

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OSİLATÖR DEVRE SİMÜLASYONLARI VE SONUÇLARIN
OCTAVE'DE İŞLENMESİ

EYÜP ÇUĞALIR
YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

GEBZE
2019

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OSİLATÖR DEVRE SİMÜLASYONLARI VE
SONUÇLARIN OCTAVE’DE İŞLENMESİ

EYÜP ÇUĞALIR
YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
YRD. DOÇ. DR. ÖNDER ŞUVAK

GEBZE
2019

T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OSCILLATOR CIRCUIT SIMULATIONS
AND PROCESSING OF RESULTS IN
OCTAVE

EYÜP ÇUĞALIR
A THESIS SUBMITTED FOR THE DEGREE OF
MASTER OF SCIENCE
DEPARTMENT OF ELECTRONIC ENGINEERING

THESIS SUPERVISOR
ASSIST. PROF. DR. ÖNDER ŞUVAK

GEBZE
2019



YÜKSEK LİSANS JÜRİ ONAY FORMU

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
...10/07/2019... tarih ve 2019./...31... sayılı kararıyla oluşturulan
jüri tarafından 06./...11./2019 tarihinde tez savunma sınavı yapılan Eyüp
Çuğalır'ın tez çalışması Elektronik Mühendisliği Anabilim Dalında YÜKSEK
LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE

(TEZ DANIŞMANI) : Dr. Öğr. Üyesi Önder Şuvak

ÜYE

: Dr. Öğr. Üyesi Mustafa Berke Yelten

ÜYE

: Doç. Dr. Koray Kayabol

ONAY

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
..... tarih ve/..... sayılı kararı.

İMZA/MÜHÜR

ÖZET

Bu çalışmada, bir devre benzetim aracı olan Ngspice’ın Octave aracılığıyla kontrol edilebilmesine olanak sağlamaktır. Bu çalışma ile birlikte, benzetim sonucunda elde edilen grafikler üzerinde daha geniş kontrol imkanı sağlanmış ve benzetim verileri üzerinde ileri işleme yapabilme yeteneği kazanılmıştır. Benzetim yapılırken parametrik analiz yapılmıştır. Elektronik devrelerin belirsizlik analizleri, aynı devrenin farklı parametreler ile tekrar tekrar benzetime tabi tutulmasını ve sonuçların üzerinde birtakım işlemler yapılmasını gerektirir. İleri işleme için kullanılacak prototip oluşturma ortamının Octave olarak düşünüldüğü bu gibi bir durumda, Octave ortamında üretilen değişken parametre takımının her bir elemanının devre netlistine yazılması, benzetimin gerçekleştirilmesi ve elde edilen sonuçların ileri işleme yapılmak üzere Octave ortamına aktarılması gerekir. Söz konusu işlemlerin bir otomasyon aracılığıyla gerçekleştirilmemesi önemli ölçüde zaman kaybına neden olabilmektedir. Belirsizlik analizi örneği, elektronik devrelerin benzetiminin gerçekleştirildiği ve akabinde sonuçlar üzerinde ileri işleme yapıldığı senaryolardan yalnızca biridir. Bu çalışmada, söz konusu senaryolarda kullanılmak üzere, açık kaynak koda sahip bir elektronik devre benzetim aracı olan Ngspice’ın Octave tarafından kontrol edilebilmesine olanak tanıyan Ngspice bağlantı aracının tasarım ayrıntıları sunulmaktadır. Tasarım esnasında Ngspice’ın sunmuş olduğu “callback” fonksiyonlarından yararlanılmış, “callback” sonrasında elde edilen verinin organize edilebilmesi için bir C++ sınıf yapısı kullanılmış ve Octave tarafına aktarılacak verinin saklanması için mex data tiplerinden faydalanılmıştır. Bununla birlikte, mex kütüphanesinin içe aktarıldığı ilgili proje dosyasını derlemek/link etmek üzere makefile projesinin mex paradigmaları kullanılmıştır. Tasarlanan bu araç sayesinde, Octave tarafından manipüle edilen devre netlistleri benzetim aracına gönderilebilmekte ve diske herhangi bir veri yazımı olmadan sonuçlar Octave tarafına aktarılabilir.

Anahtar Kelimeler: Osilatör, Parametrik Analiz, LTSpice, NGSpice, Mex, Octave.

SUMMARY

In this work, a wider control is provided on the graphs obtained as a result of simulation and the ability of further processing on the simulation data is gained. Parametric analysis was performed during simulation. The uncertainty analysis of electronic circuits requires that the same circuit be simulated over and over again with different parameters and that a number of operations are performed on the results. In such a case where the prototyping environment to be used for further processing is considered Octave, each element of the variable parameter set produced in the Octave environment must be written to the circuit netlist, the simulation performed, and the results obtained transferred to the Octave environment for further processing. Failure to perform such operations by automation can result in considerable time loss. The example of uncertainty analysis is only one of the scenarios in which electronic circuits are simulated and then further processed on the results. In this study, the design details of the Ngspice connection tool, which allows the control of Ngspice, an electronic circuit simulation tool with open source code, can be controlled by Octave for use in these scenarios are presented. Design during that submitted the ngspice "callback" function has been utilized, "callback" is a C++ class structure subsequent to organize the data obtained were used and have been utilized mex data type for storing the data to be transferred to Octave side. However, the mex paradigms of the makefile project were used to compile / link the corresponding project file into which the mex library was imported. Thanks to this designed tool, circuit netlists manipulated by Octave can be sent to the simulation tool and the results can be transferred to Octave without writing any data to disk.

Key Words: Oscillator, Parametric Analysis, LTSpice, NGSpice, Mex, Octave.

TEŞEKKÜR

Başta, yüksek lisans eğitimimde ve akademik hayatımda desteğini ve yardımlarını hiçbir zaman esirgemeyip bilgisi ile bu çalışmanın oluşmasının yolunu açan danışmanım Yrd. Doç. Dr. Önder ŞUVAK'a,

Bütün çalışmam boyunca yanımda olan, bilgi ve tecrübelerini benimle paylaşan değerli çalışma arkadaşım Mehmed Akif PAKSOY'a ve vermiş olduğu desteklerinden dolayı değerli eşime ve aileme en içten teşekkürlerimi sunarım.

İÇİNDEKİLER

	<u>Sayfa</u>
ÖZET	v
SUMMARY	vi
TEŞEKKÜR	vii
İÇİNDEKİLER	viii
SİMGELER ve KISALTMALAR DİZİNİ	ix
ŞEKİLLER DİZİNİ	x
TABLolar DİZİNİ	xii
1. GİRİŞ	1
1.1. Osilatör Devrelerinin Kullanım Alanları	2
1.2. Osilatör Devrelerinin Sınıflandırılması	2
1.3. Osilatör Devrelerinin Çalışma Prensipleri	3
2. YÖNTEM	6
2.1. Octave-Ngspice Bağlantısının Amaç ve Kapsamı	6
2.2. Ngspice- MEX Bağlantısı	7
2.3. MEX- Octave Bağlantısı	8
2.4. Octave-Ngspice ve Octave’de Yapılan Çalışmalar	10
2.4.1. RC Osilatör	11
2.4.2. Colpitts Osilatör	14
2.4.3. Hartley Osilatör	17
2.4.4. Armstrong Osilatör	20
3. SONUÇLAR ve ÖNERİLER	24
KAYNAKLAR	25
ÖZGEÇMİŞ	26
EKLER	27

SİMGELER ve KISALTMALAR DİZİNİ

<u>Simgeler ve</u> <u>Kisaltmalar</u>	<u>Açıklamalar</u>
f	: Osilatör çalışma frekansı
A_v	: Amplifikatör kazancı
β	: Kazanç katsayısı
C	: Kondansatör
L	: İndüktör
R	: Direnç
RC	: Direnç ve kondansatör
N	: RC katlarının sayısı
M	: Bobinlerin birbirine etki katsayısı
FET	: Field Effect Transistor
MOSFET	: Metal Oxide Semiconductor Field Effect Transistor

ŞEKİLLER DİZİNİ

<u>Sekil No:</u>	<u>Sayfa</u>
1.1: Osilatör devreleri çıkış örnekleri.	1
1.2: Osilatör blok şeması.	4
2.1: Geri döndürülen fonksiyonlar.	7
2.2: Kullanıcının Ngspice kütüphane kullanma yöntemi.	8
2.3: C++/C geçiş şeması.	8
2.4: Mex giriş parametreleri.	9
2.5: Octave benzetim başlatma komutları.	9
2.6: Paylaşılan kütüphanenin C++ nesnesine dönüşme şeması.	9
2.7: Bellek kaçaklarını önlemek için komutlar.	10
2.8: Octave-Ngspice genel bağlantı şeması.	11
2.9: RC devre şeması.	12
2.10: $C_1 = 20\text{nF}$ için RC devresi benzetim sonucu.	13
2.11: $C_1 = 30\text{nF}$ için RC devresi benzetim sonucu.	13
2.12: $C_1 = 40\text{nF}$ için RC devresi benzetim sonucu.	14
2.13: $C_1 = 50\text{nF}$ için RC devresi benzetim sonucu.	14
2.14: Colpitts devre şeması.	15
2.15: $C_1 = 3\text{nF}$ için Colpitts devresi benzetim sonucu.	16
2.16: $C_1 = 5\text{nF}$ için Colpitts devresi benzetim sonucu.	16
2.17: $C_1 = 7\text{nF}$ için Colpitts devresi benzetim sonucu.	17
2.18: $C_1 = 9\text{nF}$ için Colpitts devresi benzetim sonucu.	17
2.19: Hartley devre şeması.	18
2.20: $C_1 = 0.1\mu\text{F}$ için Hartley devresi benzetim sonucu.	19
2.21: $C_1 = 0.2\mu\text{F}$ için Hartley devresi benzetim sonucu.	19
2.22: $C_1 = 0.3\mu\text{F}$ için Hartley devresi benzetim sonucu.	20
2.23: $C_1 = 0.4\mu\text{F}$ için Hartley devresi benzetim sonucu.	20
2.24: Armstrong devre şeması.	21
2.25: $C_1 = 30\text{nF}$ için Armstrong devresi benzetim sonucu.	22
2.26: $C_1 = 60\text{nF}$ için Armstrong devresi benzetim sonucu.	22
2.27: $C_1 = 90\text{nF}$ için Armstrong devresi benzetim sonucu.	23

2.28: $C_1 = 120\text{nF}$ için Armstrong devresi benzetim sonucu.

23

TABLÖLAR DİZİNİ

Tablo No:

Sayfa

1.1: Osilatör türleri.

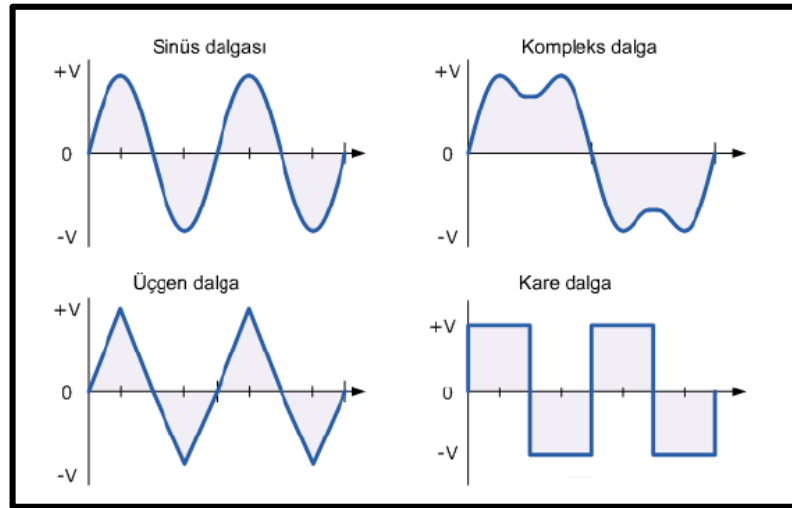
3

1. GİRİŞ

Osilatörler sinyal üretebilen elektronik devre elemanlarıdır. DC seviyeden darbe üretimine kadar geniş bir yelpazede üretim yapabilirler. Saniyedeki değişim osilatörün frekansını belirler. Dört basit osilatör dalga tipi vardır: sinüs, kare dalga, testere dişi dalga, üçgensel dalga. Bazı devrelerde çalışma frekansını değiştirip dalga tipini değiştirebilir.

