new technology. One of safety requirements was tasks that have stable and predictable timing behavior and the amount of computing power needed.

CHAPTER 3

REAL-TIME SYSTEMS

This chapter have detailed discussion which is started from real-time systems to scheduling algorithms. It is started with the differences of real-time system and it is go on with the classification of real-time system. Then, general concept which is related with scheduling is explained. The classification of scheduling algorithms is found in the end of this chapter.

A system is a blackbox that includes a set of one or more inputs and a set of one or more outputs which are related to inputs. Meaning of determinism in the dictionary, all events are determined completely by previously existing causes [6]. A deterministic system is also a system that has always produce the same output from a given input. Predictability is important feature for automotive industry. Therefore, deterministic systems are common used in this industry.

Real-Time system can respond and schedule tasks in a certain time. Let's think popular auto-piloted car. It has cameras and sensors to detect pedestrians. It determines own speed according to incoming pedestrians. An urgent request can occur to slow down the speed of car. This request needs to communicate with CPU, fastly. Real-time system is able to respond exactly in the same time. We can show usage field of system as a spectrum from non-real time to hard real-time like Figure 3.1.

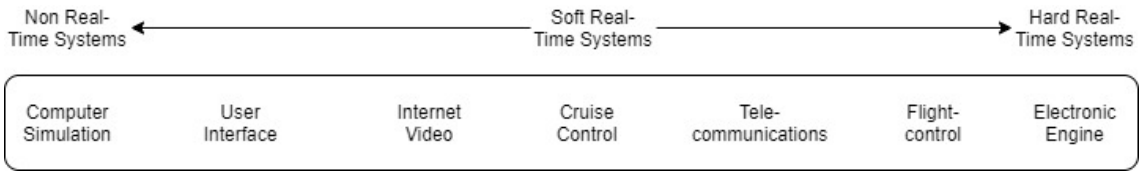

Figure 3.1: Spectrum of usage fields

It was told in thesis before that time is very important element to provide secure vehicle system. Time is the main element that distinguishes systems between real-time

and other type systems. Real-Time systems are special systems that must react within certain time contraint to events [7]. If a reaction occurs too late, it can have very dangerous consequencies. The systems are classified under three different names according to the hazards they cause. These are soft real-time systems, firm real-time systems and hard real-time systems.

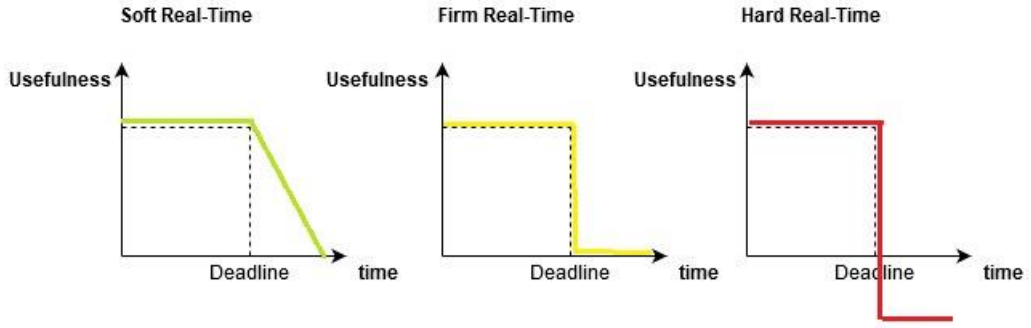

Figure 3.2: Behavior of real-time systems

Soft real-time system is a system that causes only performance degradation when the system gives response after its deadline. Firm real-time system is a intermediate system according to soft and hard real-time system. Infrequent deadline misses can be tolerable in this system. The most strict system is hard real-time system. If it gives response after its deadline, it cause catastrophic consequences in this system. The general comparison between hard and soft real-time system is given in Table 3.1.

Table 3.1. Comparison between hard and soft real-time system

Parameter	**Soft Real-Time System**	**Hard Real-Time System**
Response Time	Soft-Tolerable	Strict-Required
Safety	Non-critical	Critical
Consequences of missing deadline	Tolerable reduction on system quality	Serious damage to system

Hard real-time systems are used in automotive industry to provide ASIL-D grade requirements. These systems have many tasks which has grade from ASIL-A to ASIL-D. Each computer contains a CPU that processes commands inside the software that is

running [8]. This CPU should manage these tasks according to safety grade. For this purpose, CPU use scheduler which has scheduling algorithm to provide task scheduling.

The CPU makes the computer more efficient by switching between processes. It is aimed to run a process in each time interval. The CPU runs other processes in standby situations that will occur in processes. There are many processes in the memory. When a process goes into standby in any way, the CPU switches to another process. Process execution includes CPU execution and Input/Output (I/O) wait cycle. Processes switch between these two states. Processes start working with CPU burst and continue with I/O burst. A CPU burst is performed for calculations. An I / O burst is performed for waiting data transfer between the systems.

Each CPU Scheduler has state machine for task execution states. Sample state machine is represented in Figure 3.3.

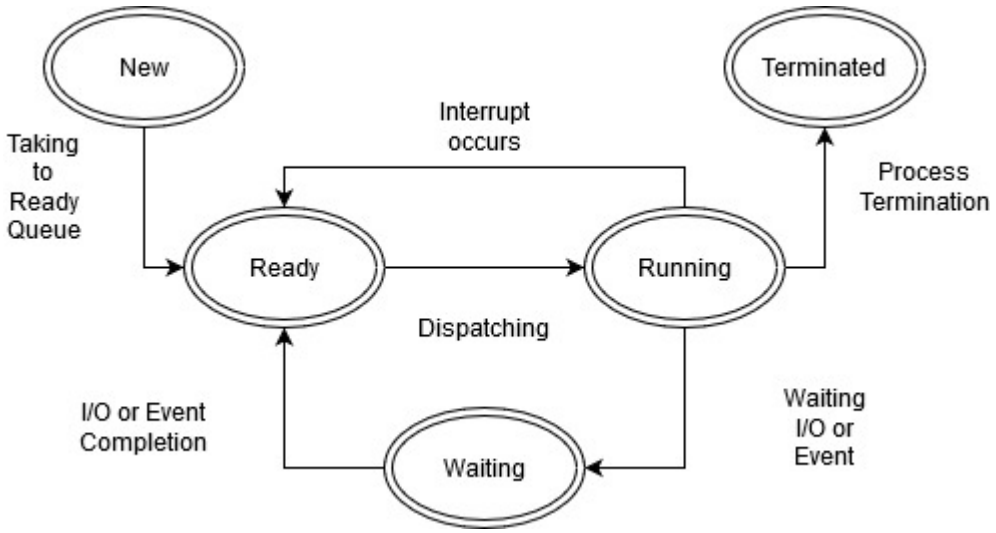

Figure 3.3: Task State Machine

When the CPU enters standby state, the operating system has to select a process to run from its ready queue. This selection is performed by the CPU scheduler. There are two types scheduling to make a decision between states. These are preemptive and nonpreemptive scheduling. In preemptive scheduling, if the CPU is allocated to a process, it can make a decision it when passing from running state to ready state or passing from

waiting state to ready state. In nonpreemptive scheduling, if the CPU is allocated to a process, it keeps it until it is terminated, until it releases the CPU, or until it waits.

Dispatcher is a component that, selects the process to be assigned to the CPU by the CPU scheduler. It is mandatory for Dispatcher to switch very quickly. The transition time between processes is called dispatch latency. CPU scheduling algorithms determine which of the processes waiting in the ready queue will be assigned to the CPU. CPU scheduling algorithms are selected based on many different criteria. These are:

- CPU utilization: The CPU is requested to be in use as much as possible.
- Throughput: The number of processes completed in each time interval.
- Waiting time: The time that a process waits in its ready queue.
- Response time: The time it takes to respond to a request sent to a process.
- Turnaround time: Waiting time for a process to be taken to memory, ready waiting time in the queue is the sum of the time it takes to run on the CPU and process I/O.

It is aimed to maximize CPU utilization and throughput; to minimize waiting time and response time.

Basically, scheduling algorithms are categorized under two main titles. These are offline/static and online/dynamic scheduling [9]. In offline scheduling, there is a schedule table which has list of tasks and their activation times. At runtime, a simple dispatcher executes the decisions which are represented in the table. Round Robin (RR) is a scheduling algorithm which is used for offline scheduling. In online scheduling, there is a set of predefined rules. At runtime, dispatcher makes a decision based on these predefined rules to a given task set.

Online scheduling can be divided into two parts, these are priority driven and quality driven [10]. Priority driven is common part in hard real-time systems [21]. A fixed-priority algorithm assigns the same priority to all the jobs in each task. On the other hand, dynamic-priority algorithm assigns different priorities to the individual jobs in each task. RM and EDF are popular scheduling algorithms for fixed priority and dynamic priority scheduling, respectively.

