Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı 28, S. 1274-1278, Kasım 2021

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue 28, pp. 1274-1278, November 2021

Research Article

Jetson Nano Üzerinde Gerçek Zamanlı Linux Çekirdeği Uygulanması

Salih Palamut[1*], Abdullah Elewi[2] ve Erdinç Avaroğlu[3]

[1*] Mersin University, Faculty of Engineering, Departmant of Computer Engineering, Mersin, Turkey, (ORCID: 0000-0003-3431-6268), palamutsalih@gmail.com
[2] Mersin University, Faculty of Engineering, Departmant of Computer Engineering, Mersin, Turkey, (ORCID: 0000-0001-9774-5292),elewi@mersin.edu.tr
[3] Mersin University, Faculty of Engineering, Departmant of Computer Engineering, Mersin, Turkey, (ORCID: 0000-0003-1976-2526), eavaroglu@mersin.edu.tr

(1st International Conference on Applied Engineering and Natural Sciences ICAENS 2021, November 1-3, 2021)

(**DOI:** 10.31590/ejosat.1014660)

ATIF/REFERENCE: Palamut, S., Elewi, A. & Avaroğlu, A. (2021). Jetson Nano Üzerinde Gerçek Zamanlı Linux Çekirdeği Uygulanması. *European Journal of Science and Technology*, (28), 1274-1278.

Öz

Bu çalışmamızda gerçek zamanlı sistemler ve yaklaşımları hakkında bilgilere yer verilmiştir. Gerçek zamanlı sistemlerin esnek, sıkı ve katı uygulama tiplerine değinilmiş ve gerçek zamanlı bir sistemin modellemesi yapılmıştır. Gerçek zamanlı bir sistemin gömülü bir Linux sürümüne nasıl uygulanacağı araştırılmış, en yaygın kullanılan açık kaynaklı yaklaşımları olan RTLinux, Xenomai ve RTAI hakkında bilgiler sunulmuş ve Jatson Nano üzerinde çalıştırılacak olan tam kesme destekli çekirdek (PREEMPT_RT) hakkında bilgi verilmiştir. Uygun Linux yaması ile Jetson Nano üzerinde bulunan Linux sürümüne tam kesme destekli gerçek zamanlı çekirdek derlemesinin adım adım uygulanması gösterilerek gerçekleştirilmiştir. Yapılan bu çalışmada eski ve yeni çekirdek cyclictest programıyla test edilmiş ve sonuçlar ortaya konulmuştur. Böylece Jatson Nano üzerinde gerçek zamanlı uygulama çalıştırılacak sistem hazırlanmıştır.

Anahtar Kelimeler: Jetson nano, Linux, gerçek zamanlı sistemler, Gerçek zamanlı linux, Linux çekirdeği, Zamanlama.

Real-Time Linux Kernel Implementation on Jetson Nano

Abstract

In this study, information about real-time systems and their approaches is given. Flexible, tight and rigid application types of real-time systems are mentioned and a real-time system is modeled. How to apply a real-time system to an embedded Linux version is researched, information about the most widely used open-source approaches, RTLinux, Xenomai and RTAI, and information about the fully interrupt supported kernel (PREEMPT_RT) to be run on Jatson Nano is given. Demonstrated step-by-step implementation of real-time kernel compilation with full interrupt support to the version of Linux on Jetson Nano with the appropriate Linux patch. In this study, the old and new cores were tested with the cyclictest program and the results were presented. Thus, the system to run real-time applications on Jatson Nano has been prepared.

Keywords: Jetson nano, Linux, Realtime systems, Realtime linux, Linux kernel, Timing.

[*] Corresponding Author: palamutsalih@gmail.com

1. Giriş

Gerçek zamanlı sistemlerde, gerçek zamanlı bir programın doğru yürütülmesi yalnızca çıktının mantıksal doğruluğuna değil, aynı zamanda bu tür sonuçların belirli bir zaman çerçevesi veya son tarih içinde gerçekleştirilip gerçekleştirilmediğine de bağlıdır (Reghenzani, Massari ve Fornaciari, 2019).

Bir sistemin gerçek zamanlı olabilmesi için görev zamanlama algoritmalarını ve gerçek zamanlı bir programın doğru yürütülmesini desteklemelidir. Literatürde birden fazla gerçek zamanlı algoritma mevcuttur, sık kullanılan algoritmalardan (RM, EDF, PIP, PCP, SRP) en az birini tam anlamıyla desteklenmelidir (Palamut, Gonultas, Elewi ve Avaroglu, 2019).