Elektronik devrelerde belli bir zaman içinde değişen yönü ve şiddeti belli bir disiplinde hareket eden sinyallere osilasyon denir. Devreleri kara kutu olarak gördüğümüzde çıkıştaki sinyalin osilasyon binmemiş temiz bir sinyal olması çoğu zaman gereklidir. Bu amaçla çıkış osilasyonlarını azaltmak için elektronik devrelerde aktif veya pasif filtreler kullanılır. Doğru akım kaynağından aldığı akımı bir başka devreyi yüksek güçte beslemek için alternatif akıma çeviren osilatörlere ise invertör yani güç çevirici adı verilir.

Mikrodenetleyici saati, darbe üretici devreler gibi sistemlerde osilasyona ihtiyaç vardır. Bu nedenle osilasyon sinyalleri üreten osilatör devreleri tasarlanır ve kullanılır[1]. Genel olarak Osilatör devrelerinin çıkış sinyalleri aşağıdaki şekilde gibidir:



Şekil 1.1: Osilatör devreleri çıkış örnekleri.

1.1. Osilatör Devrelerinin Kullanım Alanları

Osilatör 1892 yılında ilk olarak Elihu Thomson tarafından keşfedilmiş olup, 1900 yılında William Duddell'in çalışmaları ile geliştirilmiştir. Duddell Osilatör kullanarak ses dalgaları üretmeyi başarmıştır. Daha sonra 1902 yılında Valdemar Poulsen Duddell'in çalışmalarını devam ettirerek 1920 yılında ses dalgaları üreten osilatörü geliştirerek, ilk frekans modülasyonu yapan alıcı verici cihazını tasarlamıştır. Bu gelişmelere ek olarak 'radyonun babası' olarak tanınan Lee de Forest sinema filmlerinin üzerine ses kaydını geliştirmiştir, osilatörü telsiz telgraf sistemi için de kullanmıştır. Günümüzde kullandığımız modern mikrodalga frekansında yayılım yapan osilatörler 1969 yılında K. Kurokawa tarafından hayata geçirilmiştir.

Genel olarak osilatörler, sinüsoidal ve sinüsoidal olmayan (kare, üçgen dalga) osilatörler olarak ikiye ayrılmıştır. Bunların dışında; Armstrong, Colpits, Hartley, köprü osilatör, kristal osilatör gibi özel osilatör çeşitleri de bulunmaktadır. Bu kadar çok osilatör çeşidinden işimize yarayacak olanı seçmek için kalite faktörü ve devreye uyumu göz önünde bulundurulmalıdır[2].

Osilatörler kontrol ve otomasyon sistemlerinde, elektronik ve haberleşme sistemlerinde (özellikle kablosuz haberleşme sistemlerinde modülasyon için yüksek frekanslı taşıyıcı dalga üretimi için), sinyal jeneratörlerinde, lojik devre tasarımlarında gördüğümüz clock (kare dalga) sinyallerinde kullanılırlar.

1.2. Osilatör Devrelerinin Sınıflandırılması

Osilatör türleri aşağıdaki Tablo 1.1'de görüldüğü gibi RC tipi osilatörler, LC tipi osilatörler ve sinusoidal olmayan osilatörler olarak üç ana gruba ayrılır.

Sabit frekanslı osilatör devrelerinde RLC devre elemanları ile yapılan devrelerdir. Buna alternatif olarak ve kristalli osilatör çeşitlerinin istenmeyen bazı kusurları vardır. Bunun yanı sıra basit devre elemanları ile yapılan osilatör devrelerinden üstünlükleri de mevcuttur. Radyo alıcılarının istasyon ayar düğmesi bir değişken kondansatöre bağlıdır. Düğmenin döndürülmesiyle bu kondansatörün değeri ve (bir bobin ile bir kondansatörden ibaret olan LC devresi) rezonans devresinin tepki gösterdiği frekans aralığını değiştirilir. Herhangi bir radyo

dalgasının frekansı vericideki osilatörün meydana getirdiği elektron akışının frekansının aynısıdır. Radyo ve televizyon sinyallerin üzerine ses ve görüntü işareti bindirilmiş olduğu dalgaının frekansı sabit olmalıdır. Ancak yukarıda bahsi geçen osilatörler bu kadar sabit frekanslı elektron osilasyonları verecek kabiliyette değildir. Vericilerde kullanılan osilatörler sabit frekanslı osilatörlerdir.

Tablo 1.1: Osilatör türleri.

RC Tipi Osilatörler	LC Tipi Osilatörler	Sinusoidal Olmayan Osilatörler
Wien Köprü Osilatörü	Armstrong Osilatör	Kare Dalga Osilatör (Multivibratör)
Faz Kaydırmalı Osilatör	Collpits Osilatör	Üçgen Dalga Osilatör
	Hartley Osilatör	
	Clapp Osilatör	
	Kristal Kontrollü Osilatör	

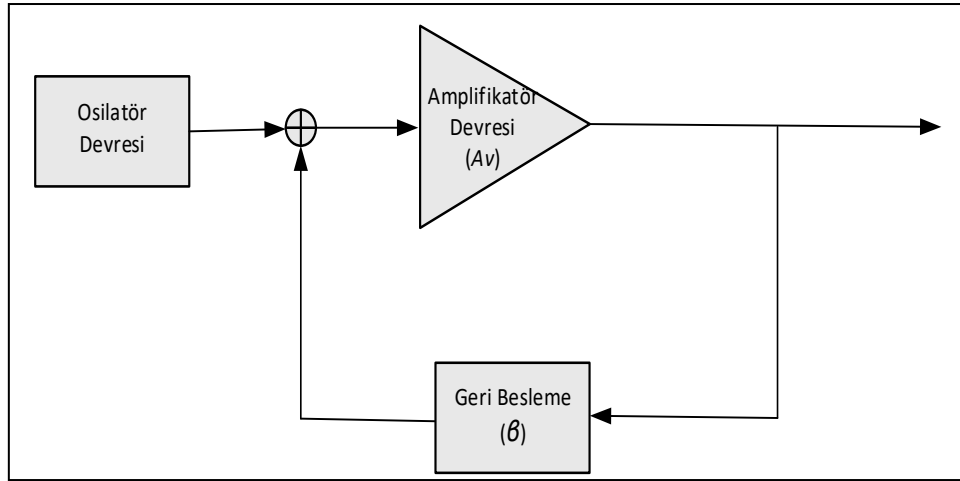
Kristalli osilatörler basit devre elemanları ile yapılmış devrelerin frekans karakteristiğinin geniş olması kusurunu ortadan kaldırır. Kristal osilatörün çalışma mantığı şu şekilde tanımlanabilir; ince kuartz diliminin belli bir frekansta mekanik titreşimler sonucu oluşan periyodik sinyaller sayesinde osilasyon yapmasıdır. Temel frekans harici devamında bulunan periyodik sinyaller temel frekanstan ayrılarak zararlı etkileri önlenmiş olur. Titreşimlerin frekansı kuarts diliminin ne kadar ince kesildiği ile alakalıdır. Frekans kuartz diliminde kararlılıkla sabit kalır. İnce kesilmiş bir kuartz dilimi sıkıştırılıp gevşetilerek karşılıklı yüzeyler arası bir elektrik gerilimi oluşturulur. Oluşturulan bu gerilimle titreşim yaptırılırsa bir alternatif gerilim elde edilir. Böylece kuartz kristali osilatör devrelerinde kullanılan kondansatör ve bobinden oluşan rezonans devrelerde kullanılabilir [3].

1.3. Osilatör Devrelerinin Çalışma Prensibi

Osilasyon kriterleri için öncelikli olarak iki kapılı kapalı bir devre elemanına ihtiyaç vardır. Geri besleme yapılırken herhangi bir faz farkı olmaması için faz açısının sıfır olması gerekir. Kapalı çevrim kazancının da 1'e eşit olması gerekir.

Osilatörlere kendi kendine sinyal üretebilen cihazlar tanımlaması yaparsak pek de yanlış sayılmaz. Osilatörler bir kez çalışmaya başladıktan sonra alternatif akım sinyali üretmeye başlarlar. Bu sinyalin bir kısmı girişe geri beslenip, yükseltilirler ve çıkışa geri gönderilerek bir geri besleme yapısı oluşturulur. Osilatörler DC güç kaynağındaki elektrik enerjisini, AC elektrik enerjisine belirli bir frekansta harici sinyal uygulamadan transfer ederler, aynı zamanda devrede bulunan elemanların toleranslarından faydalananarak belli bir zaman sonra kendi kendine osilasyon haline geçebilmektedir. Osilasyon oluşturmak için aşağıdaki şartları sağlamak gereklidir;

- Pozitif geri besleme
- $1 + A_v * \beta = 0$



Şekil 1.2: Osilatör blok şeması.

Şekil 1.2’de görüldüğü gibi bir osilatör, çıkış sinyalinin bir kısmını tekrar girişine geri besleyen bir amplifikatörden oluşturulur. Bu, şekilde gösterildiği gibi herhangi bir harici sinyal girişine ihtiyaç duymadan bir sinyal üretmesini sağlayacak şekilde yapılır. Bir DC kaynağını bir AC sinyaline dönüştürmenin bir yolu olarak da düşünülebilir. Bir osilatörün amplifikatör bölümündeki geri besleme pozitif olmalıdır. Böylece amplifikatörün çıkış sinyalinin bir kısmının giriş ile aynı fazda olacak şekilde geri beslenmesi ve geri besleme ile giriş sinyallerinin bir araya getirilmesiyle giriş sinyalinin genliğinin artırıldığı durumdur. Örneğin; ortak bir yayıcı yükseltici, giriş ve çıkış arasında 180° 'lik bir faz değişikliği yaratır, bu nedenle pozitif geri besleme döngüsü, pozitif geri besleme için çıkıştan girişe geri beslenen sinyalde 180° 'lik bir faz değişikliği üretmelidir.

Amplifikatörlerdeki küçük miktarda pozitif geri beslemenin sonucu, artan gürültü ve bozulma maliyetine rağmen daha yüksek kazançtır. Ancak, pozitif geri besleme miktarı yeterince büyükse, sonuç, yükselteç devresinin kendi sinyalini ürettiği salınımdır.

LC osilatörler üzerinden osilatörlerin çalışma mantığına örnek verecek olursak, yapılması gereken bir kondansatör ve bir indüktör birbirine bağlamak yeterli olacaktır. Her iki elemanda enerji depolama riskine sahiptir. Enerji verildikten sonra kapasitör şarj olur, kapasiteye seri olacak şekilde bir indüktör bağlanırsa kapasite indüktör üzerinden boşalır. Kapasite üzerinde elektrik alan oluşurken indüktör üzerinde ise manyetik alan oluşumuna neden olacaktır. Kapasite boşaldıktan sonra indüktör akımı devam ettirip kapasitörün diğer tabakasını şarj eder[4].

