Journal of Computer Science
https://dergipark.org.tr/en/pub/bbd

ISSN,e-ISSN: 2548-1304
Volume:IDAP-2023, pp:83-89, 2023
https://doi.org/10.53070/bbd.1350343
Research Paper

Nesnelerin İnternetinde Dallanmasız Programlama Tekniklerinin Uygulanması

Application of Branchless Programming in Internet of Things

Muhammed Saadetdin KAYA [*1], Kenan İNCE [2]

[1]Bilgisayar Mühendisliği Bölümü, KTO Karatay Üniversitesi, Konya, Türkiye

[2]Bilgisayar Mühendisliği Bölümü, İnönü Üniversitesi, Malatya, Türkiye

(muhammed.saadetdin.kaya@karatay.edu.tr, kenanince@gmail.com)

Received:Sep.02,2023 Accepted:Oct.16,2023 Published:Oct.18,2023

Özetçe— Akıllı evlerden endüstriyel otomasyona, sağlık hizmetlerinden askeri silah sistemlerine kadar çeşitli alanlarda etkinliğini gittikçe arttıran Nesnelerin İnterneti, günümüz dünyasındaki dijital dönüşümün kritik bir bileşeni olarak her geçen gün hayatımıza daha fazla dahil olmaktadır. Kullanımının alanlarının yaygınlaşması ile Nesnelerin İnterneti daha çetrefilli bir konu haline gelmektedir. Nesnelerin İnterneti cihazları uç işlem birimleri olmaları sebebiyle, gerek tekil gerek bir bütün olarak uygulama alanlarının genişlemesine rağmen tükettikleri enerjinin azaltılması ve daha esnek, daha kompakt yapılara dönüştürülmesi, geniş bir uygulama alanı haline gelmiştir. Bu nedenle hafif siklet şifreleme algoritmaları, veri sıkıştırma teknikleri ve özel donanımsal tasarımlarının yanında bu sistemlerde kullanılan mikroişlemcilerin performans odaklı programlanması da önem arz etmektedir. Bu çalışmada, koşullu yapıların aritmetik operatörlerin ve mantık operatörlerinin kombinasyonel olarak değiştirilmesiyle performans artışını hedefleyen yeni bir konsept olan Dallanmasız Programlama teknikleri Nesnelerin İnterneti alanında uygulanmıştır. Gerçekleştirilen uygulama sonucunda sistem geleneksel yöntemlerle zaman ve uzay karmaşıklığı açısından kıyaslanmıştır. Elde edilen sonuçlar doğrultusunda Dallanmasız Programlama tekniklerinin en uygun senaryolarda doğru şekilde kullanılmasıyla ciddi manada performans artışı ve gözle görülebilir şekilde depolama alanı tasarrufu sağladığı saptanmıştır.

Anahtar Kelimeler : *Dallanmasız Programlama, Nesnelerin İnterneti, Optimizasyon, Karmaşıklık*

Abstract— Increasing its effectiveness in various fields from smart homes to industrial automation, from health services to military weapon systems, the Internet of Things is becoming more and more involved in our lives as a critical component of the digital transformation in today's world. With the widespread use of the Internet of Things, the Internet of Things is becoming a more complicated subject. Due to the fact that IoT devices are end-processing units, reducing the energy they consume and transforming them into more flexible and more compact structures has become a wide application area, despite the expansion of their application areas, both individually and as a whole. For this reason, performance-oriented programming of the microprocessors used in these systems is important, as well as light weight encryption algorithms, data compression techniques and special hardware designs. In this study, Branchless Programming techniques, a new concept aiming to increase performance by combinatorial replacement of conditional structures, arithmetic operators and logic operators, have been applied in the field of Internet of Things. As a result of the implementation, the system was compared with traditional methods in terms of time and space complexity. In line with the results obtained, it has been determined that the correct use of Branchless Programming techniques in the most appropriate scenarios provides a significant performance increase and a visible storage space savings.