Gerçek zamanlı sistemlerin uygulama tipleri üç madde halinde incelenebilir. Bu maddeler zaman limitlerine göre oluşturulmuştur (Kopetz, 2013).

Esnek uygulamaların amacı hizmet kalitesini (QoS) korumaktır. Buna en iyi örnek video karelerinin bir saniyede geçmesi gereken kare sayısını korumaktır. Bazı karelerin teslim sürelerinin kaçırılması insan veya nesneler üzerinde güvenlik açısından kritik sonuçlara yol açmadığından sadece kalitenin düşmesi anlamına gelir, fakat bu durumun kabul edilebilir bir kullanıcı deneyimi sağlaması esastır.

Sıkı uygulama, esnek uygulamaya benzer, ancak herhangi bir durumda gerçekleşen işlem kaçırma çıktıyı geçersiz kılar. Bu tarihten sonra gerçekleşen sonuçların hiçbir değeri yoktur ve kullanılmamalıdır.

Katı uygulama, genellikle gerçek zamanlı bir sistemden bahsedildiğinde bu tip sistemler akla gelir. Bu sınıfta son teslim tarihinin kaçırılması istenmeyen sonuçlara yol açacağından zaman kısıtlamalarını garanti etmek için tüm sistemin ve sistem kodlarının detaylı analizlerini gerektirir. Bu tarz uygulamalar güvenlik açısından kritik uygulamalarda kullanılırlar. Havacılıkta kullanılan otomatik pilot uygulamaları bu sınıfa girer.

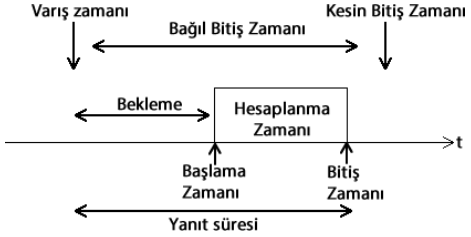
Gerçek zamanlı görevler (RT Task), gerçek zamanlı bir çekirdek (RT Kernel) tarafından yürütülür. Gerçek zamanlı sistemleri daha iyi anlayabilmek için yapısına bakalım. Bu yapı için önemli olan bazı terimler şunlardır:

İş Cevap Süresi (Job Response Time), işin sonlandığı zamandır. Serbest bırakma anında ölçülür.

Görev Cevap Süresi (Task Response Time), tüm işler arasındaki maksimum yanıt süresidir.

Kritik An (Critical Instant), bir görev için; en büyük görev yanıt süresini üreten varış zamanıdır.

Sistem modeli Şekil 1'de olduğu gibi, serbest bırakılma | varış zamanı (release | arrival time) ile işin kesin bitirilme zamanı (deadline) arasındaki süre, hesaplanma zamanı (execution time) olarak tanımlanmaktadır (Liu, 2000).



Şekil 1. Sistem Modeli

Girişte sistem tanımlanması yapılmış olup, ikinci bölümde gerçek zamanlı Linux çekirdeğinden bahsedilmiş, üçüncü bölümde Jetson Nano üzerinde Linux çekirdeğinin nasıl kullanılacağı gösterilmiştir. Üçüncü bölümde yapılan işlemlerden elde edilen bilgiler bulgular kısmında toplanmış, bu bilgilerden yola çıkılarak tartışma bölümünde faydası açıklanmıştır. Son olarak sonuç bölümünde çalışma özetlenmiştir.

2. Materyal ve Yöntem

Gerçek zamanlı programları çalıştırabilme özelliklerine sahip çekirdek ve çekirdek yaklaşımları gerçek zamanlı Linux çekirdeği, yardımcı çekirdek yaklaşımları ve öncelikli çekirdek yaklaşımı olmak üzere üç başlık altında toplanıp bu bölümde incelenmiştir.

2.1. Gerçek Zamanlı Linux Çekirdeği

RTLinux, robotları, veri toplama sistemlerini, üretim tesislerini ve diğer zamana duyarlı aletleri ve makineleri kontrol etmeyi mümkün kılan katı gerçek zamanlı Linux sürümüdür (Yodaiken, 2000).