2. YÖNTEM

Bu bölümde sistemler arası geçişlerde kullanılan bağlantıların nasıl gerçekleştirildiği anlatıldı. Ngspice bağlantı aracı, paylaşılan bir kütüphane olarak derlenebilen Ngspice'in, diğer C/C++ programları ile haberleşebilmesine olanak sağlayan tip ve fonksiyon tanımlarının bulunduğu ".h" dosyalarını kullanan bir mex projesidir. Mex'in C++ sınıflarının Octave' in anlayacağı sınıflara çevrilerek haberleşmesi sayesinde tüm haberleşme işlemleri gerçekleştirildi.

2.1. Octave-Ngspice Bağlantısının Amaç ve Kapsamı

Bu çalışmada, daha çok sinüs dalgası üretmeye yoğunlaşıldı. Frekansa etki eden devre elemanlarını parametrik olarak değiştirip bunun periyoda ve dalga şekline etkisini gözlemlendi. Bu bağlamda gerçekleştirilen ilk çalışma C tabanlı bir benzetim programı olan Ngspice'in Octave aracılığıyla kontrol edilmesine olanak tanımaktadır. Elektronik devrelerin belirsizlik analizleri, aynı devrenin farklı parametreler ile tekrar tekrar benzetime tabi tutulmasını ve sonuçların üzerinde birtakım işlemler yapılmasını gerektirir. İleri işleme için kullanılacak prototip oluşturma ortamının Octave olarak düşünüldüğü bu gibi bir durumda, Octave ortamında üretilen değişken parametre takımının her bir elemanının devre netlistine yazılması, benzetimin gerçekleştirilmesi ve elde edilen sonuçların ileri işleme yapılmak üzere Octave ortamına aktarılması gerekir. Söz konusu işlemlerin bir otomasyon aracılığıyla gerçekleştirilmemesi önemli ölçüde zaman kaybına neden olabilmektedir. Belirsizlik analizi örneği, elektronik devrelerin benzetiminin gerçekleştirildiği ve akabinde sonuçlar üzerinde ileri işleme yapıldığı senaryolardan yalnızca biridir. Bu çalışmada, söz konusu senaryolarda kullanılmak üzere, açık kaynak koda sahip bir elektronik devre benzetim aracı olan Ngspice'in Octave tarafından kontrol edilebilmesine olanak tanıyan Ngspice bağlantı aracının tasarım ayrıntıları sunulmaktadır. Tasarım esnasında Ngspice'in sunmuş olduğu "callback" fonksiyonlarından yararlanılmış, "callback" sonrasında elde edilen verinin organize edilebilmesi için bir C++ sınıf yapısı kullanılmış ve Octave tarafına aktarılacak verinin saklanması için Mex data tiplerinden faydalanılmıştır. Bununla birlikte, Mex kütüphanesinin içe aktarıldığı ilgili proje dosyasını derlemek/link etmek üzere

“makefile” projesinin Mex paradigmaları kullanılmıştır[5]. Tasarlanan bu araç sayesinde, Octave tarafından değiştirilen devre netlistleri benzetim aracına gönderilebilmekte ve diske herhangi bir veri yazımı olmadan sonuçlar Octave tarafına aktarılabilir. Çalıştırılan devreler için parametrik analiz, periyod fonksiyonu bulma yöntemi ve osilatörün kendi kendine başlamaması veya uzun süre sonra başlamasının önüne geçmek için osilatör prob devresi benzetim yapılan devrelerin öncesine konulmuştur. Parametrik analiz devre içinde kullanılan osilasyona direk etki eden elemanlar için kullanılmış, sonucunda periyodun nasıl değiştiği gözlemlenmiş ve sonuç olarak kaydedilmiştir.

2.2. Ngspice- MEX Bağlantısı

C++ tarafında, birkaç konfigürasyon işleminin ardından yapılan ilk iş, Ngspice tarafından üretilen verilere ulaşabilmek amacıyla tanımlı "callback" fonksiyonlarından hangilerinin kullanılacağını kütüphaneye bildirmektir. Bu tez kapsamında, benzetim başlamadan hemen önce mevcut benzetimde üretilecek tüm vektörler hakkında bilgileri geri döndüren "callback" fonksiyonu kullanılmaktadır [6].

```
typedef int (SendInitData)(pvecinfoall, int, void*)  
pvecinfoall* pointer to array of structs containing data from all vectors right after initialization  
int identification number of calling ngspice shared lib  
void* return pointer received from caller  
send back initialization vector data.
```

Şekil 2.1: Geri döndürülen fonksiyonlar.

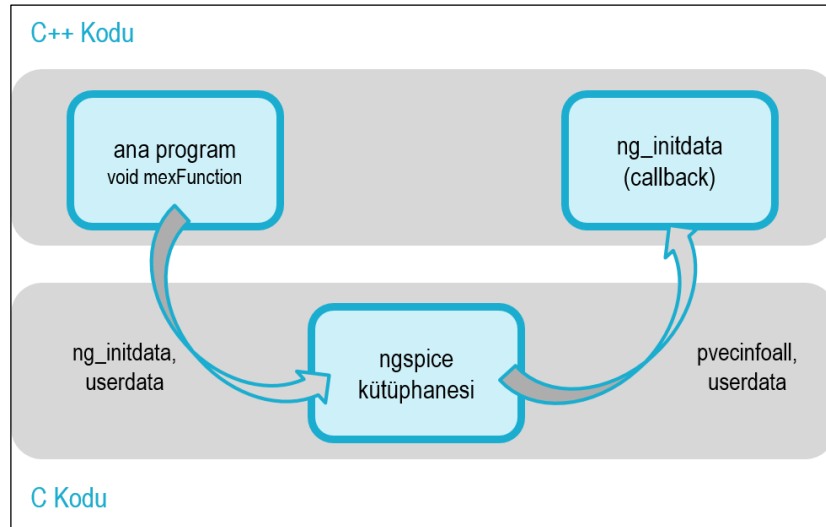
Kütüphane, içeriği hakkında bilgi sahibi olmadığı bu fonksiyonu benzetimin hemen öncesinde yukarıda gösterilen argümanlarla çağırır. “pvecinfoall” vektörler hakkında bilgileri içeren yapı dizisine ait işaretçidir (“pointer”). “pvecinfoall” ile döndürülen bilgileri saklamak ve ana program içinde kullanabilmek üzere “callback” fonksiyonunun içerisinde kullanıcı tarafından tanımlanan bir konteynerde bulunmalıdır. İlgili konteyner “callback” fonksiyonunun 3. argümanı olan “void pointer”dır. Ngspice kütüphanesi, kullanılacak “callback” fonksiyonlarının bildirilmesi esnasında kullanıcının tanımladığı bir sınıfa ait nesnenin adresinin de

bildirilmesine olanak sağlamaktadır [7]. Bu destek sayesinde kullanıcı, nesneyi ana program içinde üretebilir ve "static_cast" ile ona ait işaretçinin tipini "void*" olarak değiştirerek kütüphaneye bildirebilir.

```
void * userdata = static_cast<void*>( currentPlot_ob );
ret = ngSpice_Init( NULL, NULL, ng_exit, NULL, ng_initdata, NULL, userdata );
```

Şekil 2.2: Kullanıcının Ngspice kütüphane kullanma yöntemi.

Şekil 2.2’de ayrıca görülebildiği üzere tanımlı olan ancak kullanılmayacak “callback” fonksiyonları için NULL bildirimi yapılır. Bir hata oluşması durumunda hatayı kullanıcıya iletecek exit “callback” fonksiyonunun ise bildirilmesi zorunludur. Nihayetinde kullanılmak istenen “callback” fonksiyonları ve kullanıcı verisi (userdata) kütüphaneye bildirilir, benzetim başlamadan hemen önce kütüphane vektör bilgilerini ve kullanıcı verisini giriş argümanları olarak kabul eden “ng_initdata” fonksiyonunu çağırır.



Şekil 2.3: C++/C geçiş şeması.

2.3. MEX- Octave Bağlantısı

Mex projelerine ait giriş noktası aşağıda gösterildiği gibidir ve giriş parametreleri olarak sırasıyla çıkış argümanları sayısı, çıkış argümanları dizisi, giriş argümanları sayısı ve giriş argümanları dizisini içerir.

```
void mexFunction
(
    int         nlhs ,
    mxArray * plhs[] ,
    int         nrhs ,
    const mxArray * prhs[]
)
```

Şekil 2.4: Mex giriş parametreleri.

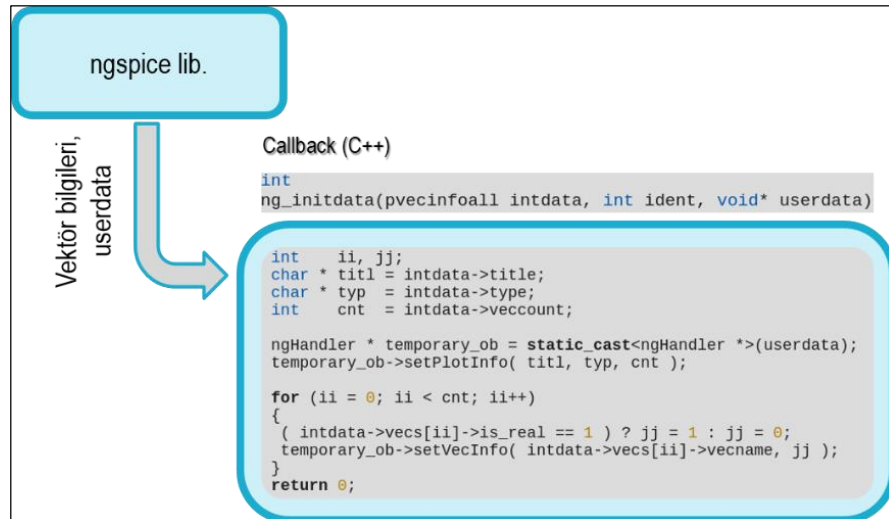
Octave tarafından yollanan bilgi prhs ile alınır ve sonuç plhs'e yazılarak Octave tarafından erişilebilir duruma getirilir. Tez kapsamında prhs ile gönderilen veri benzetimi yapılacak devrenin netlistine ait dizin bilgisi iken plhs ile alınan veri ise benzetim verisi olacaktır.

Gerekli geri bildirimlerin yapılmasının ardından, Octave tarafından gönderilen netlist dizin bilgisi ve benzetimi başlatma komutu gönderilir. Bu komutlar şöyledir.

```
filepath = mxArrayToString( prhs[0] );
ret = ngSpice_Command( filepath );
ret = ngSpice_Command( "bg_run" );
```

Şekil 2.5: Octave benzetim başlatma komutları.

Benzetime başlanmadan hemen önce ("bg_run" gönderildikten sonra) "callback" fonksiyonu gerekli argümanlar ile çağrılır.



Şekil 2.6: Paylaşılan kütüphanenin C++ nesnesine dönüşme şeması.

Şekil 2.6’de görüldüğü gibi, “callback” fonksiyonu içinde, “userdata” tekrar bir C++ sınıf tipi işaretçiye dönüştürülür ve kütüphaneden gönderilen bilgiler (oluşturulan vektör sayısı, isimleri) ilgili alanlara yazılır.

Benzetimin tamamlanmasının ardından ana programa dönüldüğünde ana programda hala “void*” tipinde olan “userdata”, “dynamic cast” ile orijinal tipine dönüştürülür ve ilgili alanlardaki bilgiler bir döngü içinde “ngGet_Vec_Info” fonksiyonuna giriş argümanı olarak verilir. Bu aşamadan sonra Octave’e gönderilecek tüm gerekli bilgiler “plhs[0]”da oluşturulan mex yapı dizisine (mex structure array) kopyalanır.

Kopyalama işleminin ardından bellek temizleme gerçekleştirilir. Aynı netlist dosyasının çok kez benzetimini gerektiren analizler için bellek kaçağını (memory leakage) büyük oranda önlemek amacıyla “destroy all” komutunun gönderilmesi son derece önemlidir.