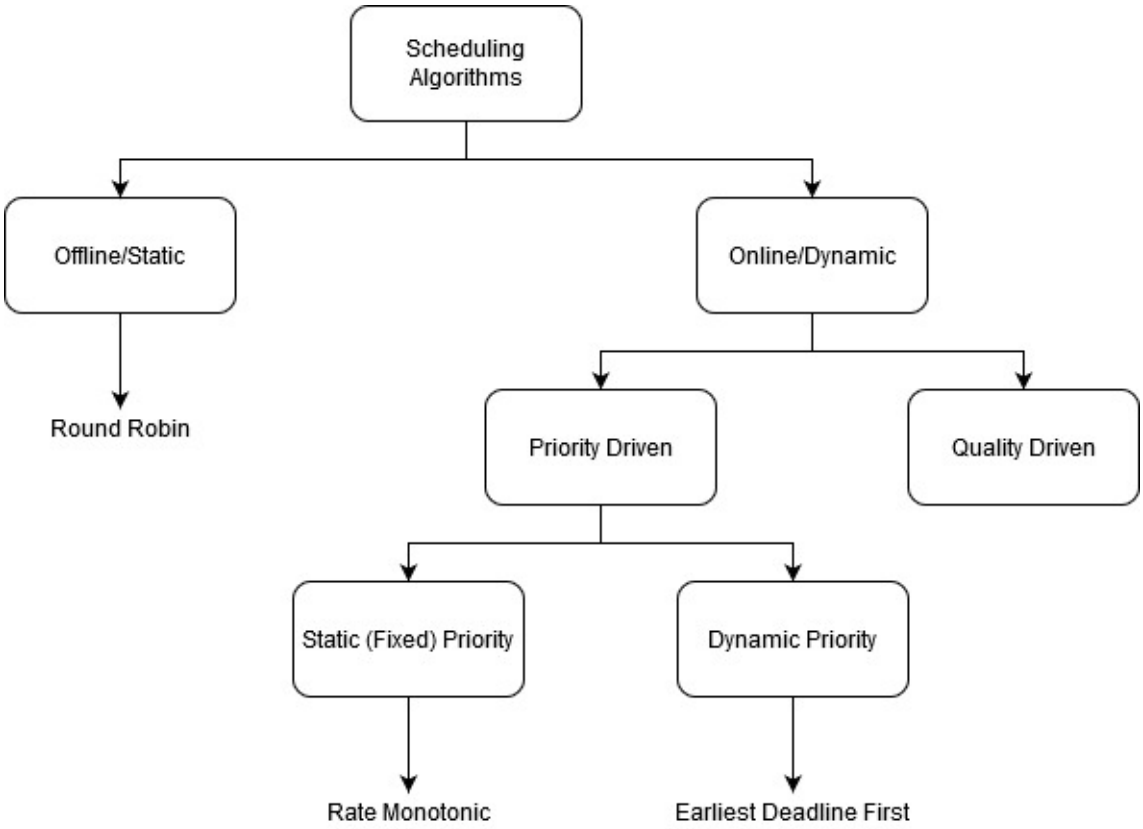

Figure 3.4: Classification of Scheduling Algorithms

CHAPTER 4

SCHEDULING ALGORITHMS

The classification of scheduling algorihms is shown in previous chapter. These scheduling algorithms and their specifications will be given in this chapter. There are two main sections when we look to Figure 3.4. One of them is dynamic side. It is dynamic because rules are processed in run time according to event's attributes. We can call event triggered scheduling architecture for this section. Another is static side. It is static because WCET of tasks are determined in compile time and there are no changes in run time. We can call time triggered scheduling architecture for that part.

4.1 Event Triggered Scheduling

The event-triggered approach is that an event must be triggered by another event to initiate. The main difference between event-triggered and time-triggered system is that event-triggered systems decide at runtime which task is released at the current time instant. Pessimistic assumptions are made to always ensure the correct functionality of the system. This gives reason to an overestimation of resource requirements compared to an event-triggered system. However, timing analysis methods are also applied for event-triggered systems to guarantee the correct system behavior also for worst-case scenarios. Timing analysis methods are also based on worst-case assumptions and a periodic execution. Thus, event-triggered systems use resources more efficiently than time-triggered systems, but estimated resource requirements are generally pessimistic to fulfill hard real-time requirements. Consequently, for safety-critical systems, event-triggered systems are not used to a higher system utilization. However, event-triggered systems are able to react faster than a time-triggered system to events for functionality.

4.1.1 Rate Monotonic

The RM scheduling algorithm uses pre-emptive scheduling with static priorities [11]. The rule is the priorities of its tasks are determined at compile time. Their priorities do not change over time. Their priorities are directly proportional with their period. Task, which has the shortest period, has the highest priority. There is deterministic behavior on overloads. So tasks are affected by priority level. Therefore, RM is closer to static scheduling than EDF. If needs to calculate upper bound utilization for RM, needing to benefit from Lui and Layland proofs [12][13]. A task is a three-tuple, (Ci, Ti, n), where Ci is the execution time for the task, Ti is the task's period, and n is the total number of task which is defined. The formula is given in Figure 4.1.

$$U = \sum_{i=1}^{n} \frac{Ci}{Ti} < n(2^{1/n} - 1)$$

Figure 4.1: The formula of utilization calculation

According to this formula, U tends to ln(2) about 0.69 for n tending to infinity. It means that is RM can guarantee a feasible schedule only if the total load is no greater than 69 percent [6]. Calculation steps are given in Figure 4.2.

$$\lim_{n\to\infty}\left(n\left(2^{\frac{1}{n}}-1\right)\right) =$$

$$= \lim_{n\to\infty}\left(\frac{-1+2^{\frac{1}{n}}}{\frac{1}{n}}\right)$$

$$= \lim_{n\to\infty}\left(\frac{-\frac{\ln(2)\cdot 2^{\frac{1}{n}}}{n^2}}{-\frac{1}{n^2}}\right)$$

$$= \lim_{n\to\infty}\left(\ln(2)\cdot 2^{\frac{1}{n}}\right)$$

$$= \ln(2)\cdot\lim_{n\to\infty}\left(2^{\frac{1}{n}}\right)$$

$$= \ln(2)\cdot\lim_{n\to\infty}\left(e^{\frac{1}{n}\ln(2)}\right)$$

$$= \ln(2)\cdot 1$$

$$= \ln(2) \quad = \quad 0.69314\ldots$$

Figure 4.2: The calculation steps of RM feasibility

4.1.2 Earliest Deadline First

Earliest Deadline First is a dynamic priority scheduling algorithm. The rule of this algorithm is it selects the task for execution according to their absolute deadlines. The ready queue is sorted according with decreasing priorities whenever there is a priority change. Therefore, this algorithm has higher overhead. It has complex implementation, so that requires kernel support. In spite of that it can scales well and also it can accommodate sporadic tasks easily. According to proof of Lui and Layland [12][13], EDF can exploit full processor capacity, scheduling task sets with utilization up to 1.

4.1.3 Comparison between RM and EDF

There was clarification of scheduling algorithms in previous chapter. When we look priority driven scheduling algorithms under the online scheduling, there is an separation as static priority and dynamic priority. RM and EDF are popular scheduling algorithms which are represented as static and dynamic priority, respectively. The general comparison between them is given in Table 4.1.

Table 4.1. Comparison between RM and EDF

Parameters	**RM**	**EDF**
Priority	Static	Dynamic
Implementation Effort	Low	High
Preemption	Preemptive	Preemptive
Criteria of Rule	Period	Deadline
CPU Utilization	Less (~0.69)	Full

4.2 Time Triggered Scheduling

Time Triggered Architecture can be defined an architecture that based on executes sets of tasks according to schedule table which is predefined by system creator. This architecture has a timer to interrupt tasks and schedule them with the process of task state machine which is mentioned in Figure 3.3. This timer creates a periodic world for tasks which are defined in schedule table.

In time triggered scheduling, tasks initiated at predefined points in time. Run-time dispatching is performed according to a set of rules. These rules are defined at schedule table which is prepared at compile time. The dispatcher takes scheduling decisions according to this table. Therefore, there is lower overhead in runtime. This also brings disadvantages like there is no flexibility.

For this scheduling, everything should be known before runtime [14][15]. This is also cost. However it provides determinism which is important qualification for

developers. They can plan everything should be when. Problems can be detected and can be solved, easily. Because tasks are defined for exact time by developers. This make an easier to maintenance process. Besides, it reduces testing effort, because tester can reproduce system behaviour easily.

If there is needing a comparison for the suitability between time triggered and event triggered architecture, it depends on the function requirement. Event-triggered systems are clearly better suited to efficiently use the available resources and react to the user input for non-critical functionality in the comfort domain which is generally highly asynchronous. On the other hand, the deterministic behavior of time-triggered systems helps to reduce the design and integration effort for critical applications.

Creating a time triggered system is not complex. In these systems, events are generated by clock. Events have arrival time and scheduler takes these events to running state when their arrival time comes in a hyperperiod at clock. Figure 4.3 is an example represantation for time triggered scheduling.