Keywords : *Branchless Programming, Internet of Things, Optimization, Complexity*

1. Giriş

Nesnelerin İnterneti (Nİ), son yıllarda araştırma ve uygulama alanları sürekli genişleyen interneti kullanan diğer cihazları algılama ve bunlarla iletişim kurma yeteneğine sahip varlıkların oluşturmuş olduğu bir ekosistemdir (Gubbi vd., 2013). İnternet erişiminin global çapta artması ve erişim maliyetlerinin artması gibi pozitif gelişmelerle beraber bu alanda ortaya koyulmuş olan başarılı örnekler Nİ nesnelerinin farklı alanlarda daha yaygın bir şekilde kullanılması amacıyla bu alandaki çalışmaların ve internete bağlı cihazların sayısını gün geçtikçe arttırmaktadır (Want ve Dustdar, 2015; Qadri vd., 2020). Farklı kullanım alanları ve bu alanlardaki uygulama şartları Nİ nesnelerinin genellikle düşük bilgisayar gücüne sahip az kaynakla verimli işler yapmaya çalışan cihazlar olarak karşımıza çıkmasına sebep olmaktadır.

Nİ sistemlerinde kullanılacak olan nesnelerin ve sistemin tasarımı esnasında ortaya çıkan depolama alanı kısıtlamaları, veri iletimi ve sistem güvenliği, işlem hızı, kaynak tüketimi ve kompakt tasarım ihtiyacı gibi sorunların çözülmesi için uygulama alanına bağlı olarak çeşitli çözümler sunulmaktadır (Mangla vd., 2022). Nİ nesneleri arasında güvenli iletişimin sağlanması için geleneksel şifreleme ve güvenlik protokollerinin yerine hafif siklet uygulamaların seçilmesi (HaddahPajouh vd., 2021), depolama alanı ve bant genişliği sorunlarını ortadan kaldırmak adına veri sıkıştırma yöntemlerinin kullanılması (Chowdhury vd., 2020), veri bütünlüğünün korunması gayesiyle blokzincir uygulamalarının hafifletilerek Nİ sistemlerine entegre edilmesi (Zafar vd., 2022) gibi birçok çözümün senaryo şartları ve ihtiyaçları göz önünde bulundurularak Nİ'de bu alana özel bir şekilde uygulandığı görülmektedir.

Nİ için sunulan çözüm yöntemlerinin her birinin aslında birer bilgisayar mühendisliği problemi olduğu ve daha büyük ölçekte çözülmeye çalışıldığı düşünüldüğünde, tıpkı geleneksel bilgisayar uygulamalarında olduğu gibi, çözüm yollarının doğru bir şekilde seçilmesinin yanında sisteme doğru bir şekilde uygulanması da önem arz ettiği görülmektedir (Khan vd., 2021). Özellikle kaynak kapasitesi açısından çok düşük ölçekte olan Nİ nesnelerinde de geleneksel bilgisayar uygulamalarında olduğu gibi bir kazanç-kayıp takası durumu söz konusu olacaktır. Bununla birlikte seçilen çözüm yöntemlerinin doğru bir şekilde kodlanmasıyla çözüm yöntemi daha da iyileştirilebilecektir (Liv vd., 2019).

Bu çalışmada, kodlama yöntemlerinden biri olan dallanmasız programlama yönteminde kullanılan çeşitli teknikler incelenmiş olup; Nİ alanında 32-bitlik bir işlemcide bu tekniğin pratik uygulamalarından bazıları deneyimlenmiş ve bu tekniğin doğru bir şekilde kullanılmasıyla sistem iyileştirmesi sağlanıp sağlanamayacağı gözlemlenmiştir. Elde edilen bulgular zaman ve uzay karmaşıklıkları açısından yorumlanmış, dallanmasız programlamanın sınırlamaları incelenmiş ve bu tekniğin hangi durumlarda uygulanıp hangi durumlarda uygulanamayacağı hakkında öngörülerde bulunulmuştur.

2. Dallanmasız Programlama

Günümüzde Nİ'de kullanılan işlemciler genellikle bir kod bloğunda bulunan kodları sırasıyla ve tek tek işlemektedir. Bu şekilde çalışma prensibine sahip olan güncel Nİ işlemcileri, kodu işlerken o anda işlenen kod satırından sonraki kod satırını işlem öncesinde hazırlayarak satırlar arası beklemeyi asgari düzeye indirmeyi hedeflemektedir (Wang vd., 2016). Ancak bu durum koşullu bölümlere sahip kod bloklarında bir problemi ortaya çıkarmaktadır. Kodun işlendiği andaki şartlara göre yapılacak işlemlerde farklılıklar olması anlamına gelen koşullu bölümler, kod bloğunda mevcutsa işlemci koşulun sağlanıp sağlanmadığını kodu işlemeden kesin olarak bilemeyeceği için işlem tamamen gerçekleşene kadar gelecekteki durumu da tespit edemeyecek ve hangi işlemleri hazırlaması gerektiğini anlayamayacaktır. Bu da bir kod bloğunda bulunan her bir koşullu bölümün işlemcinin gitmesi muhtemel bir güzergah oluşturacağı anlamına gelmektedir.