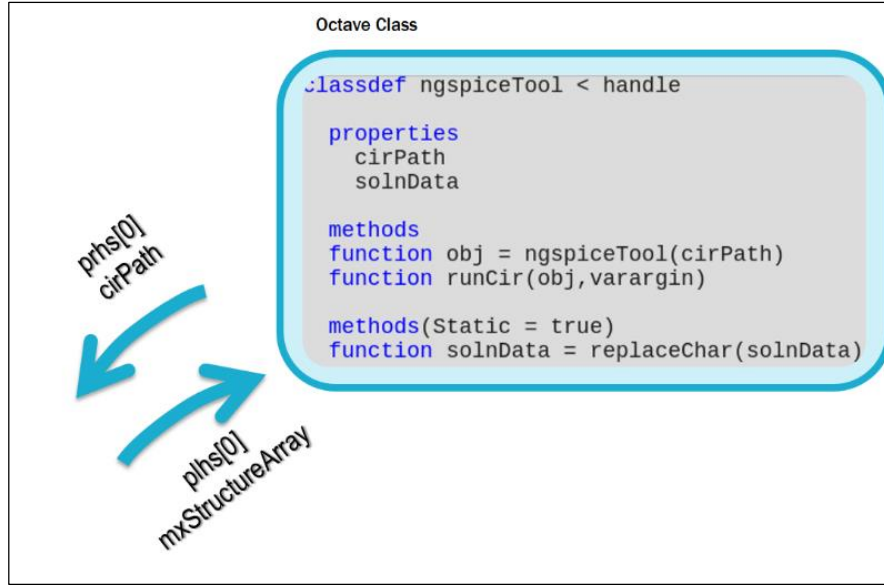
```
mxFree(filepath);  
delete(ob);  
ret = ngSpice_Command( "destroy all" );
```

Şekil 2.7: Bellek kaçaklarını önlemek için komutlar.

2.4. Octave-Ngspice ve Octave’de Yapılan Çalışmalar

Oluşturulan mex dosyasına bir Octave scripti aracılığıyla doğrudan ulaşabilmek mümkündür. Buna rağmen tekrar eden bazı ek görevleri arka plana itmek ve bir miktar işlevsellik katmak amacıyla “handle” yapısına sahip bir Octave sınıfı oluşturulmuştur [8].

İlgili sınıfın bir nesnesi, incelenecek devrenin netlist dizin bilgisi kullanılarak oluşturulur ve bu bilgi “cirPath” alanına yazılır. Akabinde “runCir” fonksiyonu, ilgili nesne ile çağrılır ve benzetim başlatılır. Bu fonksiyona opsiyonel olarak saklanması istenen vektör adları ek bir argüman olarak verilebilir. Bu koşulda benzetim verisinden yalnızca istenen vektörler ve zaman vektörü saklanır. Her iki durumda da Octave’e gönderilen benzetim verisi “replaceChar” fonksiyonuna girdi olarak verilir ve vektör adlarından " # " , " (" , ") " , " . " gibi karakterler çıkarıldıktan sonra “solnData” üzerine yazılır. İlgili işlemin ardından benzetim verisi ileri işleme uygulamaları için kullanılabilir durumdadır.



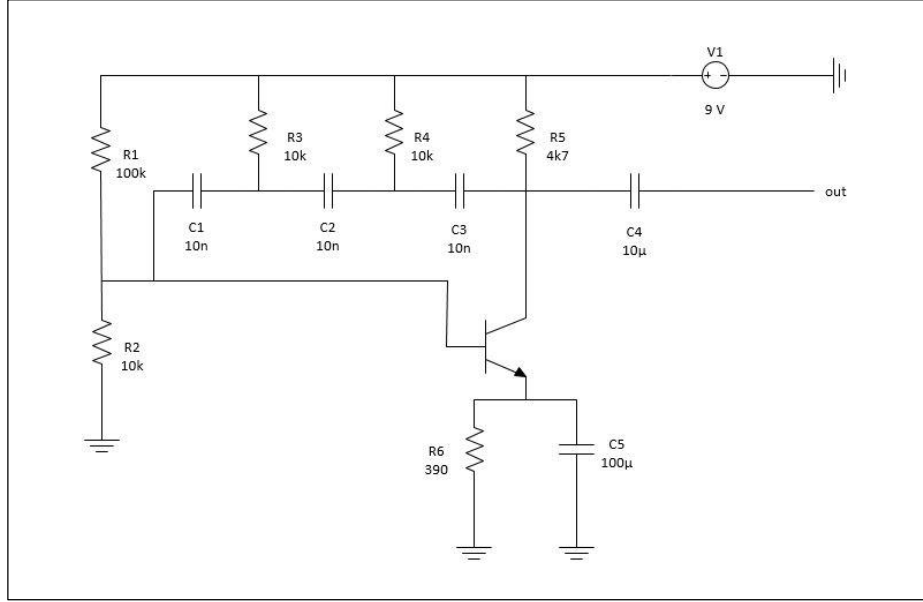
Şekil 2.8: Octave-Ngspice genel bağlantı şeması.

Parametrik analiz için tasarlanmış olan RC, Collpits, Hartley ve Armstrong osilatör devreleri kullanılmıştır. Devre tasarımları önce Ltspice programında test edilmiş daha sonra Ngspice'a göre netlistleri yazılmıştır. Parametrik analizler devrelerde osilasyona doğrudan etki eden kapasite veya bobinlerde uygulanmıştır. Bu analizler sonucunda devrelerin periyodunun nasıl etkilendiği gözlemlenmiştir. Armstrong osilatör devresinde parametrelerin değişimi sonucunda osilasyona girememe durumu ortaya çıktığından devrenin öncesine osilasyon prob devresi tasarlanıp koyulmuştur. Periyod bulurken; benzetim sonucu oluşan vektörün minimum ve maximum değerlerinin ortalaması alınır, bu ortalama bizim tahmin ettiğimiz ortalama değerle karşılaştırılıp devrenin osilasyona tam olarak girdiği yer tespit edilir. Bundan sonraki ilk iki maximum noktası arası bizim periyodumuz olur [9]. Kontrolleri daha sonra grafik üzerinden ölçüm yaparak da test edilebilir.

2.4.1. RC Osilatör

Bir RC Osilatöründe, geri bildirim dalındaki RC elemanlarını kullanarak, örneğin bir RC ağına giriş ile aynı ağdan çıkış arasında bir faz kaymasının meydana gelmesi gerçeğinden yararlanılır. Denklem 2.1'de RC osilatörün frekansını hesaplama yöntemi verilmiştir. Burada R direnç, C kapasitans, N ise R-C katlarının sayısını ifade eder [10].

$$f = \frac{1}{2\pi RC\sqrt{N}} \quad (2.1)$$

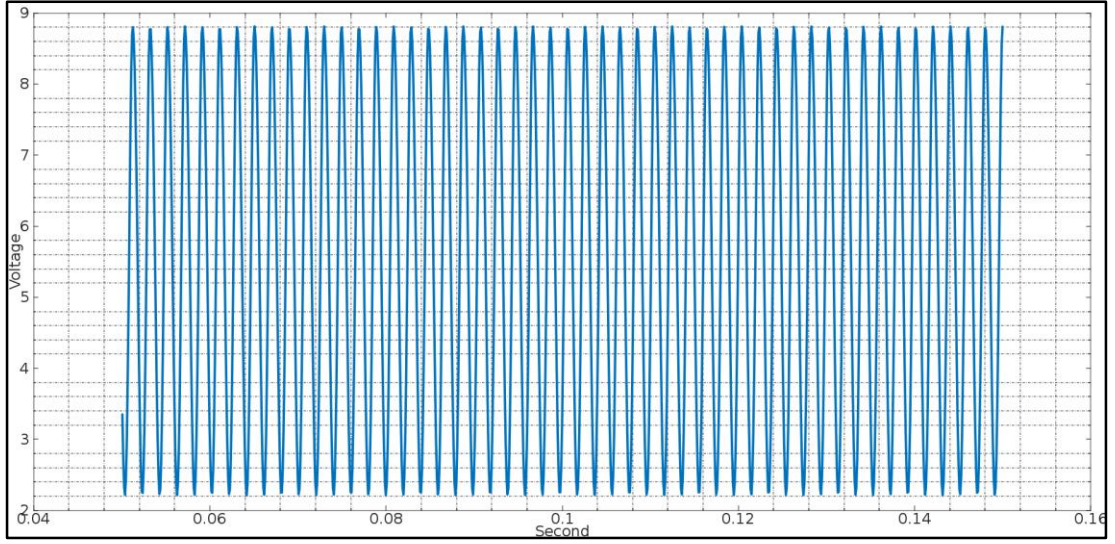


Şekil 2.9: RC devre şeması.

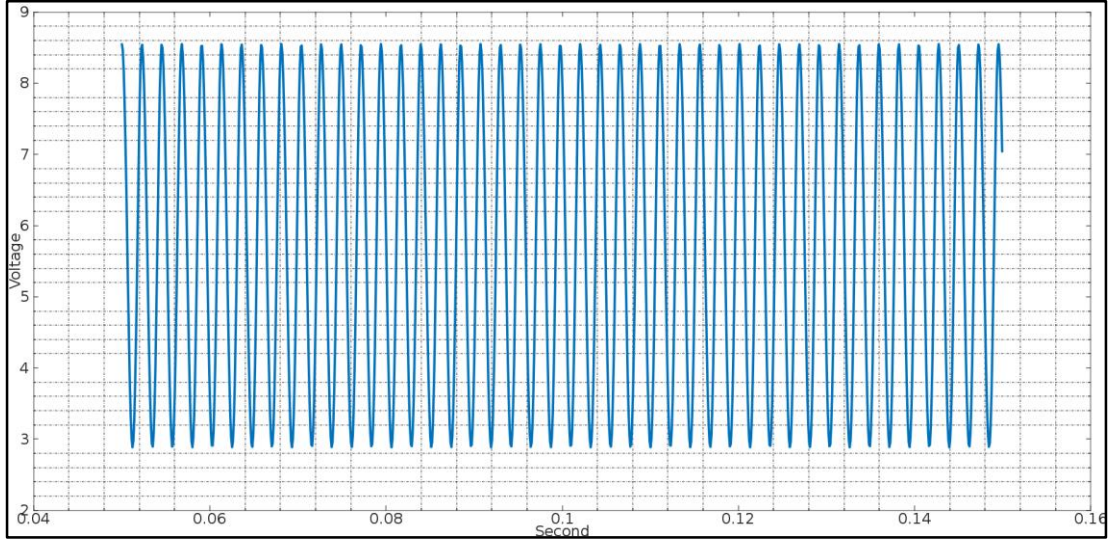
RC devresi üzerinde C1 kapasitesi parametrik olarak değiştirilmiş ve periyodu takip edilmiştir. Şekil 2.10:'da gösterildiği gibi her bir değer için ayrı grafik ve periyod elde edilmiştir. Kapasitansın değeri arttıkça periyodun arttığı gözlenmiştir.