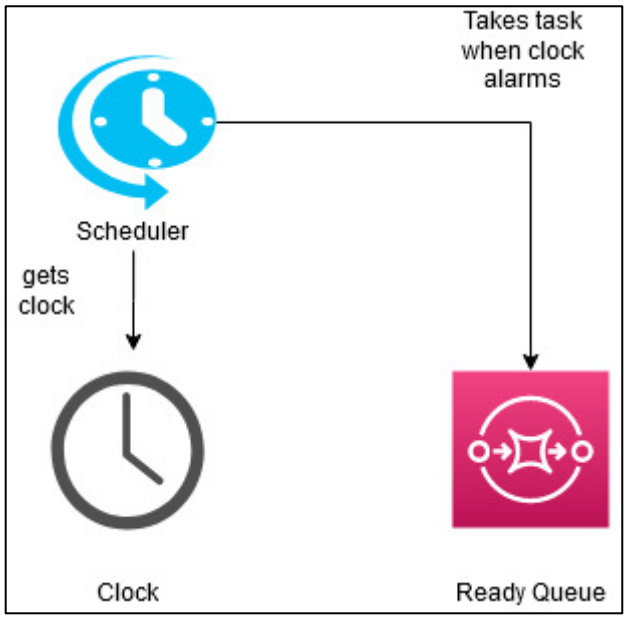

Figure 4.3: Representation of TT Scheduling

The system holds a table which is called schedule table. Scheduler gets clock and goes table to check whether has a task should it be taken to ready queue. If there is a task, scheduler takes task to running state.

CHAPTER 5

HARD REAL-TIME OPERATING SYSTEMS

Hard real-time systems were explained until this chapter. Operating systems, which are used in these systems, will be discussed in this chapter. A classification table will be put to this chapter to see difference between them, easily. In the end of this chapter, which OS is why preferred will be told.

Operating system standards can provide portability of application from one platform to another one [16][20]. Besides, they can allow the possibility of having several kernel providers for a single application. Therefore, these standards are very important. They can be classified according to where it is used. The classification of OS are shown in Table 5.1.

Table 5.1. The classification of popular hard real-time operating systems

Name	**Company**	**Description**	**Free**
VxWorks	Wind River Systems	VxWorks is certified to be compliant with ISO 26262 Automotive Safety Integrity Level D (ASIL-D) and IEC 61508 (SIL 3).	No
Microsar	Vector	The AUTOSAR 4 basic software from Vector fulfills the requirements of ISO 26262 up to the ASIL-D level.	No
Integrity RTOS	Green Hills Software	INTEGRITY real-time operating system has availability of a compatible AUTOSAR Application Programming Interface.	No
Erika OS	Evidence	Erika Enterprise is a royalty free automotive OSEK/VDX certified Hard Real-Time Operating System (RTOS)	Yes

Trampoline OS	-	Trampoline is a static RTOS for small embedded systems. Its API is aligned with OSEK/VDX OS and AUTOSAR OS 4.2 standards.	Yes
RTLinux	Wind River Systems and FSMLabs	RTLinux is a hard realtime real-time operating system (RTOS) microkernel that runs the entire Linux operating system as a fully preemptive process.	Yes

Real Time-Portable Operating System Interface (RT-POSIX) is popular one that is real time extension of main general-purpose operating system standard. APEX is common in avionics systems.

Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen (OSEK) is a standard that was founded by a German automotive company consortium for automotive embedded systems [17]. It has produced specifications about a communications stack, a network management protocol and other related topics. It was designed as standard software architecture for ECUs. French car manufactures had a similar project called Vehicle Distributed eXecutive (VDX). The official name has been OSEK/VDX, after it joined the consortium. The founders of OSEK/VDX are given in Figure 5.1.

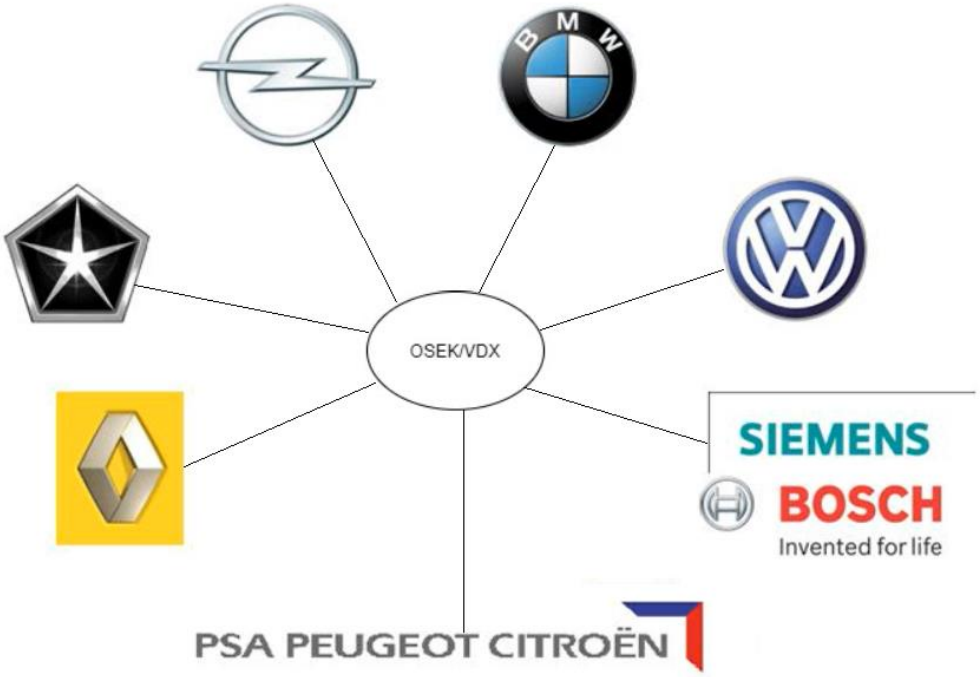

Figure 5.1: The founders of OSEK/VDX

There are many Operating System (OS), but they have some disadvantages from an automotive perspective. They are quite bulky and their real time performance is not quite real enough. Their Application Programming Interfaces (APIs) are not appropriate to use synchronous and asynchronous task scheduling together. OSEK API was specially designed for automotive industry. For these reasons, OSEK OS is used on automotive embedded systems.

In addition to these OSEK/VDX has many advantages. It supports portability and reusability of the application software because of it has application-independent architecture. Its specification of interfaces are independent of hardware and network. This advantage provides usage of an application at different ECUs. With its architecture was designed efficiently, available functionalities are configurable and scalable. Besides, new functionalities can be added to the system, easily.

Standard compliant means that it supports the specifications of standard. Erika Enterprise is an open-source OSEK/VDX Hard Real-Time Operating System (RTOS)

[18]. It implements an API inspired to a subset of the AUTomotive Open System ARchitecture (AUTOSAR) API. Erika was certified in 2012 and then AUTOSAR OS specifications are implemented to Erika. It is the first open-source Free RTOS that has been certified OSEK/VDX compliant. It supports event triggered architecture and time triggered architecture. This OS is suitable for 1-4 Kb Flash footprint and 8 to 32 bit microcontrollers. Besides, it has conformance classes that are used to limit footprint. One of its best advantages, it can be configured with an OIL file or AUTOSAR Extensible Markup Language (XML), statically.

CHAPTER 6

IMPLEMENTATION AND EVALUATION

In this chapter, a scenario is determined according to calculations and it is implemented to Arduino Uno with OSEK/VDX certified Erika OS. The findings are visualised and evaluated for each scheduling algorithms. Pros and cons of each scheduling algorithms are discussed in the end of each evaluation.

Uniprocessor system is a good choice to evaluate scheduling algorithms. Context switch between tasks, missing deadline and CPU utilization for each task can be seen clearly. So that, Arduino Uno board is selected for implementation. Supporting of Erika OS for this board is a good advantage.

6.1 The Board Specifications

Arduino Uno is a simple board which use Atmega328 processor [19]. There are 14 digital I/O output pins, 6 of which can be used as Pulse-Width Modulation (PWM) output. It has 16 MHz crystal oscillator, Universal Serial Bus (USB) connection, 2.1mm power input, In-Circuit Serial Programming (ICSP) header and reset button. It is enough to connect to DC 7 ~ 12V power supply to operate. Pin diagram [19] is given in Figure 6.1.