Yaklaşık 43 yıl önce araştırmacılar Bell Laboratuvarlarında Multi-Environment Real-Time (MERT) adında deneysel bir işletim sistemi gerçekleştirdi (Jacobs, 1978). Bu işletim sisteminin hem gerçek zamanlı hem de genel amaçlı uygulamaları çalıştırması amaçlanmıştır. Ancak MERT tasarımcıları, her ikisini de destekleyebilecek tek bir işletim sistemi yapmaya çalışmak yerine, gerçek zamanlı bir işletim sistemi ile genel amaçlı (zaman paylaşımlı) bir işletim sisteminin birlikte çalıştığı bir sistem yapmaya çalıştılar.

Tasarımcılar gerçek zamanlı işletim sistemi ile aynı makinede sofistike bir zaman paylaşım sisteminin mevcudiyeti, insan-makine ara yüzünün gerçek zamanlı süreçlere tasarlanmasında yararlanılabilecek güçlü araçların sağlandığına dair kanıtları öne sürmüştür.

Yani MERT'in tasarımcıları, gerçek zamanlı işletim sistemini gerçek zamanlı olmayan işletim sisteminden ayırarak, uygulama geliştiricilerin gerçek zamanlı olmayan işletim sisteminin hizmetlerini kullanmasına izin verebildiklerini iddia ettiler (Yodaiken, 2000).

RT çekirdeğinin görevi, gerçek zamanlı görevler için ham donanıma doğrudan erişim sağlamaktır, böylece ihtiyaç duyduklarında minimum gecikme ve maksimum işlemeye sahip olabildiğinden, MERT'de bulunan temel işleme fikrinden daha iyidir (Yodaiken, 2000).

Bu çalışmada PREEMPT_RT yaması kullanılacağından diğer çekirdek yaklaşım tarzlarından bahsedilmiştir.

2.2. Yardımcı Çekirdek Yaklaşımları

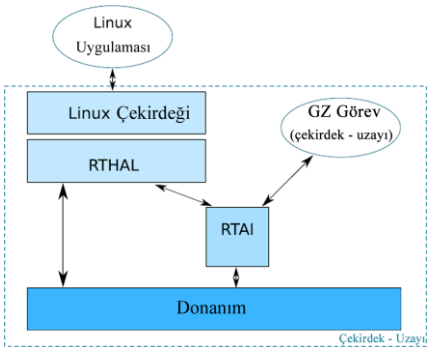
PREEMPT_RT yamasına yönelik en yaygın alternatif yaklaşımlar, genellikle gerçek zamanlı iş parçacıklarının yönetiminden sorumlu ek bir işletim sistemi "pico-kernel" in varlığına dayanır. Eşdeğer olarak kokernel, pico-kernel, nano-kernel veya dual kernel (dk) tabanlı yaklaşımlar olarak adlandırılırlar. Buradaki temel fikir, piko-çekirdeğin donanım ve genel amaçlı Linux çekirdeği arasında bir katman olarak çalışmasını sağlamaktır (Yodaiken, 2000).

En yaygın kullanılan açık kaynaklı yaklaşımlar RTLinux, Xenomai ve RTAI dir. Bu yaklaşımlardan ikisi, Xenomai veya

RTAI oluşan donanım kesintilerini (interrupt) yönetmek için Adaptive Domain Environment for Operating System (ADEOS) katmanını kullanır. Aslında ADEOS, her biri özel bir adres alanına ve kaynaklara sahip ayrı işletim sistemi alanları yaratır. Etki alanları arasında kaynak paylaşımı (ör. Cihazlar) yalnızca ADEOS aracılığıyla mümkündür (Yaghmour, 2002).

2.2.1. RTAI

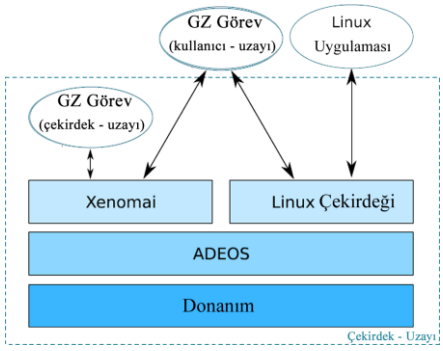
Paolo Mantegazza tarafından Politecnico di Milano'da 2000 yılında geliştirilen RTAI (Mantegazza, Dozio ve Papacharalambous, 2000), çekirdek alanı düzeyinde Linux'a gerçek zamanlı yeteneklerini sağlamayı amaçlayan bir ortak çekirdek yaklaşımıdır. Kullanıcı alanında LXRT arabirimi aracılığıyla sınırlı sayıda bu tür yetenekler de sağlanır. Daha sonra RTAI, ADEOS alt sistemine taşındı. Bu durumda, Xenomai'den farklı olarak, kesintilerin işlenmesi esas olarak RTAI tarafından yönetilir. Bu yaklaşım Şekil 2'de gösterilmiştir.