Bir işlemcinin mevcut kod bloğunda gidebileceği güzergahların her birine dal ismi verilmektedir. Programlamada dallanma ise bir kod bloğunda işlemcinin izleyebileceği birden çok muhtemel yol bulunuyor olması demektir. İşlemci dallanmaların sayısının artmasıyla sıkça belirsizlik problemiyle karşılaşacaktır. Dolayısıyla sonraki satırın hazırlanma işlemleri mevcut kod satırı tamamen işlenene kadar bitmeyecektir. Bu da işlemler arası bekleme süresinin artması anlamına gelecek olup iş yükünün tamamlanmasını geciktirecektir.

Dallanmasız programlama, dallanma kaynaklı belirsizlik problemini çözmek adına ortaya çıkmış bir programlama konseptidir. Dallanmasız programlama genel olarak koşula bağlı kod bloklarının, atamaların ve metodların, aritmetik ve mantıksal operatörler yardımıyla değiştirilmesini önererek dallanma kaynaklı belirsizliklik sayısının yani işlemler arası bekleme süresinin asgari seviyeye indirilmesi hedeflemektedir.

2.1. Mantıksal Operatörler ile Dallanmasız Programlama:

Mantıksal (bitsel) operatörler bir bilgisayarın belleğindeki verileri temsil eden bitleri (durum bayraklarını) doğrudan manipüle ederek koşulların geçerliliğini işlemcinin tespit etmesini sağlarlar. Mantıksal operatörlerin bu özelliğinden dolayı; dallanmasız programlamada sıklıkla kullanılan yöntemlerden biri koşula bağlı tanımlamalar yerine mantıksal operatörlerin matematiksel fonksiyonlarla birlikte kullanılarak tek bir tanımlama yapılmasıdır.

Kod 1'de girdi olarak gönderilen iki sayıdan hangisinin büyük olduğunu bulma işleminde sayılar arasında kıyaslama yaparak iki yoldan birini seçildiği bir kod bloğu bulunmaktadır. Burada işlemci koşulun sağlanıp sağlanmadığını koşul kıyaslaması yapılmadan bilemeyeceği için EĞER'den sonraki satırı mı veya YOKSA'dan sonraki satırı mı işlemesi gerektiğini önceden bilemeyecek, dolayısıyla sıradaki işlemi hazırlayamayacaktır.

Kod 1. İki sayıdan büyük olanı bulan metodun sözde kodu

```
BüyükSayıyıBul Fonksiyonunu PARAMETRELER a ve b ile başlat
EĞER a, b'den büyükse,
  SONUÇ olarak a döndür.
YOKSA,
  SONUÇ olarak b döndür.
Fonksiyon sonu.
```

Kod 2'de ise Kod 1'deki gibi girdi olarak gönderilen iki sayıdan hangisinin büyük olduğunu bulma işleminde işlemci hangi sayının büyük olduğunun önemi olmaksızın kesin olarak SONUÇ bölümündeki işlemi gerçekleştirecektir. Burada matematiksel olarak işlem yükü daha fazla olmasına karşın kesinlik söz konusu olduğundan dolayı işlemcinin önünde tek bir yol olacaktır ve bu yol verilen matematiksel işlemin gerçekleştirilmesiyle tamamlanacaktır. Dolayısıyla işlemci hangi yoldan gideceğini bildiği için sonrasında yapacağı işlemleri önceden hazırlayarak zaman kaybına uğramayacaktır.