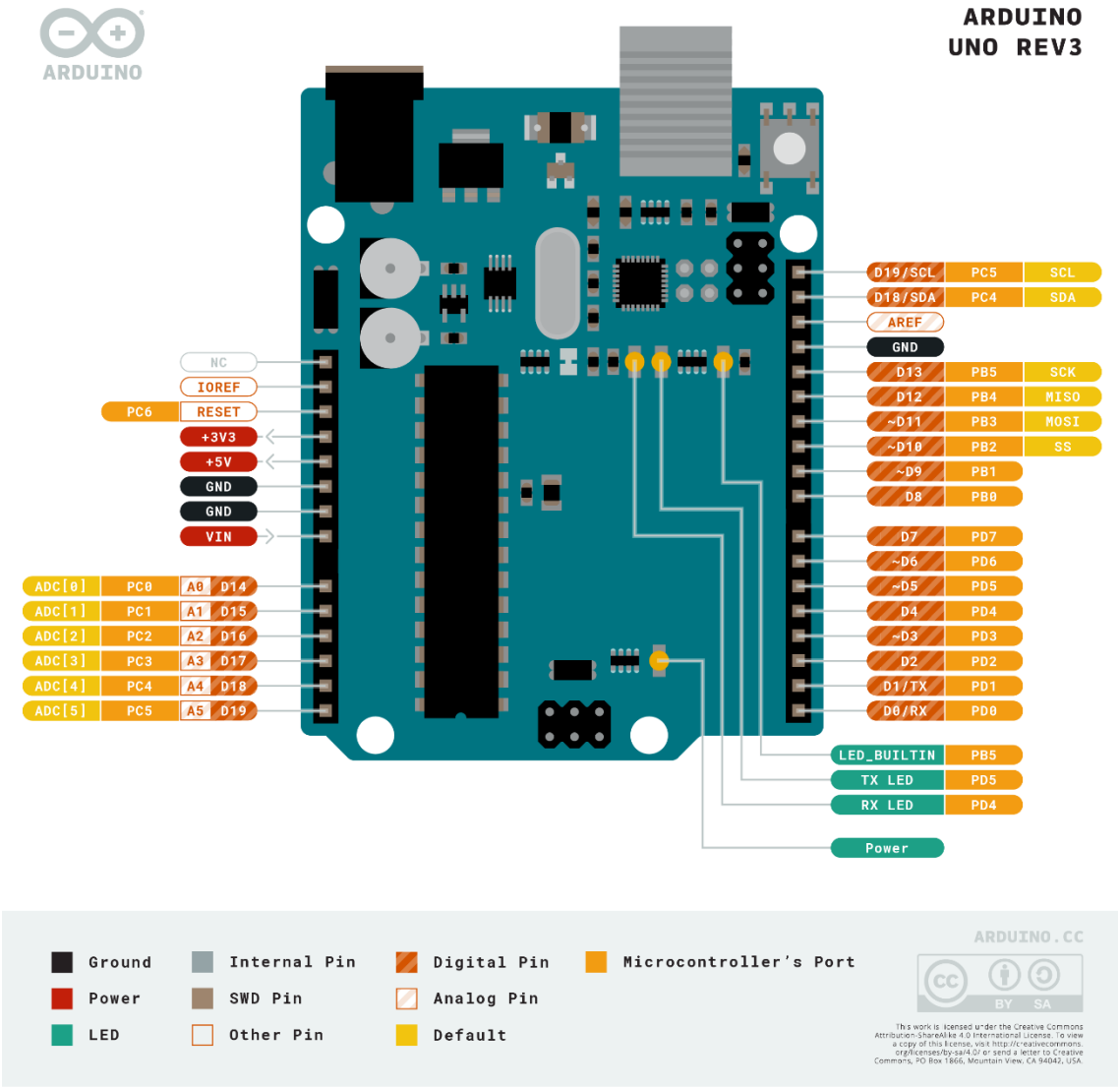

Figure 6.1: Pin diagram of Arduino Uno Board

The board needs some configuration. It connects USB port to personal computer like Figure 6.2.

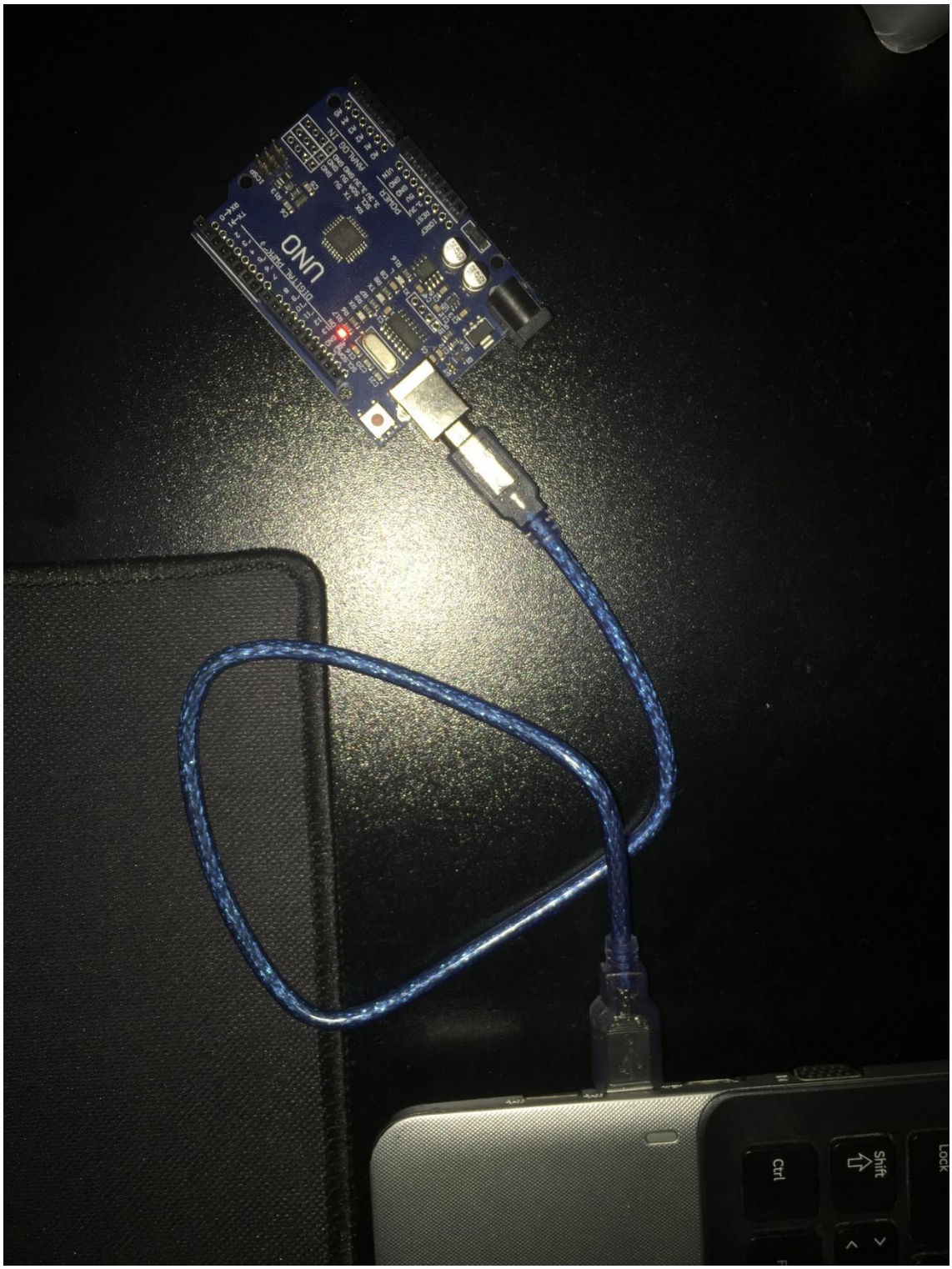

Figure 6.2: The connection of board

It is shown in device manager. The port information of board can be accessed there. Ubuntu users should give permission to this port with “sudo chmod a+rw /dev/<portname>” command.

6.2 Implementation

Erika OS was downloaded from its official website [18]. There is package content which is shown in Figure 6.3.

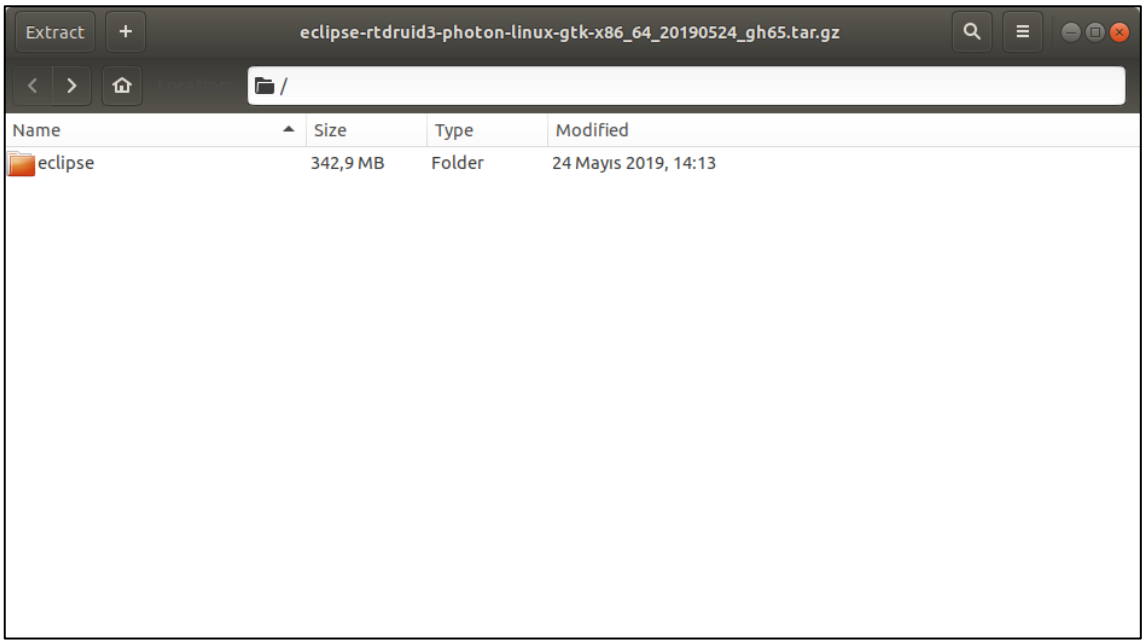

Figure 6.3: Erika downloading package content

When eclipse application is opened, a workspace should be chosen. Default RT-Druid Eclipse IDE C/C++ perspective should be selected from top right-hand corner to open to use RT-Druid Interface. RT-Druid v3 Oil and C/C++ Project can be used from File>New menu to open an example project. Light Emitting Diode (LED) blink example is selected and then it was tested. In previous step, actually related toolchain should be chosen. This step depends on user toolchain. Some paths should be defined either from Window>Preferences of eclipse menu or makefile of the project. The connection of board should be checked from device manager. There is a serial port to communicate with

Arduino board, when board is connected to personal computer. These paths and port are defined as Figure 6.4.