Periyod değerleri sırasıyla aşağıdaki gibidir:

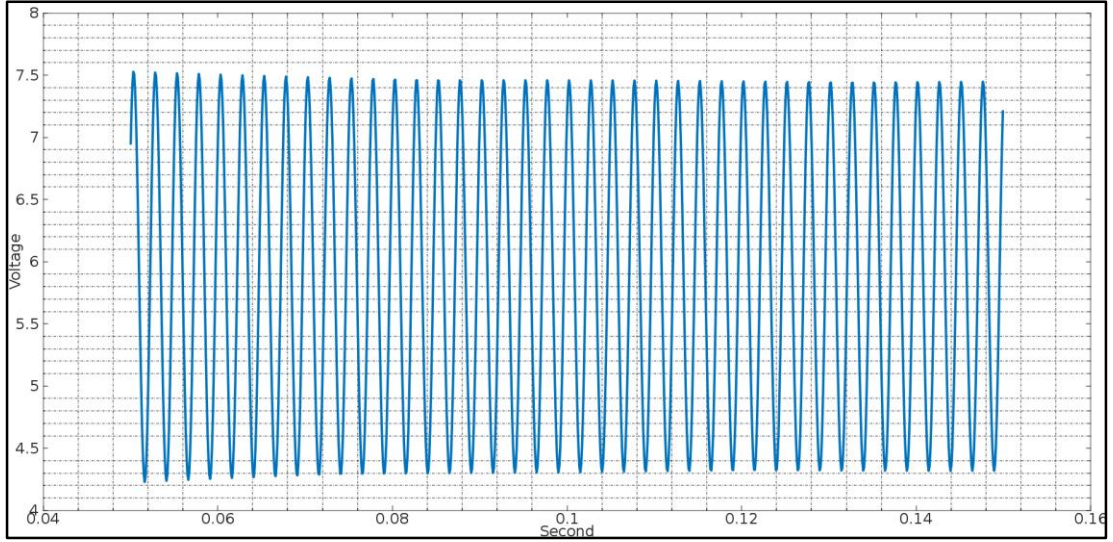
- C_1 20nF iken 1.94 ms
- C_1 30nF iken 2.22 ms
- C_1 40nF iken 1.38 ms
- C_1 50nF iken sonuç yok



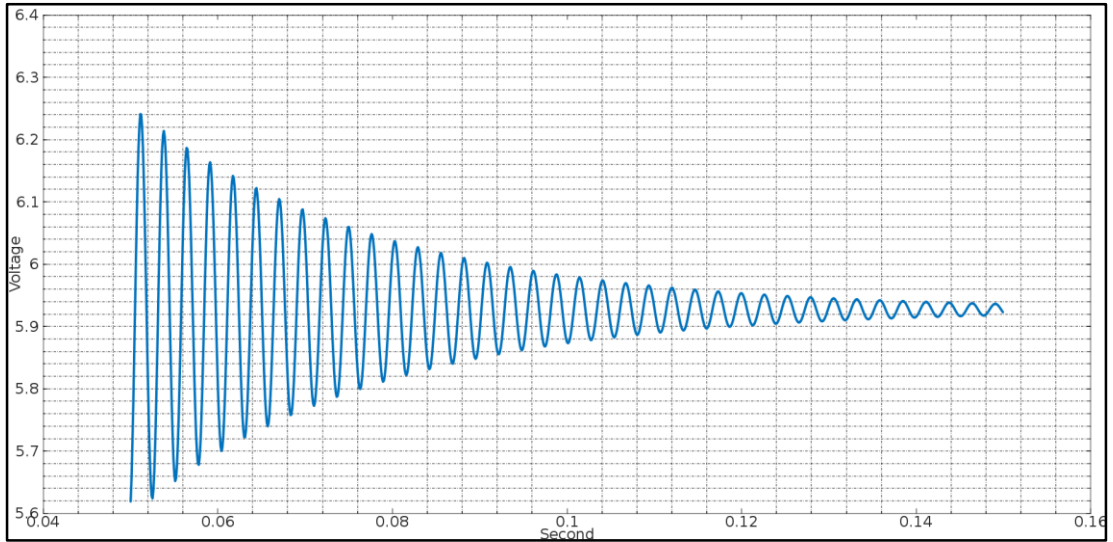
Şekil 2.10: $C_1 = 20\text{nF}$ için RC devresi benzetim sonucu.



Şekil 2.11: $C_1 = 30\text{nF}$ için RC devresi benzetim sonucu.



Şekil 2.12: $C_1 = 40\text{nF}$ için RC devresi benzetim sonucu.



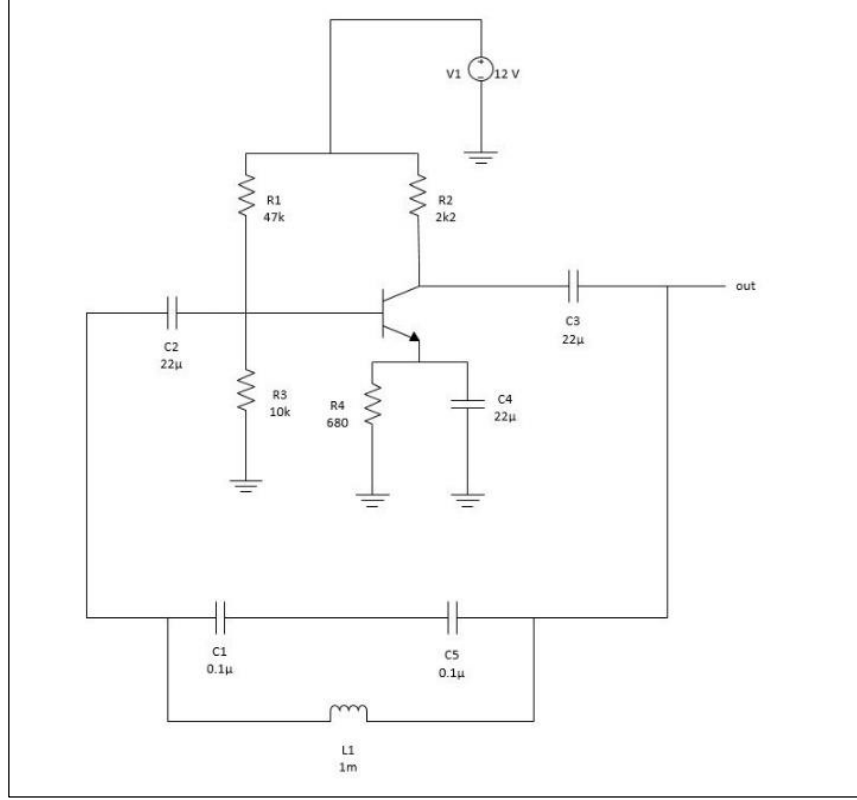
Şekil 2.13: $C_1 = 50\text{nF}$ için RC devresi benzetim sonucu.

2.4.2. Colpitts Osilatör

Colpitts osilatör geri besleme kaynağı olarak iki adet kapasitör ve bu kapasitörlerin tam ortasına bağlı bir bobini kullanır. C_1 ve C_2 değerlerini çok fazla geri bildirim yapacak şekilde ayarlarsak çıkıştaki sinüs sinyalinin bozulmasını neden oluruz, eğer çok düşük ayarlarsak ise devre salınım yapmayabilir[11]. Bu nedenle bu değerleri seçerken 2.2 denklemini de göz önünde bulundurmalıyız.

$$f = \frac{1}{2\pi(\sqrt{LC_T})} \quad (2.2)$$

$$\frac{1}{C_T} = \frac{1}{C_1} + \frac{1}{C_2} \quad (2.3)$$

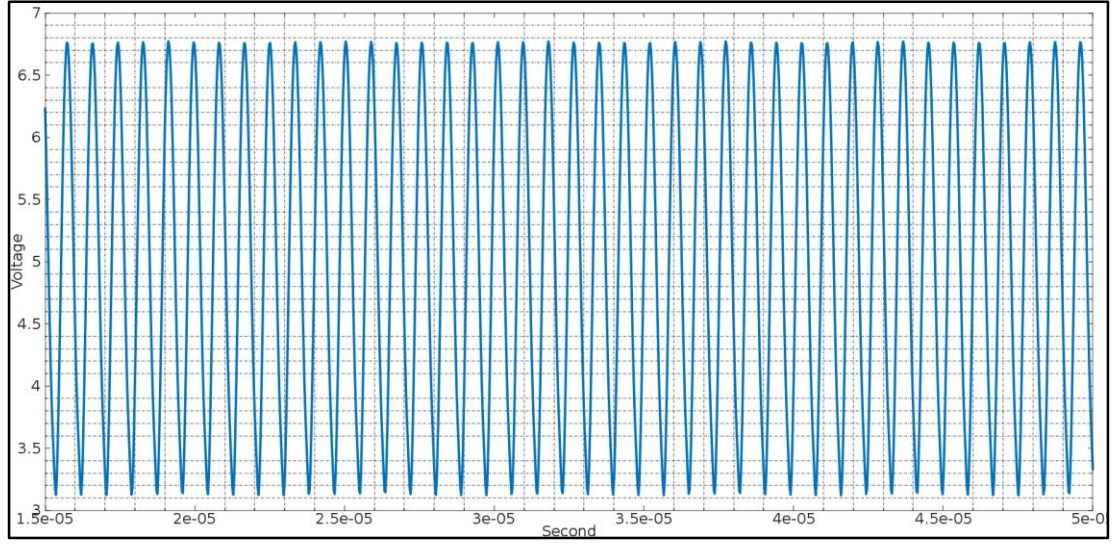


Şekil 2.14: Colpitts devre şeması.

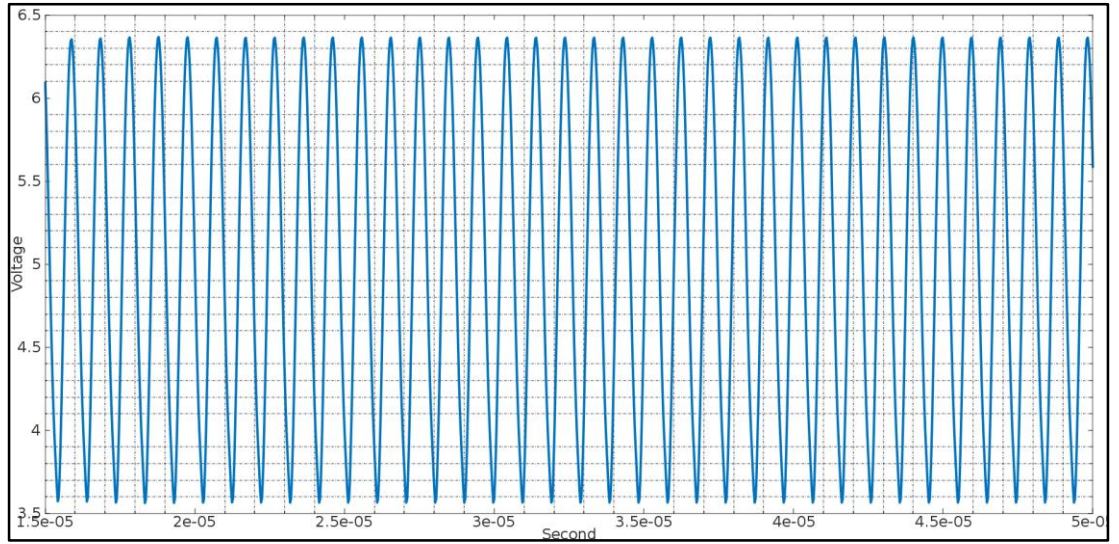
Değiştirilen kapasite değeri artan yönde bir hareket yaptıkça toplam kapasitansı düşürüyor, bu da karşımıza periyodun düşmesi sonucu olarak çıkıyor.