Kod 2. İki sayıdan büyük olanı bulan metodun dallanmasız kodlanmış sözde kodu

```
BüyükSayıyıBul Fonksiyonunu PARAMETRELER a ve b ile başlat
SONUÇ olarak (a > b) * a + (a <= b) * b işleminin sonucunu döndür.
Fonksiyon sonu.
```

2.2. Aritmetik Operatörler ile Dallanmasız Programlama:

Toplama, çıkarma, çarpma ve bölme gibi aritmetik operatörlerin işlem verimliliği ile dallanmadan kurtulmak ve mutlak yola sahip kod blokları oluşturmak mümkündür. Dallanmadan kurtulmanın yanı sıra performans açısından oldukça başarılı olan bu temel dört işlemin haricinde, onlar kadar performans verimli olmasa da kullanılabilecek mod işlemi; özellikle dallanan sıralı liste işlemleri için dallanmanın ortadan kaldırılmasında önemli bir rol oynamaktadır.

Kod 3'te girdi olarak gönderilen konum değeri kullanılarak o konumda bulunan liste elemanını kullanıcıya döndüren program görülmektedir. Burada işlemci koşulun sağlanıp sağlanmadığını koşul kıyaslaması yapılmadan bilemeyeceği için EĞER'den sonraki satırı mı veya YOKSA'dan sonraki satırı mı işlemesi gerektiğini önceden bilemeyecek dolayısıyla sıradaki işlemi hazırlayamayacak dolayısıyla performans sorunu ile karşı karşıya kalacaktır.

Kod 3. Girdideki pozisyonda bulunan dizi elemanının değerini döndüren metodun sözde kodu

```
PozisyondakiElemanıDöndür Fonksiyonunu PARAMETRE i ile başlat
EĞER i, lenX'den büyük veya eşitse,
  x dizisindeki (i - lenX) indeksindeki elemanı döndür.
YOKSA,
  x listesindeki i. indeksindeki elemanı döndür.
Fonksiyon sonu.
```

Kod 4'te Kod 3'teki ile aynı amaca yönelik yazılmış olan bir metot bulunmaktadır. Ancak bu metot Kod 3'tekinin aksine mutlak bir yola sahiptir. Yapılan işlem girdi olarak gönderilen pozisyon değerini herhangi bir değer ile kıyaslamadan, koşula bağlı olmaksızın doğrudan dizi uzunluğuna göre mod alarak dizi elemanına erişim sağlamaktadır. Bu da girdi değeri ne olursa olsun yapılacak işlemin aynı olacağı anlamına gelmektedir. Dolayısıyla işlemci hangi yoldan gideceğini bildiği için sonrasında yapacağı işlemleri önceden hazırlayarak zaman kaybına uğramayacaktır.

Kod 4. Girdideki pozisyonda bulunan dizi elemanının değerini döndüren metodun dallanmasız kodlanmış sözde kodu

```
PozisyondakiElemanıDöndür Fonksiyonunu PARAMETRE i ile başlat
x dizisindeki (i mod lenX) indeksindeki elemanı döndür.
Fonksiyon sonu.
```

2.3. İşaret Maskesi ile Dallanmasız Programlama:

Bir işaret maskesi genellikle bir bit dizisi olan ve genellikle belirli bitleri seçmek, değiştirmek veya manipüle etmek için kullanılan bir değerdir. İşaret maskesi, genellikle bir sayının mutlak değerini elde etmek için kullanılan bir yöntemdir. X tam sayısı için, x'in işaret bitini bir tamsayı eksi birdeki bit sayısı kadar sağa kaydırarak bir maske oluşturulur. Maske, x negatifse 1'lerden oluşan yazmaç uzunluğundaki bir satır, x pozitif veya sıfırsa 0'lardan oluşan yazmaç uzunluğundaki bir satır olacaktır. Oluşturulan maske daha sonrasında X tam sayısına eklenir ve daha sonra sonuç maske değeri ile XOR işlemine tabi tutularak X tam sayısının mutlak değeri elde edilir. Bu şekilde işlem sayısının azaltılması hedeflenir.

Kod 5'te bulunan sözde kod, Kod 2'de yapılan işlemin optimize edilmiş halidir. İşlem sayısı azaltılmış ve çarpma gibi maliyeti çok olan işlemlerden kaçılmıştır. Bu şekilde dallanmasız olan bir kod bloğu olmasına rağmen Kod 2'deki sözde kod optimize edilmiştir.