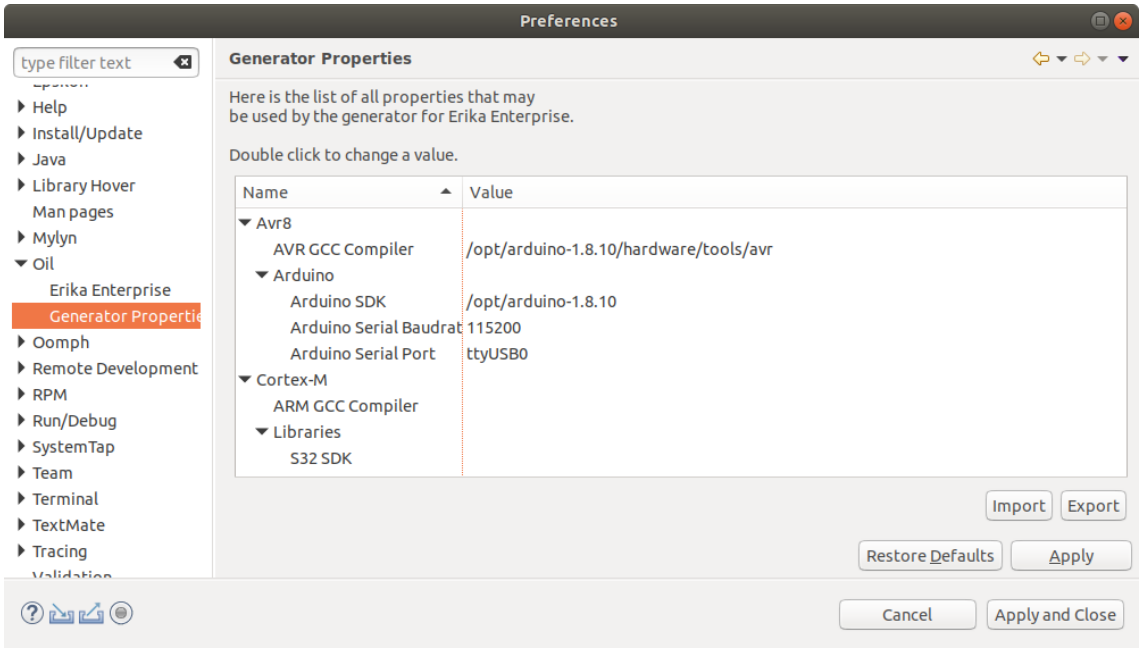

Figure 6.4: Configuration for Arduino Uno

When the project build with right-click on the project into Eclipse Project Explorer panel, successfully message should be seen like Figure 6.5.

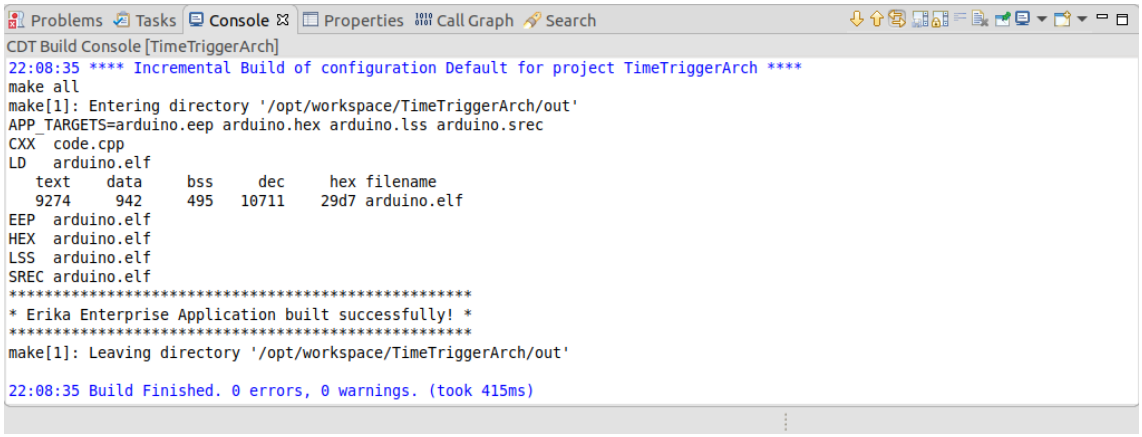

Figure 6.5: Successful build console output

Build target creation should be done for embed being built to board. This step can be done like Figure 6.6 according to related toolchain.

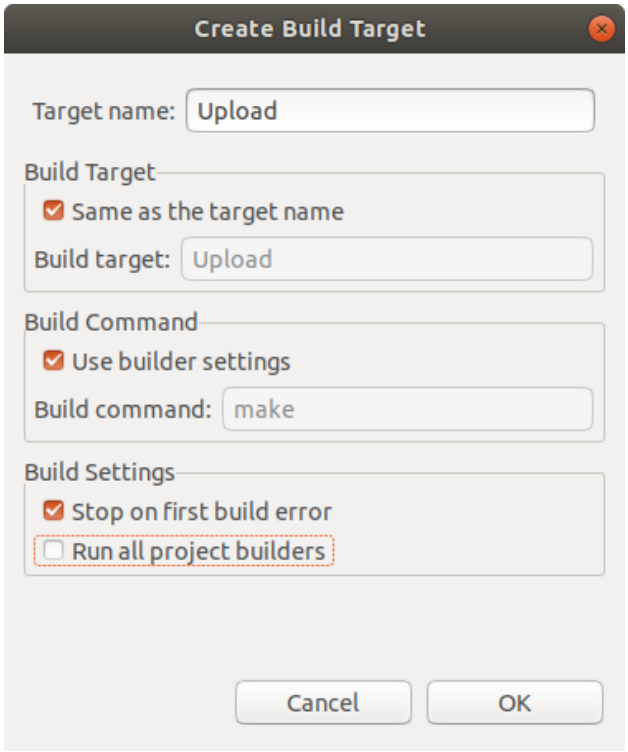

Figure 6.6: Creating upload target

When the upload process end successfully, the console output should be like Figure 6.7.

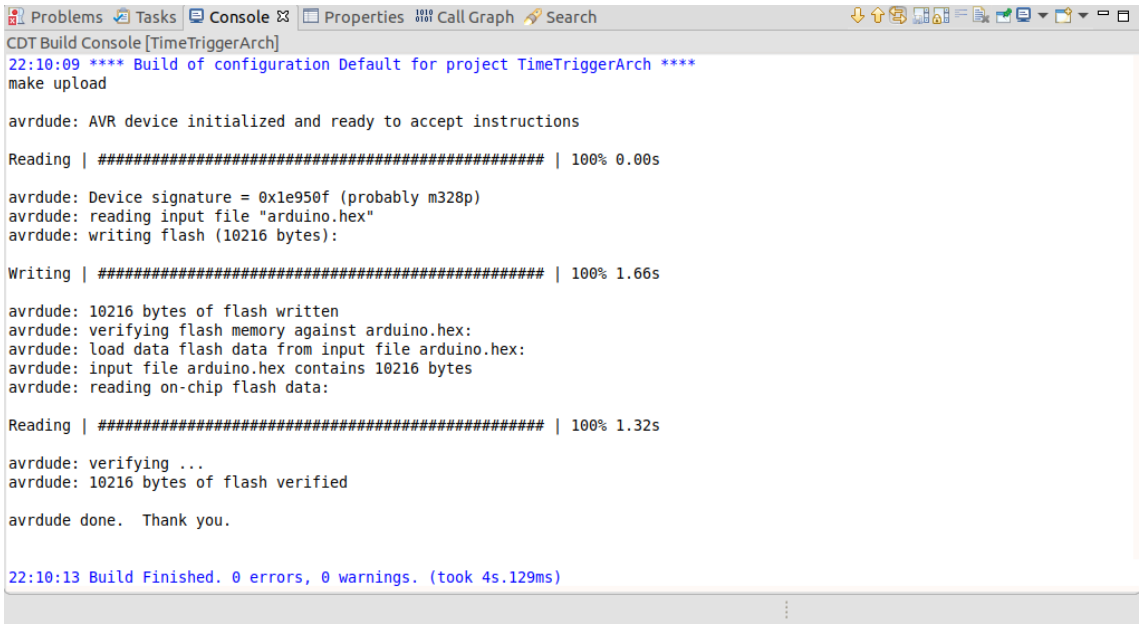

Figure 6.7: Successful flashing console output

A scenario was determined to show pros and cons of scheduling algorithms after Erika OS is built and upload successfully. This scenario has 3 tasks which have fixed priorities. Their specifications are shown on Table 6.1.