Periyod değerleri sırasıyla aşağıdaki gibidir:

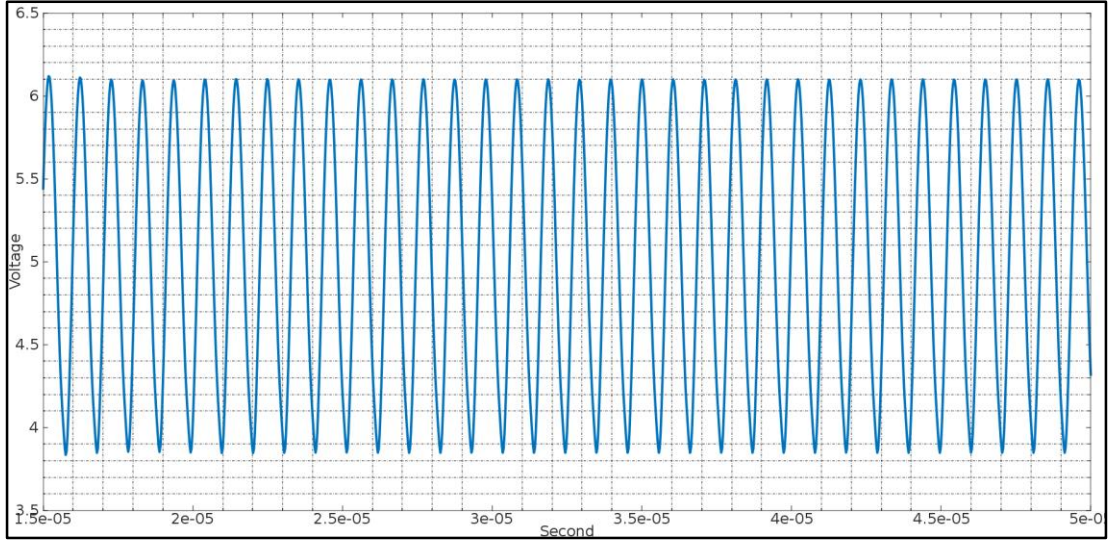
- C_1 3nF iken 0.83 μ s
- C_1 5nF iken 0.92 μ s
- C_1 7nF iken 1.00 μ s
- C_1 9nF iken 1.04 μ s



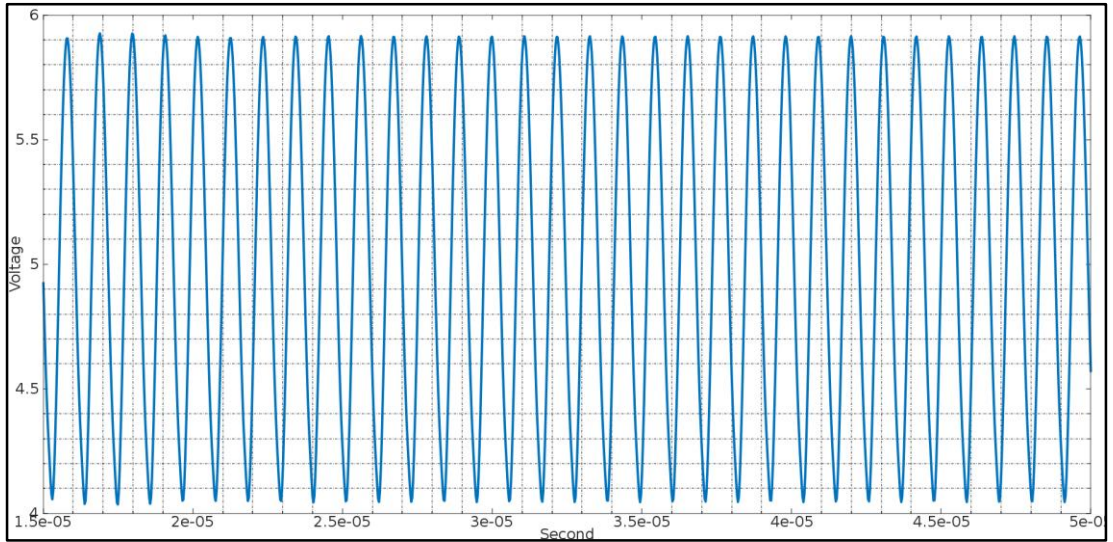
Şekil 2.15: $C_1 = 3\text{nF}$ için Colpitts devresi benzetim sonucu.



Şekil 2.16: $C_1 = 5\text{nF}$ için Colpitts devresi benzetim sonucu.



Şekil 2.17: $C_1 = 7\text{nF}$ için Colpitts devresi benzetim sonucu.



Şekil 2.18: $C_1 = 9\text{nF}$ için Colpitts devresi benzetim sonucu.

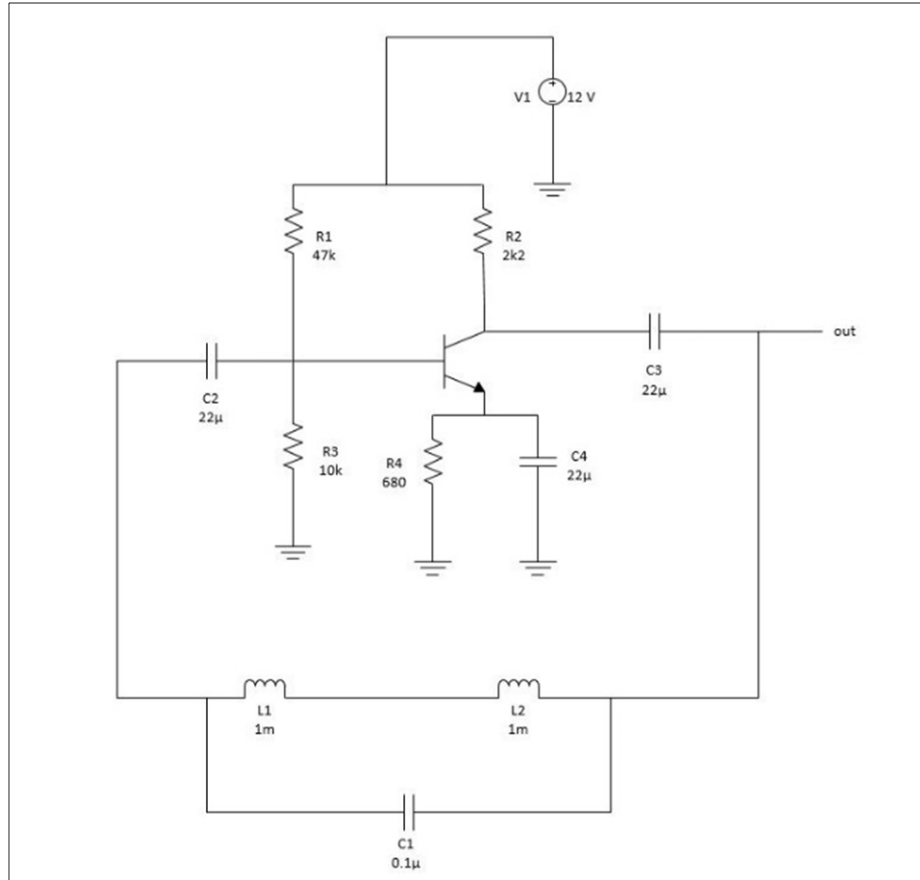
2.4.3. Hartley Osilatör

Hartley osilatörü geri besleme kaynağı olarak iki adet bobin ve bu bobinlerin tam ortasına bağlı bir kapasite kullanır [12]. L_1 ve L_2 arasındaki toplam elektromanyetik eşleşme çok küçükse, geri besleme yetersiz olur ve salınımlar sonunda sıfıra düşer. Aynı şekilde, eğer geri besleme çok güçlüyse, salınımlar sinyal bozulması üreten devre koşulları tarafından sınırlandırılana kadar genlikle beraber artmaya devam edecektir. Bu nedenle osilatörü ayarlamak çok zorlaşır. Hartley tipinde tasarlanan bir osilatörde ayar yapma işi denklem 2.4'deki gibi kolaylaşır. Burada L_T devredeki toplam endüktansı ifade etmek için kullanıldı. M değeri de iki

bobinin karşılıklı endüktansını gösterir. C değeri ayarlama kapasitörü olarak görev yapmaktadır.

$$f = \frac{1}{2\pi(\sqrt{CL_T})} \quad (2.4)$$

$$L_T = L_1 + L_2 + 2 \quad (2.5)$$

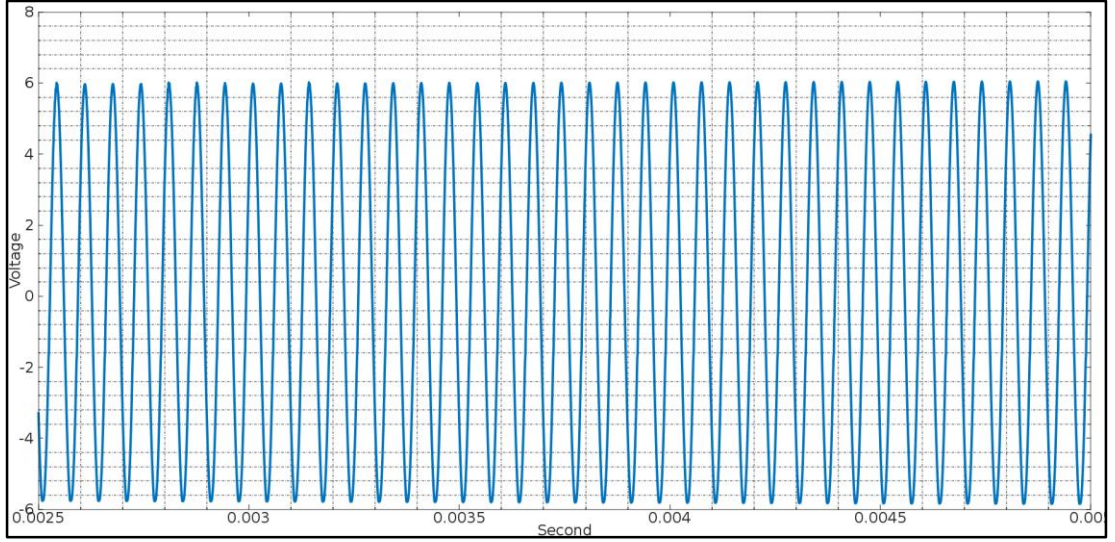


Şekil 2.19: Hartley devre şeması.

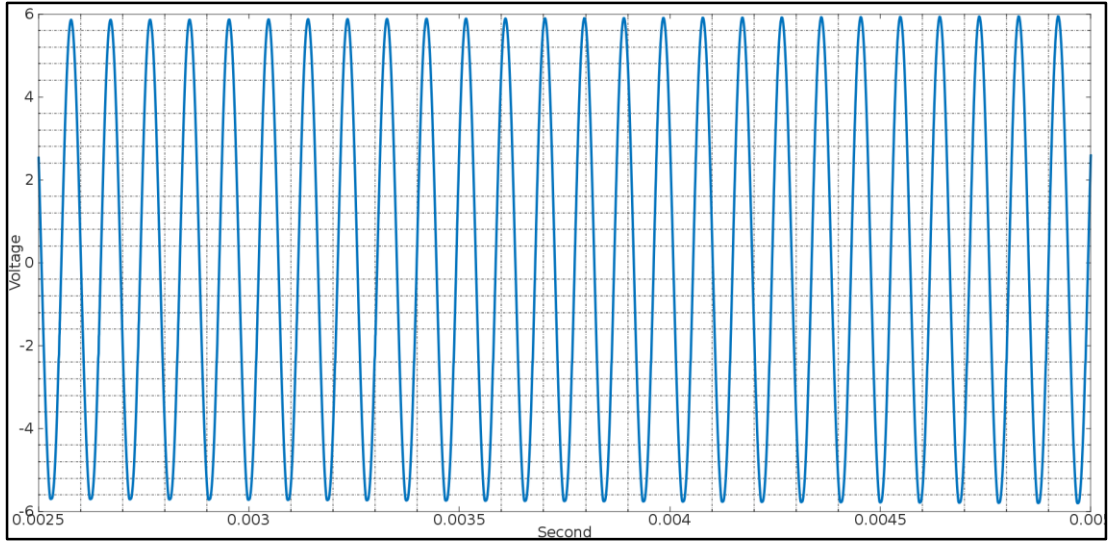
Değiştirilen kapasite değeri artan yönde bir hareket yaptıkça, periyodun arttığını gözlemlendi.