Kod 5. İki sayıdan büyük olanı bulan metodun dallanmasız kodlanmış sözde kodun maske kullanılarak optimize edilmiş hali

```
BüyükSayıyıBul Fonksiyonunu PARAMETRELER a ve b ile başlat
a-b farkını hesapla ve fark'a ata
fark'ın en sağ bitini (31. bit) sağa kaydır ve mask değerine ata
SONUÇ olarak a – (fark & mask) değerini döndür
Fonksiyon sonu.
```

3. Uygulama

Kod 1'de bulunan sözde kodun 32 bitlik bir işlemci için x86 ASM dilinde cdecl (C deklerasyonu) türünde Intel sözdizimine sahip bir şekilde güncel bir derleyici tarafından optimize edilmiş dallanmalı ve dallanmasız kodlamaları sırasıyla Tablo 1 ve Tablo 2'de verilmiştir.

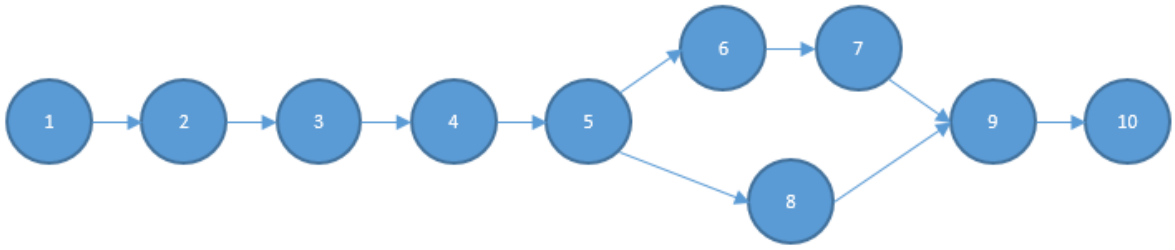
Tablo 1. Kod 1'in 32 bitlik bir işlemci için x86 ASM dilinde dallanmalı kodlanmış hali

Satır	Byte Code	Opcode
1	55	push ebp
2	8B EC	mov ebp,esp
3	8B 45 08	mov eax,[ebp+08]
4	3B 45 0C	cmp eax,[ebp+0C]
5	7E 07	jle Satır9

6	8B 45 08	mov eax,[ebp+08]
7	EB 05	jmp Satır10
8	8B 45 0C	mov eax,[ebp+0C]
9	5D	pop ebp
10	C3	ret

Tablo 1'deki kod bloğunun 10 satırdan oluşup toplamda 21 Byte uzunluğunda olduğu ve hafızada toplamda 84 bitlik yer kaplayacağı görülmektedir.

Şekil 1'de de görüldüğü üzere işlemci 1-2-3-4-5-6-7-9-10 veya 1-2-3-4-5-8-9-10 yollarından birini izlemek zorunda olacaktır. Dallanma işlemi 5. satırda gerçekleşecek olup 4. satırdaki kıyaslama işlemi tamamlanmadan işlemci hangi yolu seçeceğini bilemeyecektir.



Şekil 1. Tablo 1'deki kod bloğunda işlemcinin izleyeceği iş akışı.

Tablo 2'de bulunan kod bloğunda ise Tablo 1'deki 5. satırdaki koşullu atlama operasyonu yerine koşullu atama işlemi ile dallanmanın ortadan kaldırılarak işlemci durum bayrakları doğrudan atama işleminde kullanılmıştır. Böylelikle dallanmanın önüne geçilmiş olup hem işlemcinin işlemesi gereken kod satırı sayısı azaltılmıştır hem de toplam kod uzunluğu 15 Byte'a düşürülmüştür.

Tablo 2. Kod 1'in 32 bitlik bir işlemci için x86 ASM dilinde dallanmasız kodlanmış hali

Satır	**Byte Code**	**Opcode**
1	55	push ebp
2	8B EC	mov ebp,esp
3	8B 45 0C	mov eax,[ebp+0C]
4	39 45 08	cmp [ebp+08],eax
5	0F 4F 45 08	cmovg eax,[ebp+08]
6	5D	pop ebp
7	C3	ret

Tablo 3'te Tablo 1 ve Tablo 2'de bulunan kod bloklarına ait kodların 32 bitlik bir işlemci tarafından çalıştırılması durumunda ortaya çıkan sonuçlar bulunmaktadır. 2 numaralı kod bloğuna ait işlemlerin yaklaşık olarak 70% daha hızlı gerçekleştiği, hafızada kapladığı toplam alanın da yaklaşık olarak 25% daha az olduğu görülmektedir.