Şekil 2. RTAI

2.2.2. XENOMAI

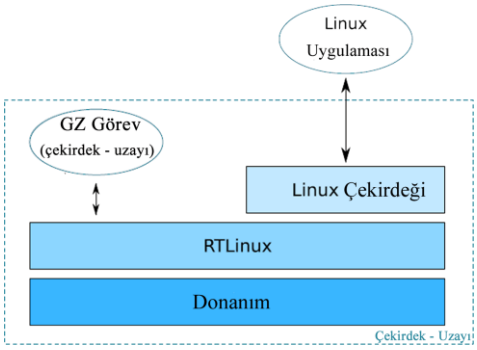
Philippe Gerum tarafından 2002 yılında piyasaya sürülen Xenomai (Gerum, 2004) kullanıcı alanında gerçek zaman sağlayan bir katmandır. Xenomai başlangıçta ADEOS katmanına dayanıyordu, ancak son sürümlerde yalnızca basitleştirilmiş bir kısmı, özellikle de I-pipe (interrupt pipeline) adı verilen kesme dağıtım alt sistemi kullanmaktadır. Bu yaklaşım Şekil 3'te gösterilmiştir.



Şekil 3. Xenomai

2.2.3. RTLINUX

Victor Yodaiken tarafından 1997 yılında geliştirildi, ilk kararlı sürüm 2000 civarında gerçekleşti ve ardından patent 2007'de Wind River Systems'a satıldı. RTLinux (Yodaiken, 2000), Linux çekirdeğini gerçek zamanlı uygulamalarla birlikte tamamen öncelikli bir işlem olarak çalıştırır. Temel olarak, RTLinux, genel amaçlı çekirdeğin ne zaman çalıştırılacağına ve gerçek zamanlı görevlerin son teslim tarihlerine göre ne zaman çalıştırılacağına karar veren bir zamanlayıcı uygular. Bu yaklaşım Şekil 4'te gösterilmiştir.



Şekil 4. RTLinux

2.3. Öncelikli Çekirdek Yaklaşımı

Gerçek zamanlı bir Linux çekirdeğinin ilk izleri, 1990'ların sonlarında literatürde bulunabilir. En başından beri çözülmesi en zor sorun, Linux çekirdeğinin doğrudan kesmeleri destekleyecek şekilde öncelikli kullanmanın imkansızlığından ibaretti.

Finney tarafından 2001 yılında yayınlanan bir makaleye göre (Finney, 2001), eğer işlerin sonlanma zamanı milisaniyeler düzeyinde gerçekleşecek şekilde kalırsa, tek çekirdekli bir sistemde gerçek zamanlı görevler için saf bir Linux çekirdeği kullanılabileceği fikrini öne sürdü.

Linux çekirdeğinin PREEMPT_RT yaması aslında bir grup çekirdek geliştiricisi tarafından geliştirilen bir dizi yamadır. Proje Ingo Molnár tarafından başlatıldı ve ilk sürüm 2.6.11 çekirdek sürümüne dayanıyordu (Mart 2005).

Böylelikle bir yama ile hali hazırda kullanılan Linux çekirdeği öncelikli çekirdek modeline çevrilebiliyordu.

3. Jetson Nano Üzerinde Gerçek Zamanli Linux Çekirdeği Uygulanmasi

Jetson nano için gerçek zamanlı çekirdek (kernel) derlenmiştir. Linux terminal kullanımı ve temel Linux bilgisi gerekmiş ve daha detaylı teknik bilgi için Jetson Nano kurulum belgesi kullanılmıştır (Palamut, 2021).

İşlem yapılacak sistemin çekirdek sürüm numarası öğrenilmiş ve bu sürüm numarasına uygun yama paketleri sistem üzerine indirilmiştir (N., 2019).