Table 6.1. The scenario parameters to be used

Task	**Execution Time (s)**	**Period (s)**
Task1	1	4
Task2	2	6
Task3	3	8

The CPU Utilization of these tasks is calculated according to formula. The calculation is shown in Figure 6.8.

$$Cpu\ Utilization = \frac{1}{4} + \frac{2}{6} + \frac{3}{8} = 0.96$$

Figure 6.8: The utilization calculation of the scenario

6.3 Evaluation

In evaluation section, OIL file was prepared to configure OS. This file was written according to desired and necessary configuration. Some configurations would be changed for each scheduling algorithms, easily. RM was first scheduling algorithm to analyse. It is a FP algorithm. OIL file was configured as Figure 6.9.

Figure 6.9: OIL File Configuration for RM

Contents of these tasks were adjusted according to their execution times like ECU application at vehicles. Universal Asynchronous Receiver Transmitter (UART) logs with timestamp were added to tasks for showing execution times of tasks. The UART logs for RM are found in Figure 6.10.

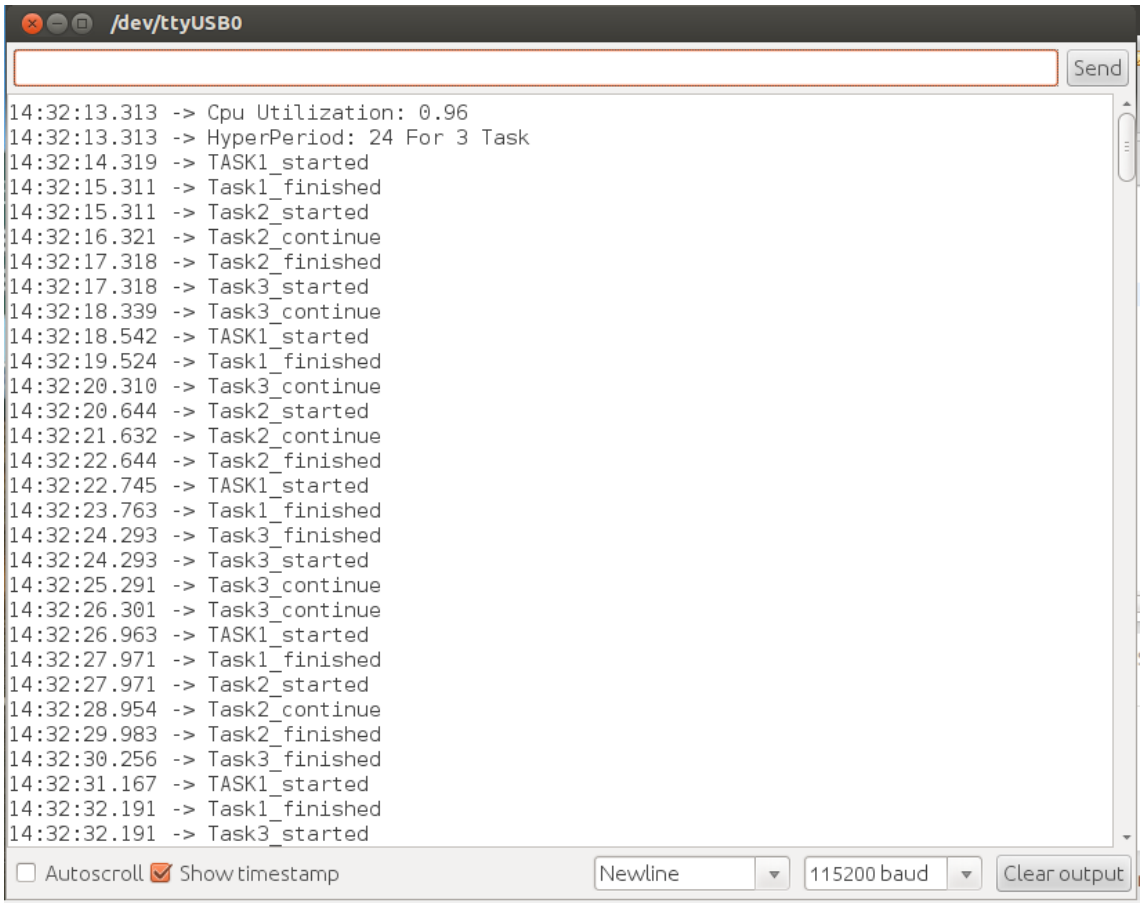

Figure 6.10: The UART output of RM

Resulting behaviour of this scenario was visualised to show pros and cons of RM scheduling algorithm. It is found in Figure 6.11.

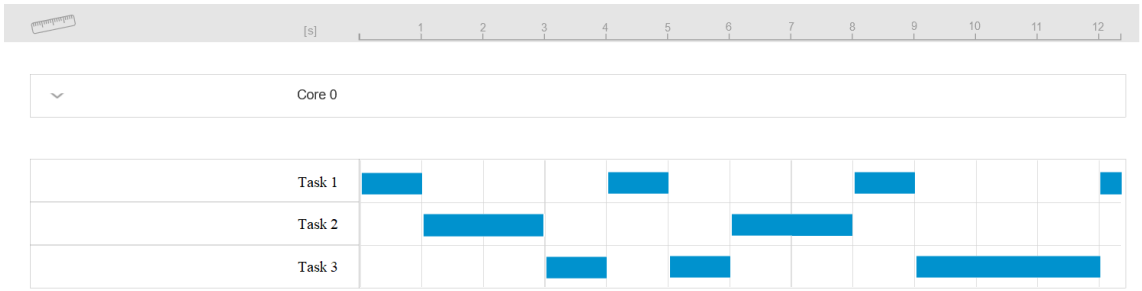

Figure 6.11: The visualization of RM output

When looking to Figure 6.11, there is missing deadline for Task 3. Task 3 has the lowest priority and its execution time was 3s. It should be completed at time 8s. This is an disadvantage for RM, however it missed deadline for the lowest priority task. This task can be ASIL-A like a task to showing door status with opens interior light. The latency of this task can not cause catastrophic consequences in this system. This missing deadline can be eliminated with decreasing CPU Utilization [22]. This analysis shows us that RM, which is a FP scheduling algorithm, can be used to create a safe system by renouncing performance.

The next scheduling algorithm is EDF which is a Dynamic Priority Scheduling Algorithm. It needs less effort than FP, because it automatically schedules to tasks according to their deadlines at runtime [23]. OIL file is rearranged for EDF. The KERNEL_TYPE variable is set as EDF. This is shown in Figure 6.12.

Figure 6.12: OIL File Configuration for EDF

This time, same scenario is run for EDF. The UART logs for EDF are found in Figure 6.13.