Tablo 3. Tablo 1 ve Tablo 2'deki kodların uygulamalı karşılaştırılması

Kod Bloğu Tablo Numarası	Toplam Satır	Toplam Boyut (Bit)	Geçen Süre (nanosaniye)
1	10	81	788
2	7	60	262

4. Sonuç ve Yorum

Dallanmasız programlamanın özellikle performans kritik sistemler için uygulanabileceği, doğru uygulanması halinde hem zaman açısından hem de alan açısından ciddi bir optimizasyon tekniği olduğu görülmektedir. Özellikle gerçek zamanlı sistemler ve akan veri uygulamaları için performans açısından kritik durumlarda dallanmasız programlama dallanmalarla ilgili ek yükü en aza indirerek ciddi performans kazanımları sağlayabilecektir.

Düşük kaynaklara sahip sistemlerde, Nİ sistemlerinde hem hızı hem de düşük depolama ihtiyacı ile kullanılan mevcut hafif siklet uygulamaların daha da hafifleştirilmesinde önemli bir rol oynayabilecektir. Bu sistemlerde gecikmeyi en aza indirmek ve enerji verimini en üst düzeye çıkarmak genellikle en önemli önceliklerden biridir ve dallanmasız programlama bu hedeflere ulaşılmasına yardımcı olabilecektir. Koşullu hareket yönergeleri, aritmetik işlemler, mantıksal operatörler gibi tekniklerin kullanılması ile daha verimli ve yüksek performanslı Nİ sistemleri oluşturulabilecektir. Bütün bunların yanında özellikle birçok kıyaslama işleminin gerçekleştiği şifreleme ve yetkilendirme algoritmaları ile beraber birçok ara katmanın bulunduğu makine öğrenmesi ve diğer yapay zeka teknikleri için de uygun koşullarda doğru şekilde uygulanması ile performans açısından gözle görülür iyileşmeler sağlayabilecektir.

Bu makalede bahsedilen ve kullanılan yöntemlerin birçok güncel işlemci için zaten derleme esnasında derleyici tarafından otomatik olarak gerçekleştirildiği bu teknikler kullanılarak yazılan kodların standartlaşmış kodlama tekniklerine göre daha az genişletilebilir ve okunabilir olduğu göz önünde bulundurulmalı, dallanmasız programlama teknikleri uygulanırken hedef platformun ve ortam şartlarının doğru seçilmesi gerektiği unutulmamalıdır.

Kaynaklar

Chowdhury, M. R., Tripathi, S., & De, S. (2020). Adaptive multivariate data compression in smart metering Internet of Things. IEEE Transactions on Industrial Informatics, 17(2), 1287-1297.

HaddadPajouh, H., Dehghantanha, A., Parizi, R. M., Aledhari, M., & Karimipour, H. (2021). A survey on internet of things security: Requirements, challenges, and solutions. Internet of Things, 14, 100129.

J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, 2013.

Khan, M. Z., Alhazmi, O. H., Javed, M. A., Ghandorh, H., & Aloufi, K. S. (2021). Reliable Internet of Things: Challenges and future trends. Electronics, 10(19), 2377.

Li, X., Liu, Y., Ji, H., Zhang, H., & Leung, V. C. (2019). Optimizing resources allocation for fog computing-based Internet of Things networks. IEEE Access, 7, 64907-64922.

Mangla, M., Kumar, A., Mehta, V., Bhushan, M., & Mohanty, S. N. (Eds.). (2022). Real-life applications of the Internet of Things: Challenges, applications, and advances.

Qadri, Y. A., Nauman, A., Zikria, Y. B., Vasilakos, A. V., & Kim, S. W. (2020). The future of healthcare internet of things: a survey of emerging technologies. IEEE Communications Surveys & Tutorials, 22(2), 1121-1167.

R. Want and S. Dustdar, "Activating the internet of things [guest editors' introduction]," Computer, vol. 48, no. 9, pp. 16–20, 2015.

Wang, Z., Liu, Y., Sun, Y., Li, Y., Zhang, D., & Yang, H. (2015, May). An energy-efficient heterogeneous dual-core processor for Internet of Things. In 2015 IEEE international symposium on circuits and systems (ISCAS) (pp. 2301-2304). IEEE.

Zafar, S., Bhatti, K. M., Shabbir, M., Hashmat, F., & Akbar, A. H. (2022). Integration of blockchain and Internet of Things: Challenges and solutions. Annals of Telecommunications, 1-20.