Periyod deęerleri sırasıyla ařaęıdaki gibidir:

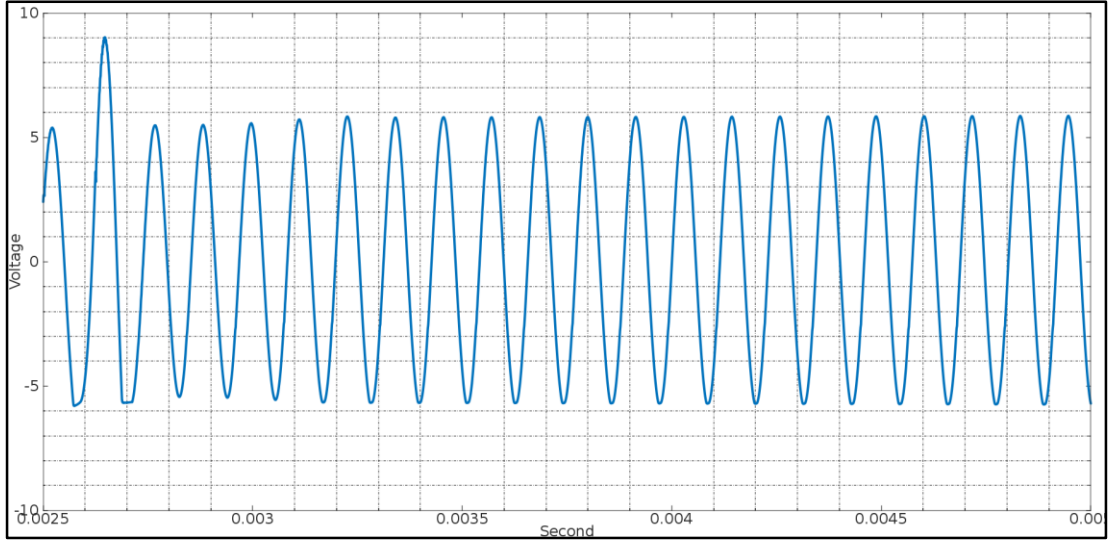
- C_1 0.1uF iken 33.91 μ s
- C_1 0.2uF iken 45.34 μ s
- C_1 0.3uF iken 56.36 μ s
- C_1 0.4uF iken 64.28 μ s



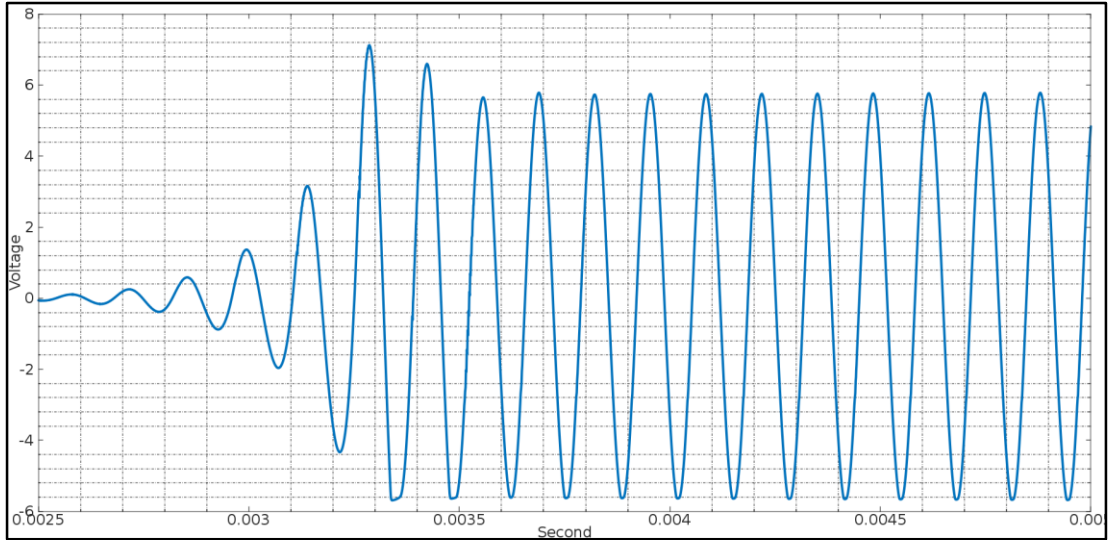
řekil 2.20: $C_1 = 0.1\mu$ F ięin Hartley devresi benzetim sonucu.



řekil 2.21: $C_1 = 0.2\mu$ F ięin Hartley devresi benzetim sonucu.



Şekil 2.22: $C_1 = 0.3\mu\text{F}$ için Hartley devresi benzetim sonucu.



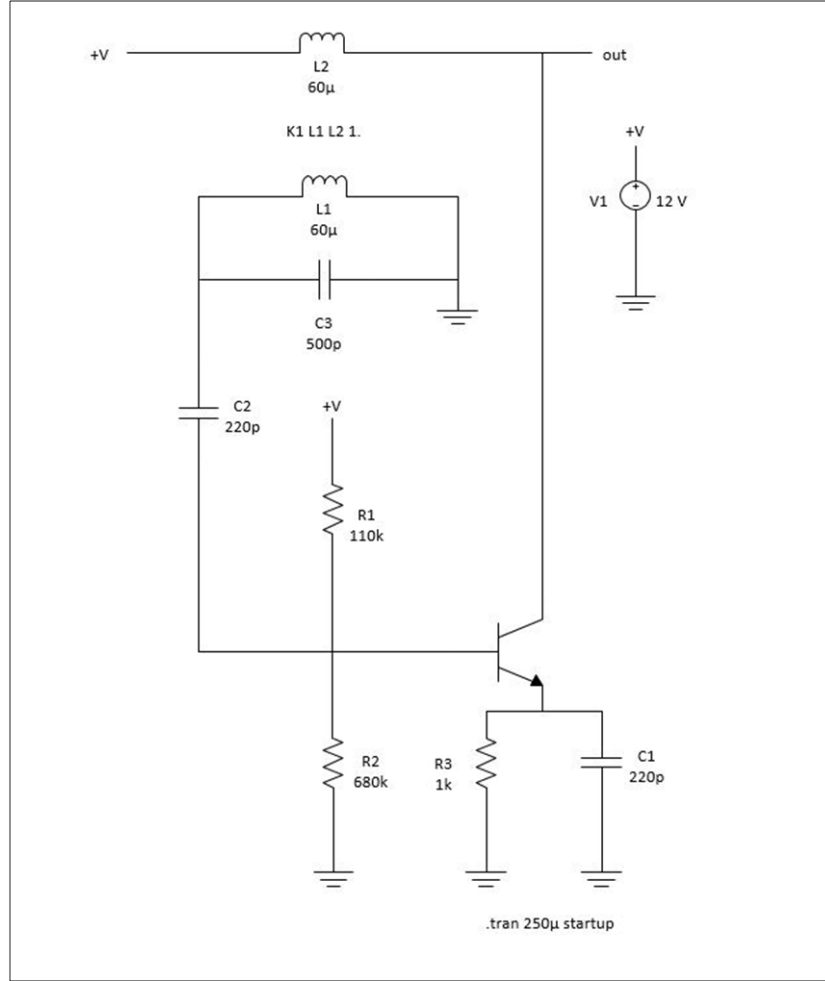
Şekil 2.23: $C_1 = 0.4\mu\text{F}$ için Hartley devresi benzetim sonucu.

2.4.4. Armstrong Osilatör

LC osilatör türlerinden biri olan Armstrong osilatör Meissner adı ile de bilinir. FET ve MOSFET kullanılarak yapılabilir. Osilatör çalışma frekansı geri beslemede bulunan kapasite ve bobin ile sağlanır. Giriş ve çıkış fazları aynı değerde olmaktadır. Armstrong osilatörün çalışma frekansı denklem 2.6'da olduğu gibi hesaplanır. Burada L_T ve C_T devredeki toplam endüktansı ifade etmek için kullanıldı. Armstrong osilatörü, verilen radyo frekansı aralığında karalı bir genliğe ve oldukça kararlı bir frekansı olan sinüzoidal bir sinyal çıkışı elde etmek için kullanılır.

Armstrong osilatörün genel kullanım yerleri şu şekildedir; radyo alıcılarında yerel bir osilatör olarak kullanılır, sinyal jeneratörlerinde bir kaynak olarak ve orta ve yüksek frekans aralığında bir radyo frekanslı bir osilatör olarak kullanılabilir[13].

$$f = \frac{1}{2\pi(\sqrt{L_T C_T})} \quad (2.6)$$

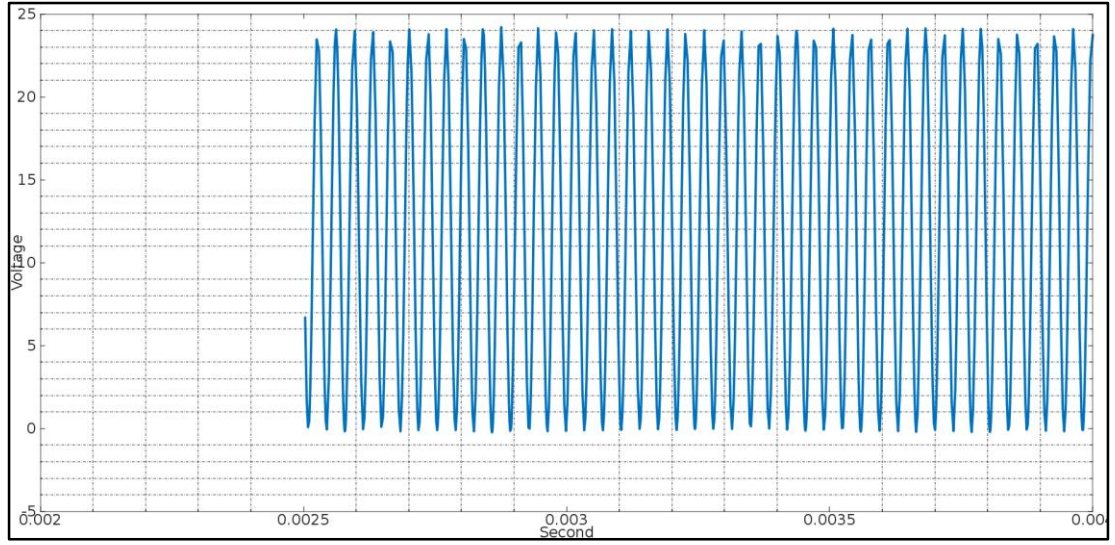


Şekil 2.24: Armstrong devre şeması.