$ wget https://developer.nvidia.com/embedded/l4t/r32_release_v5.1/r32_release_v5.1/sources/t210/public_sources.tbz2

$ tar xvf public_sources.tbz2 Linux_for_Tegra/source/public/kernel_src.tbz2

$ cp Linux_for_Tegra/source/public/kernel_src.tbz2 kernel_src.tbz2

$ rm -rf Linux_for_Tegra

$ tar -xvf kernel_src.tbz2

$ cd kernel/kernel-4.9

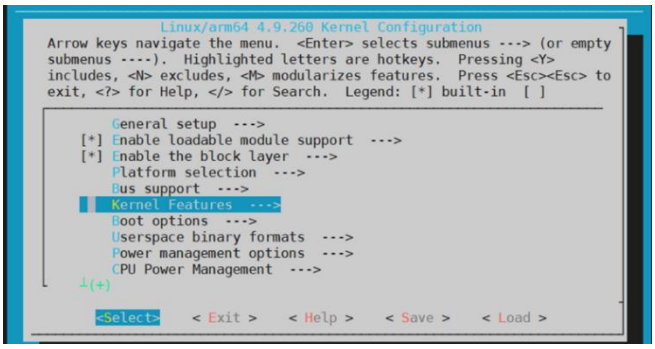
Bu işlemlerden sonra derleme aşamasına geçilmiştir. İlk adım olarak çekirdek ayarları Jetson üzerinden alındıktan sonra gerçek zaman yapasına uygun ayarlar eklenmiş ve sistem çekirdeği ayarını tam destekli kesme tanımı olan Fully Preemptible Kernel (RT) olarak gerçekleştirilmiştir (How to build NVIDIA Jetson Nano kernel, 2021).

$ zcat /proc/config.gz > .config

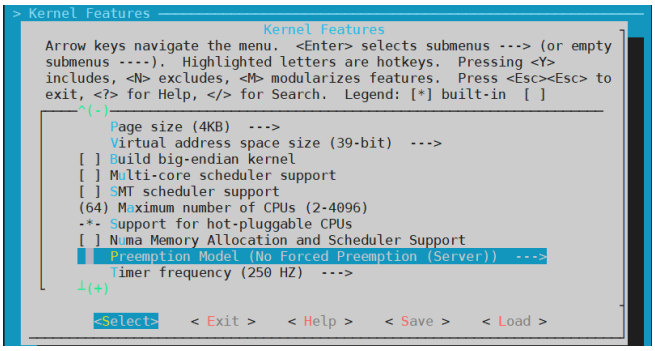
$ ./scripts/rt-patch.sh apply-patches

$ make menuconfig

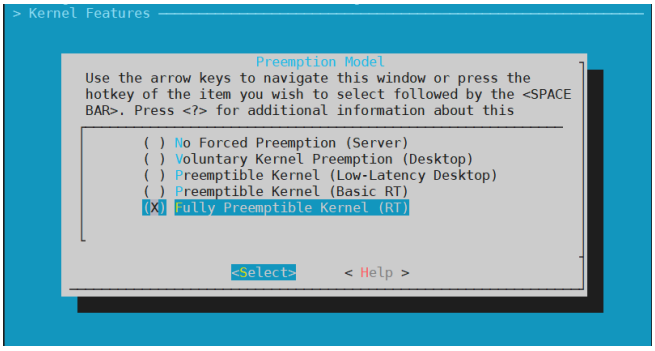
Bu aşamada gelen menüden Kernel Features ->Preemption Model-> Fully Preemptible Kernel (RT) seçimi yapılarak çekirdek gerçek zamanlı olarak ayarlanmıştır. Bu aşamalar Şekil 5,6 ve 7'de gösterilmiştir.



Şekil 5. Çekirdek Ayarları Bölümü



Şekil 6. Çekirdek Özellikleri Bölümü



Şekil 7. Gerçek Zaman Destek Çeşidi

Diğer seçim işlemleri menü üzerinden bulunmuş ve gerçekleştirilmiştir.

#Kernel Features -> Timer frequency: 1000 HZ

#General setup -> Timers subsystem : Full dynticks system (tickless)

İlgili işlem adımları tamamlanmış ve sayfadan çıkılırken dosya kaydedilmiştir. Derleme işlemi yaklaşık 60 dakika sürmüştür.

$ cd ../..