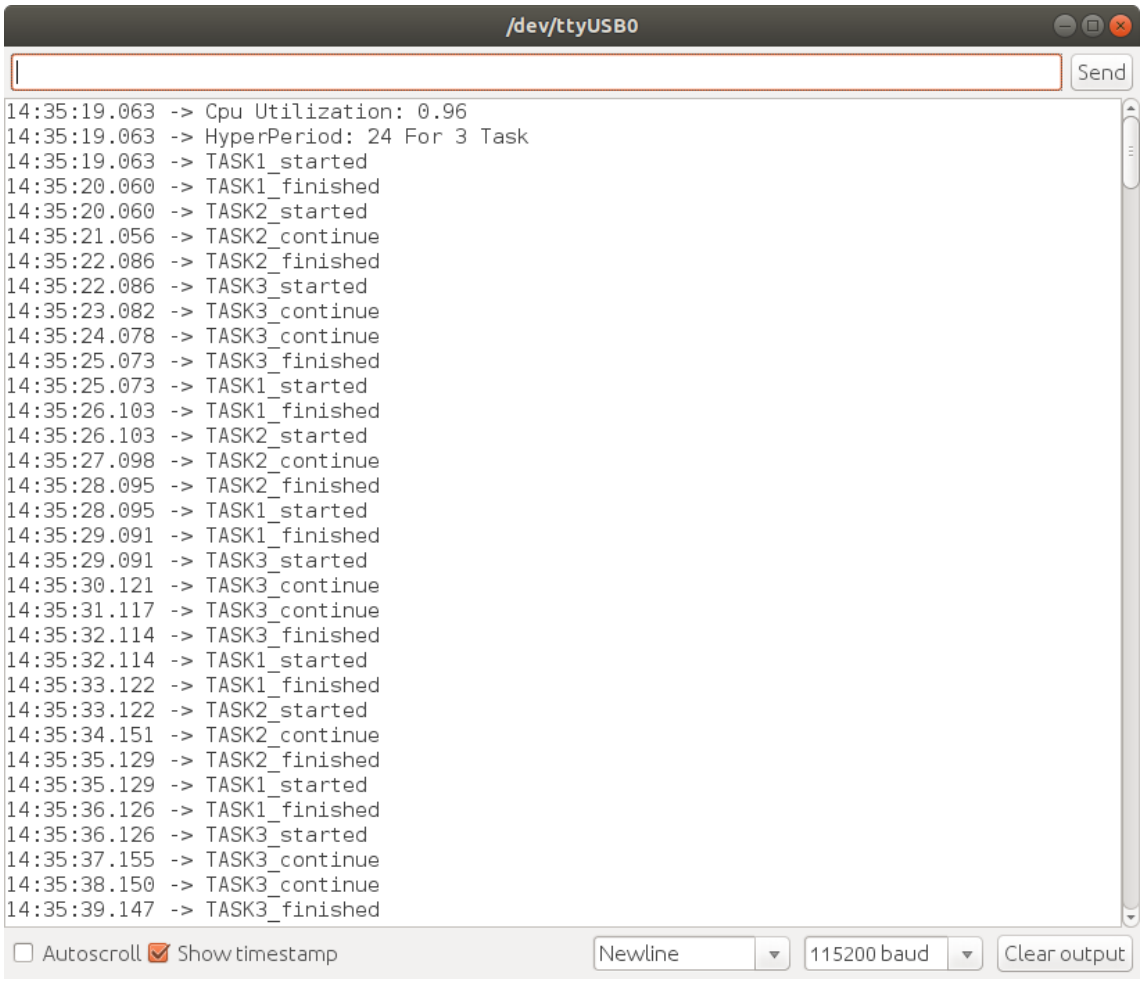

Figure 6.13: The UART output of EDF

Resulting behavior of this scenario was visualised to show pros and cons of EDF scheduling algorithm. It is found in Figure 6.14.

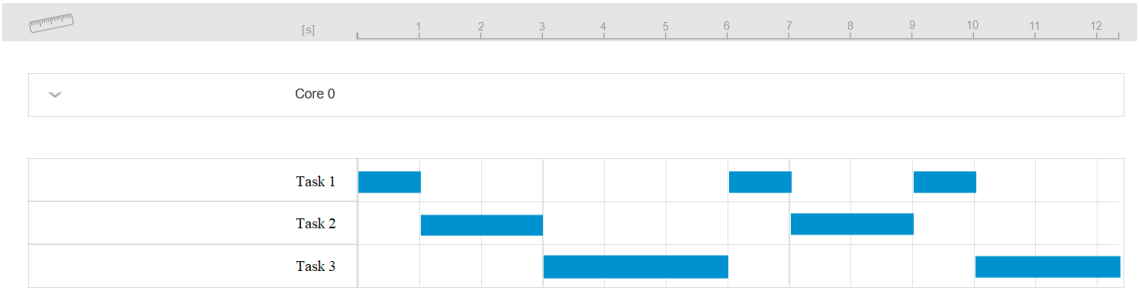

Figure 6.14: The visualization of EDF output

When looking to Figure 6.14, there is no missing deadline. This shows us that EDF can provide CPU Utilization higher than FP. However this does not show that EDF has no disadvantages. EDF guarantees the fulfillment deadline for each task, but it is not possible to provide a minimum response time for any given task. The highest priority task always has a minimum response time in RM, but it is not possible to guarantee for EDF. If the system becomes overloaded because of adding a new task or a wrong WCET estimation, domino effect can be occurred. Domino effect is that all other tasks miss their deadline after one task missed its deadline with executing more than its declared runtime. This is possibility of missing deadline of ASIL-D task like a task which opens airbags at crash time. This condition causes catastrophic consequences in this system. Therefore, EDF can be used for tasks which need high performance and low safety.

These were pros and cons of online scheduling algorithms. Automotive industry does not have only two parameters as performance and safety. Determinism and predictability are also other important parameters for developers which work in automotive industry. Therefore, time triggered scheduling architecture is important in this industry [15][24]. This architecture has schedule table which includes offline defined tasks and their triggering times. This schedule table was prepared in OIL file and each task calls at a certain time. The UART logs for TT are found in Figure 6.15.

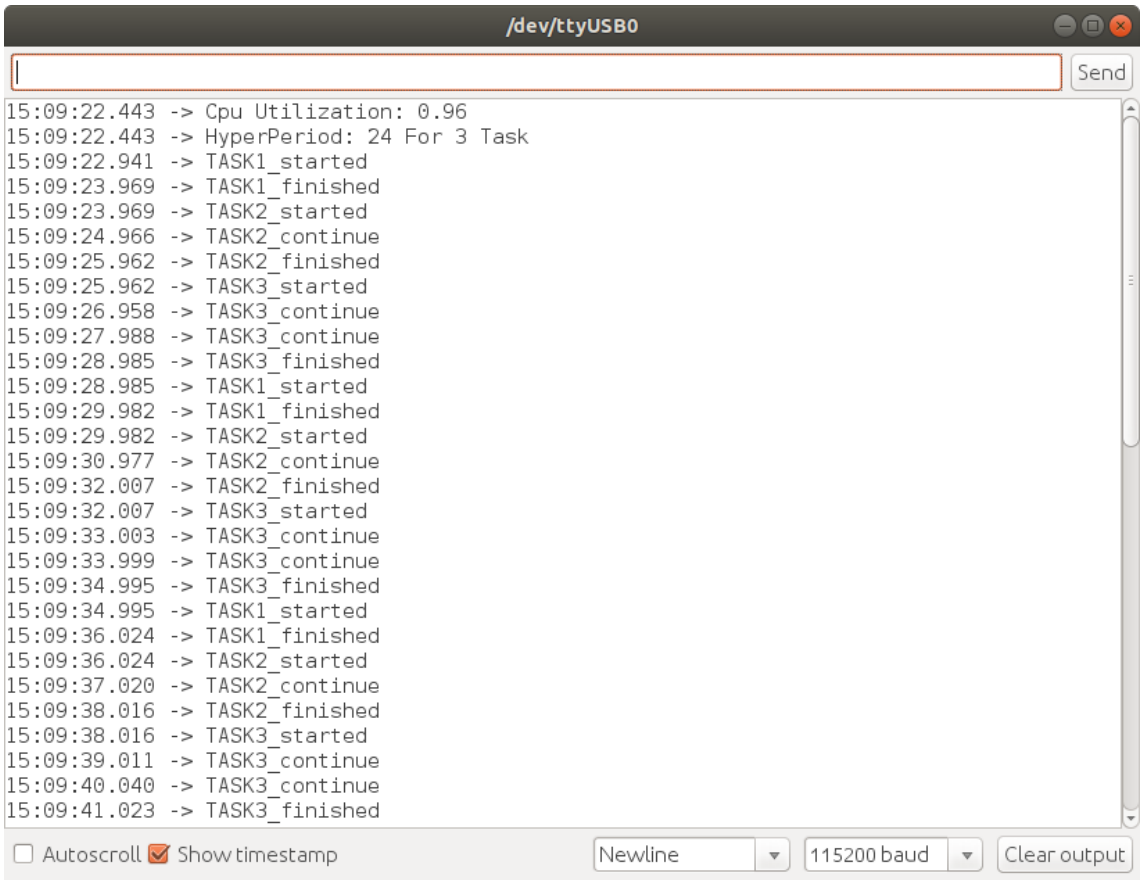

Figure 6.15: The UART output of TT Scheduling

Resulting behaviour of this scenario was visualised to show pros and cons of TT scheduling algorithm. It is found in Figure 6.16.

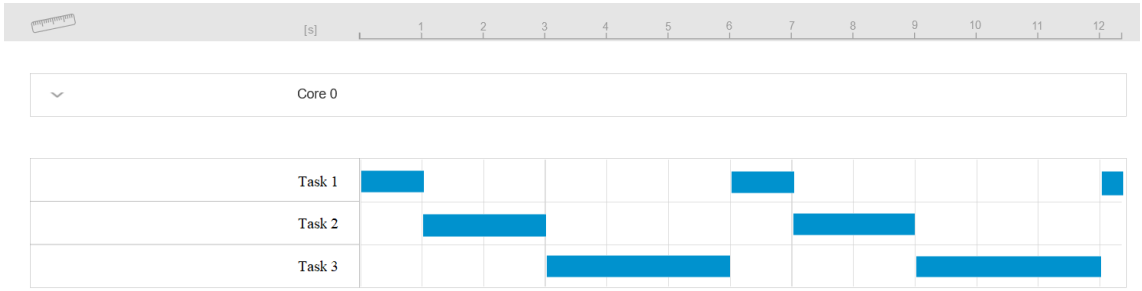

Figure 6.16: The visualization of TT Scheduling output

Similar operation was done at different CPU Utilization with more tasks. The outputs were listed in Table 6.2.

