In modern automotive systems, some computational activities are linked to the rotation speed of other subsystems, e.g., wheels, ears, engine. In robotic platforms, proximity sensors are activated more frequently when the robot gets closer to obstacles. Such self-adjusting activities are referred to as Adaptive Various Rate (AVR) task in the literature [1, 2, 3], and adapt their functional requirements based on their activation rate, i.e., period. Table 1 illustrates an example of an AVR task with three levels of functionality in different crankshaft rotation speeds:

<table>
<thead>
<tr>
<th>Rotation (rpm)</th>
<th>Functions to be executed</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,3000]</td>
<td>f1(), f2(), f3(), f4();</td>
</tr>
<tr>
<td>(3000, 6000]</td>
<td>f1(), f2(), f3();</td>
</tr>
<tr>
<td>(6000, 9000]</td>
<td>f1(), f2();</td>
</tr>
</tbody>
</table>

Table 1: An example of an AVR task with three different execution modes.

To aid the system designer how to configure and verify their systems, there are many schedulability analyses in the literature [1,2]. For example, Buttazzo et al. in [1] focus on dynamic priority assignment scheduling, i.e., EDF, whereas Huang and Chen in [2] consider fixed-priority assignment scheduling. However, there are still a lack of studies performing the implementation and evaluation in the real world. Without empirical supports, it is hard to argue which scheduling policy is better for a given system. Recently, Toma and Chen in [2] let FreeRTOS support AVR task model and evaluate the performance on a Raspberry Pi. Similarly, the objective of this thesis is to demonstrate the support of AVR tasks via a new (developing) version of Robot Operating System (ROS2) with the real-time support by enhanced PREEMPT_RT patch, which is one of the real-time solutions for Linux operating systems.

Among various RTOSs, the reason why we choose Linux with PREEMPT_RT patch because many members of the ROS community benefiting from Linux so doing real-time computing use PREEMPT_RT as their real-time solutions. We believe that using a popular / on-going robotic platform to demonstrate the applicability of a new task model is a reasonable combination to bring the theories in the literature from sky to earth. There are many opportunities to have such AVR tasks in the robotic applications, i.e., autonomous automobiles.

In this thesis, the student will study the ideas related to AVR task model in [1, 2, 3] and ROS2 to first get familiar with the expected workload in this thesis. Then the student should review the current implementation of the latest version of PREEMPT_RT patch. After narrowing down the scope of required changes, the student is expected to implement an efficient scheduling routines with reasonable designs. At the end, the implementation should be evaluated by reasonable experiments and demonstrate on an autonomous automobiles. Please note that, the produced source code in this thesis should be publically released and full documented.

Other suggestions and related topics are also welcome. Please do not hesitate to make an appointment.

Required Skills:

- Knowledgeable of C++ and Python programming
- Comfortable with system programming in Linux
- Real Time Scheduling knowledge is beneficial

Acquired Skills after the thesis:

- Design, Analysis, Implementation of scheduling routines in RTOS
- Knowledge of underlying structures in ROS2