$ ./nvbuild.sh

$ sudo make modules_install

$ sudo cp arch/arm64/boot/Image /boot/Image

Bu aşamada işlemler başarılı olmuş ve sistemi yeniden başlattığımızda gerçek zamanlı çekirdeğimiz çalışmaya başlamıştır.

Bu işlemi doğrulamak için Linux komut penceresinden uname -a yazarak çekirdek isminde #1 SMP PREEMPT RT ifadesi olup olmadığı kontrol edilmiş ve #1 SMP PREEMPT RT ifadesi yer aldığı görülmüştür.

4. Bulgular

Yapılan çalışmadan elde edilen sonuçlar karşılaştırması gereken test yazılımı ve karşılaştırma çıktıları bu bölümde yer almıştır.

4.1. Test Yazılının Çalıştırılması

Bu işlemler için cyclictest (Cyclictest start [Wiki], 2018) isimli uygulama kullanılmıştır. Bu uygulamanın kullanılabilmesi için kaynak koddan derlenmiş ve kurulmuştur.

$ sudo apt install libnuma-dev -y

$ wget https://mirrors.edge.kernel.org/pub/linux/utils/rt-tests/rt-tests-1.10.tar.gz

$ tar xvzf rt-tests-1.10.tar.gz

$ cd rt-tests-1.10/

$ make && sudo make install

4.2. Yamadan Önce ve Sonrasının Karşılaştırılması

Karşılaştırma için uygulamanın yönetici izinleriyle çalıştırılmış. Bu bilgileri grafik çıktı almak için OSADL'nin sunduğu grafik işlemleri kullanılmıştır (OSADL., 2017).

$ sudo apt install gnuplot -y

$ mkdir logs && cd logs

$ wget https://www.osadl.org/uploads/media/mklatencyplot.bash

$ chmod u+x mklatencyplot.bash

Bu test yaklaşık 6 saat sürmüştür. Daha hızlı sonuç almak için mklatencyplot.bash dosyasının ilk satırında bulunan döngü sayısını 100000 ile değiştirilmiş ve cyclictest ifadesi başına sudo ifadesini eklenmiştir.

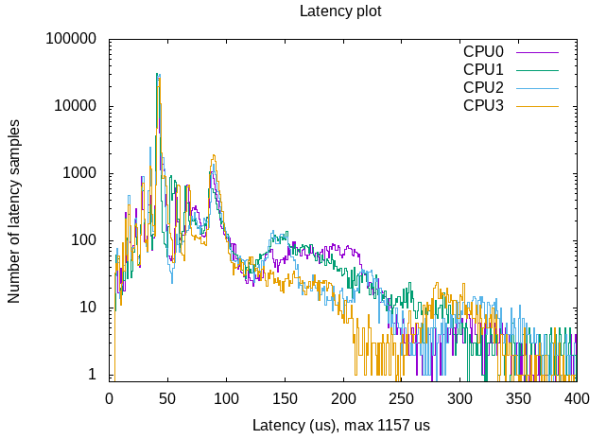
$ sudo cyclictest -l100000 -m -Sp90 -i200 -h400 -q >output

İlk satır bilgilerini açıklamaları:

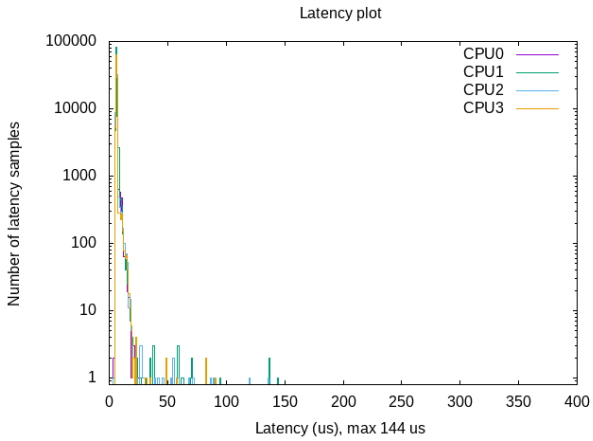
- -l: döngü sayısı
- -m: Bellek erişimini kilitle
- -Sp90: Tüm mevcut çekirdekleri kullan ve önceliği 90 olarak belirle.

- -i200: işparçaları arasında geçen süre
- -h400: 400 mikro saniye aralıklı histogram çıktısı üret
- -q: işlem bitince ekranda çıktıyı göster.
- >: Linux konsol bilgilerini dosyaya aktarma işlemi.