Table 6.2. The testing outputs for different CPU Utilization with more tasks

	CPU Utilization (%)	Tasks (Number of tasks)	Unsuccessful Tasks (Having Deadline Missing)	Successful Tasks	Success Rate (%)
RM	<%85	5	0	5	%100
EDF			0	5	%100
Time Triggered			0	5	%100
RM		10	0	10	%100
EDF			0	10	%100
Time Triggered			0	10	%100
RM		15	0	15	%100
EDF			0	15	%100
Time Triggered			0	15	%100
RM		20	0	20	%100
EDF			0	20	%100
Time Triggered			0	20	%100
RM		25	0	25	%100
EDF			0	25	%100
Time Triggered			0	25	%100
RM		30	0	30	%100
EDF			0	30	%100

Time Triggered			0	30	%100
RM	%85-%87	5	0	5	%100
EDF			0	5	%100
Time Triggered			0	5	%100
RM		10	1	9	%90
EDF			0	10	%100
Time Triggered			0	10	%100
RM		15	1	14	%93.3
EDF			0	15	%100
Time Triggered			0	15	%100
RM		20	1	19	%95
EDF			0	20	%100
Time Triggered			0	20	%100
RM		25	1	24	%96
EDF			0	25	%100
Time Triggered			0	25	%100
RM		30	1	29	%96.7
EDF			0	30	%100
Time Triggered			0	30	%100
RM		5	0	5	%100
EDF			0	5	%100
Time Triggered			0	5	%100
RM		10	1	9	%90
EDF			0	10	%100

Time Triggered	%87-%90		0	10	%100
RM		15	2	13	%86.7
EDF			0	15	%100
Time Triggered			0	15	%100
RM		20	1	19	%95
EDF			0	20	%100
Time Triggered			0	20	%100
RM		25	2	23	%92
EDF			0	25	%100
Time Triggered			0	25	%100
RM		30	2	28	%93.3
EDF			0	30	%100
Time Triggered			0	30	%100
RM	%90-%94	5	0	5	%100
EDF			0	5	%100
Time Triggered			0	5	%100
RM		10	1	9	%90
EDF			0	10	%100
Time Triggered			0	10	%100
RM		15	3	12	%80
EDF			0	15	%100
Time Triggered			0	15	%100
RM		20	2	18	%90
EDF			0	20	%100

Time Triggered			0	20	%100
RM		25	3	22	%88
EDF			0	25	%100
Time Triggered			0	25	%100
RM		30	3	27	%90
EDF			0	30	%100
Time Triggered			0	30	%100
RM	%94-%96	5	1	4	%80
EDF			0	5	%100
Time Triggered			0	5	%100
RM		10	2	8	%80
EDF			0	10	%100
Time Triggered			0	10	%100
RM		15	4	11	%73.3
EDF			0	15	%100
Time Triggered			0	15	%100
RM		20	3	17	%85
EDF			0	20	%100
Time Triggered			0	20	%100
RM		25	4	21	%84
EDF			0	25	%100
Time Triggered			0	25	%100
RM		30	4	26	%86.7
EDF			0	30	%100

Time Triggered			0	30	%100
RM	%96-%98	5	1	4	%80
EDF			0	5	%100
Time Triggered			0	5	%100
RM		10	3	7	%70
EDF			0	10	%100
Time Triggered			0	10	%100
RM		15	4	11	%73.3
EDF			0	15	%100
Time Triggered			0	15	%100
RM		20	4	16	%80
EDF			0	20	%100
Time Triggered			0	20	%100
RM		25	4	21	%84
EDF			0	25	%100
Time Triggered			0	25	%100
RM		30	5	25	%83.3
EDF			0	30	%100
Time Triggered			0	30	%100
RM		5	1	4	%80
EDF			0	5	%100
Time Triggered			0	5	%100
RM		10	3	7	%70
EDF			0	10	%100

Time Triggered	%98-%100		0	10	%100
RM		15	4	11	%73.3
EDF			0	15	%100
Time Triggered			0	15	%100
RM		20	4	16	%80
EDF			0	20	%100
Time Triggered			0	20	%100
RM		25	5	20	%80
EDF			0	25	%100
Time Triggered			0	25	%100
RM		30	5	25	%83.3
EDF			0	30	%100
Time Triggered			0	30	%100

As in previous experimental result, RM is the first scheduling algorithm which its success rate of started to decrease when CPU utilization is increased. According to results, the boundary of having deadline missing is decrease when the number of task is increased. This also proves the formula which is given in Figure 4.1. EDF is more successful algorithm than RM at higher CPU utilization as event triggered scheduling algorithm. As we talked before, Time Triggered has some disadvantages, however it is most successful scheduling architecture in periodic world.

If needs a comparison between event triggered scheduling architecture and time triggered scheduling architecture, event triggered is flexible and has characterized by high resource utilization [24][25]. However, it is not reliable from the determinism and predictability point of view which are important aspects in hard real-time systems. Besides, Event triggered has higher runtime overhead than time trigger. Time triggered has a periodic world. It is not flexible for sporadic tasks. It provides determinism and predictability but with lower rate of resource utilization.

CHAPTER 7

CONCLUSION

In this thesis, scheduling algorithms are classified at three categories. These are fixed priority scheduling, dynamic priority scheduling and time triggered scheduling. These scheduling algorithms were analysed according to parameters which are important in automotive industry. The result of analysis, each scheduling algorithm has advantages and disadvantages. It should be chosen according to requirements.

RM scheduling algorithm can be good choice for a system which includes strict safety requirements like safety-critical systems, because there is no impact to high priority task when missing deadline occurs. A system which needs performance to provide functionality, should use EDF scheduling algorithm, because it provides higher CPU utilization than RM. A deterministic system which provides predictability, should has time triggered scheduling architecture to create periodic world.

One idea is that there can be two hosts on a board for two different scheduling algorithms. One can be chosen for performance and another can be chosen for safety. Tasks can be distributed to related hosts according to their ASILs. Another idea is that developing artificial intelligent which make a decision between schedule tables to handle sporadic tasks at time triggered scheduling algorithms. The last one is a hybrid system which has time trigger architecture however it has an area in each hyper period to processing sporadic task slots. For future studies, this analysis can be done for multicore systems. Each core can be separated for proceeding different ASILs.

REFERENCES

[1] ISO: 26262 – “Road vehicles-Function safety Part 6: Product development at the software level” (2018).

[2] Redmill (1998). "IEC 61508: Principles and Use in the Management of Safety". Computing & Control Engineering Journal, 9, 5, 1998.IEE, London.

[3] ISO: 26262 – “Road vehicles-Function safety Part 2: Management of functional safety” (2018).

[4] Redmill (2000). "Safety Integrity Levels - theory and problems". InRedmill F and Anderson T (eds): Lessons in System Safety -Proceedings of the Eighth Safety-critical.

[5] Jones, Nigel. "Introduction to MISRA C" Embedded Systems Programming, July 2002.

[6] Phillip Laplante, Real-Time Systems Design and Analysis - An Engineer's Handbook, IEEE Press,1993), pp. 11-13.

[7] Buttazzo, G. C.: HARD REAL-TIME COMPUTING SYSTEMS: Predictable Scheduling Algorithms and Applications (3rd ed.), Kluwer Academic Publishers,Boston, 2011.

[8] Abraham Silberschatz, Greg Gagne, Peter B. Galvin, "Operating System Concepts 9/e", Wiley, 2013, pp. 175-210.

[9] Weirong Wang, Aloysius K. Mok, Gerhard Fohler: Pre-Scheduling: Integrating Offline and Online Scheduling Techniques. EMSOFT 2003: 356-372.

[10] D. Rajesh, "Real-time scheduler design for safety-critical systems: A supervisory control approach", 2018.

[11] D. Zmaranda, G. Gabor, D.E. Popescu, C. Vancea, F. Vancea, “Using Fixed Priority Pre-emptive Scheduling in Real-Time Systems” Int. J. of Computers, Communications & Control, Vol. VI (2011), No. 1 (March), pp. 187-195.

[12] J.W.S. Liu, “Real-time systems”, Prentice-Hall, 2000, pp. 60-82.

[13] C.Liu, J.Layland, “Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment”, JACM 20(1), 1973, https://doi.org/10.1145/321738.321743.

[14] Hermann Kopetz. Real-Time Systems - Design Principles for Distributed Embedded Applications. Springer, 2011, pp. 212-216.

[15] H. Kopetz, K. Ki, “Event triggered versus timed triggered real-time systems” Technical report 8/91, Insitut fur Technische Informatik TU Vienna, Austria, 1991.

[16] C.M. Krishna, K.G. Shin, “Real-time systems”, McGraw-Hill Computer Science series, 1997.

[17] Joseph Lemieux, "The OSEK/VDX Standard: Operating System and Communication”, March 2000, p. 90.

[18] http://www.erika-enterprise.com/index.php/community.html.

[19] https://www.arduino.cc/

[20] Giuseppe Lipari, Luigi Palopoli, “Real-Time scheduling: from hard to soft real-time systems”, 2015.

[21] Leung, Joseph; Zhao, Hairong, Real-Time Scheduling Analysis, DOT/FAA/AR-05/27, 2005, 3-1.

[22] D. Isovic and G. Fohler, “Efficient scheduling of sporadic, aperiodic, and periodic tasks with complex constraints,” in Proceedings of the 21st IEEE Real-Time Systems Symposium, Nov. 2000.

[23] Iain John Bate,"Scheduling and Timing Analysis for Safety Critical Real-Time Systems", November 1998.

[24] Stefan Schorr, Gerhard Fohler, “Integrated Time- and Event-Triggered Scheduling – An Overhead Analysis on the ARM Architecture”.

[25] Antonio Valentini, Mohamed Khalgui, Olfa Mosbahi, Embedded Computing Systems: Applications, Optimization, and Advanced Design, 2013: pp. 140-160.