İlgili komut dosyası olan mklatencyplot.bash içinden çeşitli denemeler yapılmış ve yapılan bazı denemelerde alınan çıktılar aşağıda verilmiştir. İmajın üzerinde hazır gelen çekirdek değerleri Şekil 8 üzerinde, yeni hazırladığımız çekirdeğin çalışma değerleri ise Şekil 9'da gösterilmiştir.



Şekil 8. SMP Kernel



Şekil 9. RT Kernel

5. Tartışma

RT Kernel desteği görüntü işleme yapılan sistem için önemlidir. Bu çalışmamızda gömülü sistem olarak Nvdia Cuda Destekli OpenCv çalıştırabilen Jatson Nano tek kart bilgisayarına gerçek zamanlı uygulama çalıştırabilen çekirdek derlenmiştir. Böylelikle yapayzeka destekli görüntü işleme algoritmalarında gerçek zamanlı programlama yapılabilir.

6. Sonuç

Yapılan çalışmada günümüze kadar olan gerçek zamanlı Linux kullanımları ele alınmış, farkları incelenmiş ve Jetson Nano üzerinde tam kesme önceliği destekli Linux çekirdek yaması kurulum ve kullanım işlemleri gösterilmiştir. Yapılan testlere göre SMP çekirdekte alınan 1157 mikro saniyelik gecikme değeri Rt çekirdeğinde 144 mikro saniye olarak gözlemlenmiştir. Böylelikle sistem gerçek zamanlı işlemler için hazır hale getirilmiştir.

Kaynakça

Reghenzani, F., Massari, G., & Fornaciari, W. (2019). The Real-Time Linux Kernel. ACM Computing Surveys, 52(1), 1–36. https://doi.org/10.1145/3297714

Palamut, S., Gonultas, T., Elewi, A., & Avaroglu, E. (2019). Task Scheduling Algorithms and Resource Access Protocols in Real Time Systems. 2019 International Artificial Intelligence and Data Processing Symposium (IDAP). Published. https://doi.org/10.1109/idap.2019.8875974

Kopetz, H. (2013). [Real-Time Systems: Design Principles for Distributed Embedded Applications (Real-Time Systems Series)] [By: Kopetz, Hermann] [May, 2013]. Springer.

Liu, J. W. S. (2000). Real-Time systems. Upper Saddle River, NJ: Prentice Hall.

Yodaiken, V. (2000). The RTLinux Manifesto. Yodaiken. http://www.yodaiken.com/papers/rtlmanifesto.pdf

Jacobs, I. (1978). Atlanta Fiber System Experiment: Overview. Bell System Technical Journal, 57(6), 1717–1721. https://doi.org/10.1002/j.1538-7305.1978.tb02121.x

Yaghmour, K. (2002). Adaptive Domain Environment for Operating Systems. Adaptive Domain Environment for Operating Systems. Published

Mantegazza, P., Dozio, L., & Papacharalambous, S. (2000). RTAI: Real time application interface. Linux Journal. Published.

Gerum, P. (2004). Xenomai - Implementing a RTOS emulation framework on GNU / Linux (First Edition). Philippe Gerum

Finney, S. A. (2001). Real-time data collection in Linux: A case study. Behavior Research Methods, Instruments, & Computers, 33(2), 167–173. https://doi.org/10.3758/bf03195362

Palamut, S. (2021). JETSON NANO KURULUM. Google Books. https://books.google.com.tr/books?id=6UojEAAAQBAJ&lpg=PR1&hl=tr&pg=PR1#v=onepage&q&f=false.

N. (2019, December 17). PREEMPT-RT patches for Jetson Nano. NVIDIA Developer Forums. https://forums.developer.nvidia.com/t/preempt-rt-patches-for-jetson-nano/72941/25

How to build NVIDIA Jetson Nano kernel. (2021). RidgeRun. https://developer.ridgerun.com/wiki/index.php?title=Jetson_Nano/Development/Building_the_Kernel_from_Source#3._Compile_kernel_and_dtb

Cyclictest start [Wiki]. (2018). Linuxfoundation. https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/cyclictest/start

OSADL. (2017). Create a latency plot from cyclictest histogram data: OSADL - Open Source Automation Development Lab. https://www.osadl.org/Create-a-latency-plot-from-cyclictest-hi.bash-script-for-latency-plot.0.html