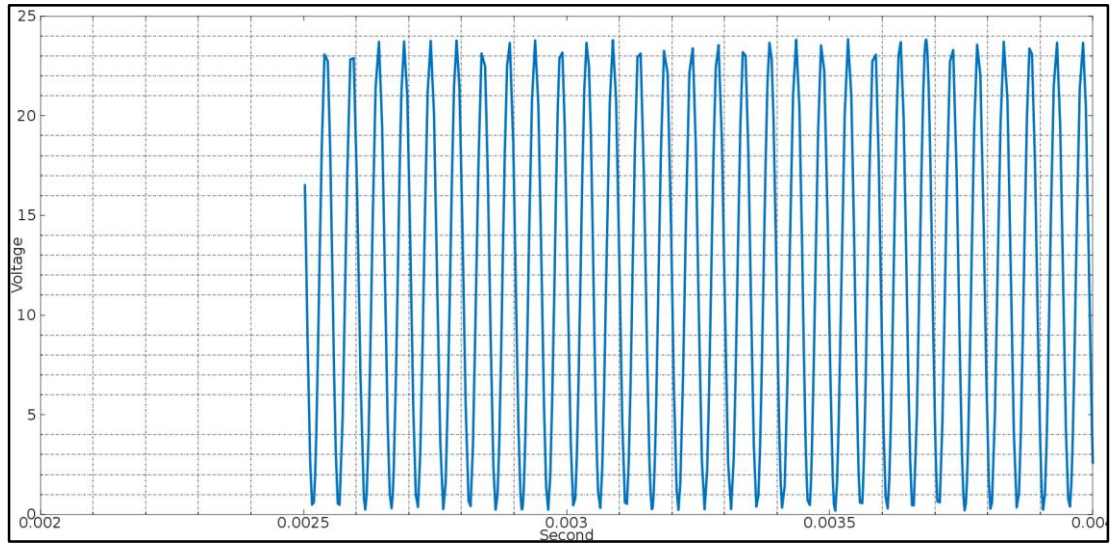
Değiştirilen kapasite değeri artan yönde bir hareket yaptıkça, periyodun arttığını gözlemlendi.

Periyod deęerleri sırasıyla ařaęıdaki gibidir:

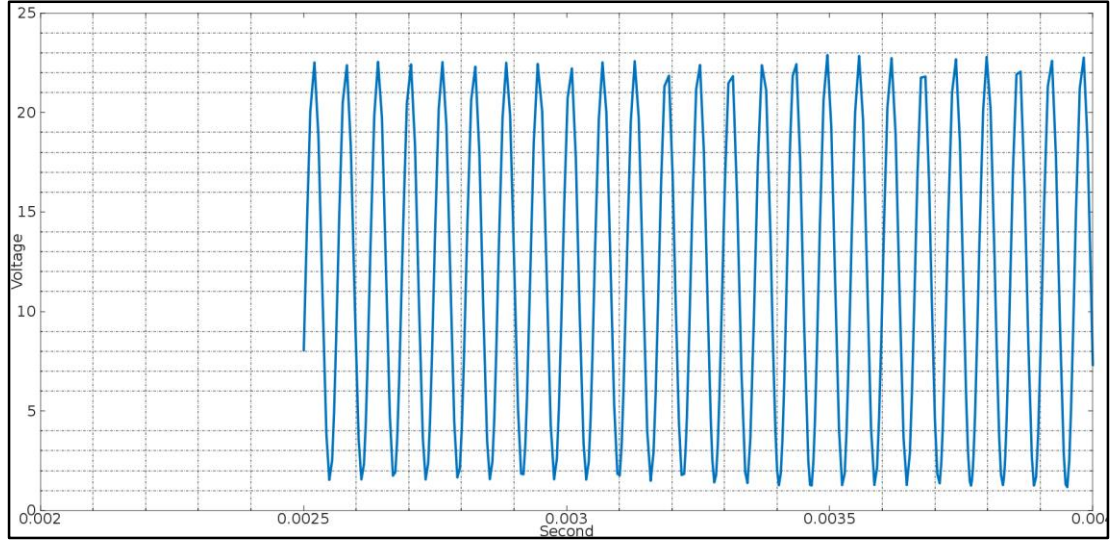
- C_3 30nF iken 36.07 μ s
- C_3 60nF iken 54.02 μ s
- C_3 90nF iken 65.90 μ s
- C_3 120nF iken 70.29 μ s



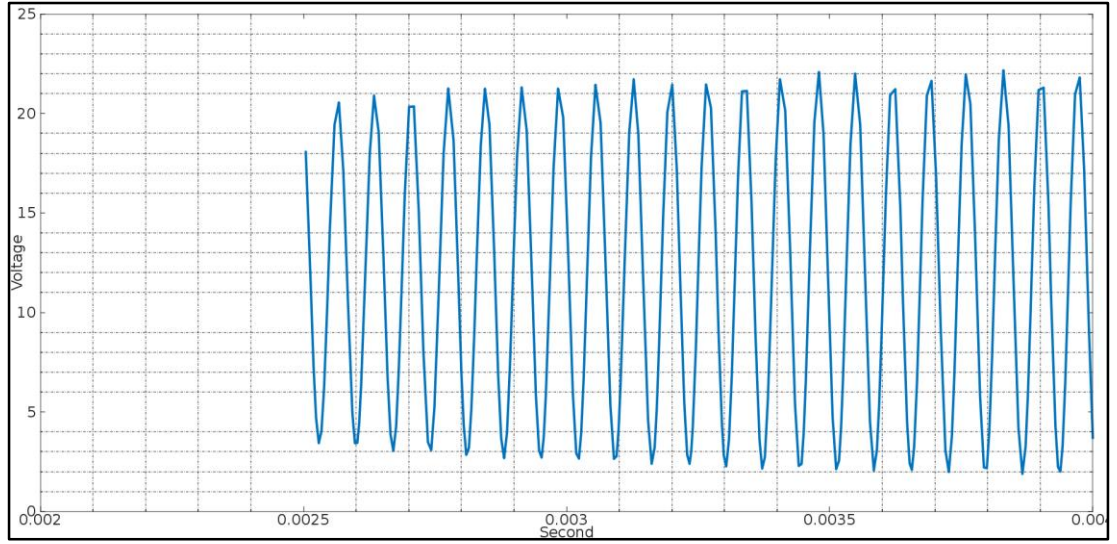
řekil 2.25: $C_1 = 30$ nF iin Armstrong devresi benzetim sonucu.



řekil 2.26: $C_1 = 60$ nF iin Armstrong devresi benzetim sonucu.



Şekil 2.27: $C_1 = 90\text{nF}$ için Armstrong devresi benzetim sonucu.



Şekil 2.28: $C_1 = 120\text{nF}$ için Armstrong devresi benzetim sonucu.

3. SONUÇLAR ve ÖNERİLER

Ngspice ve Ltspice programlarında elde edilen osilatör sonuçlarından emin olundu. Bir sonraki aşamada Octave ile bu devre benzetimleri çalıştırılıp sonuçların ileri veri işleme programı ile işlenmesi kullanıcı tarafından gerçekleştirildi. Mex/C++ mekanizma işleyişi ve kullanılan “kapsamlı makefile” projesinin bu projede nasıl etkili bir şekilde kullanılacağı üzerine çalışmalar yapıldı.

Benzetim programlarından en verimli şekilde fayda sağlayarak teorik bilgi birikiminin gerçek dünyaya yakın olan benzetim programlarında kullanıldı. Bu programlardan Ngspice’in kendi bünyesinde paylaşılabılır C kütüphane sahibi olması ve kullanıcıya istediğini yapacak alan sağlamasının faydalı sonuçları görüldü. Elde edilen benzetim sonuçları sayesinde ayarlanabilir osilatör tasarımları rahatlıkla oluşturulabilir hale geldi. Parametrik osilatör kodlaması, periyodun değişimi ve osilatörün çalışma aralığı verimli şekilde gösterdi. Belli değerlerde osilatörün başlamaması durumunda onu başlatmak için tahmin edilen çıkış sinyaline yakın çok daha düşük genlikli ve kısa bir süreliğine verildiğinde osilasyon işleminin gerçekleştiği görüldü. Osilatörün bu duruma girmesine sebep parametrik olarak ayarlamak istediğimiz fakat sınırların biraz dışında gezdiğimiz özel bir olaydır. Aşırı derecede sınır osilatörün çalışma aralığından çıkınca osilatör prob devresi de devreyi osilasyona girdiremez halde olacaktır.

Octave kullanarak Ngspice’den netlistleri çalıştırmak ve üzerinde çok daha üst düzey bir dil olan Octave/Matlab dilini kullanmak çok daha geniş bir alanda benzetim sonrası işlemlere olanak sağlamaktadır.

Osilatör tasarımlarından RC Osilatörde istenen frekans aralığını elde etmek için kullanılan kapasitör değerlerinin yaklaşık on katı değer verilince osilasyon oluşmuyor ve periyod bilgisi elde edilemiyor olduğunu gördük. Diğer Osilatörlerde kapasitenin değeri arttıkça periyodun büyüdüğü görüldü, istenen frekans değeri için kullanılan kapasite değerinden yirmi katı değerinde bir kapasitans kullanıldığında osilasyon elde edilemez ve periyod ölçülemez oluyor.

KAYNAKLAR

- [1] Razavi B., (2017), “Design of Analog CMOS Integrated Circuits”, 2nd Edition, McGraw Hill.
- [2] Mayaram K.,(2000), “Output voltage analysis for the MOS colpitts oscillator”, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 47 (2), 260-263.
- [3] Meyer R. G., Soo D. C.-F., (1980), “MOS crystal oscillator design”, IEEE J. Solid-State Circuits, 15 (2), 222-228.
- [4] Attia O. J., (2018), “PSPICE and MATLAB for Electronics: An Integrated Approach”, 2nd Edition, CRC Press.
- [5] Sorkun A., Çise A., Şen Özen E. B., Duman M. E., Şuvak Ö., (2018), “Makefile Projesinin Test ve Mex Paradigmaları – Tasarım Ayrıntıları ve İşleyiş”, GTU Lisansüstü Araştırmalar Sempozyumu & Tanıtım Günleri, 42-43, Kocaeli/Gebze/TÜRKİYE, 14-16 Mayıs 2018.
- [6] Web 1, (2019), <http://ngspice.sourceforge.net/docs/ngspice-manual.pdf>, (Erişim Tarihi: 20/05/2019).
- [7] Duman M. E., Şuvak Ö., (2017), “C Tabanlı Elektronik Devre Benzetim Programının MATLAB Aracılığıyla Kontrol Edilmesi ve Eşit Olmayan Zaman Aralıklarında Örneklenmiş Benzetim Verilerinin Frekans Analizi için Nümerik bir FFT Yöntemi”, GTU Lisansüstü Araştırmalar Sempozyumu & Tanıtım Günleri, 53, Kocaeli/Gebze/TÜRKİYE, 17-18 Mayıs 2017.
- [8] Web 2, (2019), <https://github.com/osuvak/dev-in-place>, (Erişim Tarihi: 20/05/2019).
- [9] Hao Y. S., Han L. J., Wang G. L., (2010), “A new low-noise active RC oscillator”, Computer Engineering, 36 (12), 253-255.
- [10] Lapinoja M., Rahkonen T., “Interface program design of circuit simulation system based on SPICE”, IEEE Proceedings of 40th Midwest Symposium on Circuits and Systems. Dedicated to the Memory of Professor Mac Van Valkenburg, Sacramento/CA/USA, 6 Aug. 1997.
- [11] Razavi B.,(1996), “A Study of Phase Noise in CMOS Oscillators”, IEEE journal of solid state circuits, 31(3), 341-343.
- [12] Rashid M. H., (2010), “Microelectronic Circuits: Analysis & Design”, 2nd Edition, CL Engineering.
- [13] Tchamov N.T., Niemi T., Mikkola N., (2001), “High-performance differential VCO based on Armstrong oscillator topology”, IEEE, 36 (1), 139-141.

ÖZGEÇMİŞ

Eyüp ÇUĞALIR, 1993 yılında Gebze’de doğdu. 2011 yılında Antalya Gazi Anadolu Lisesi’ni tamamladı ve ardından başladığı Gebze Yüksek Teknoloji Enstitüsü (GYTE), Mühendislik Fakültesi, Elektronik Mühendisliği Bölümü’nü 2016 yılında üçüncü olarak tamamladı. Aynı yıl yüksek lisans eğitimine GTÜ Fen Bilimleri Enstitüsü Elektronik Mühendisliği bölümünde başladı. 2017 yılından bu yana TÜBİTAK BİLGEM’de Mayın ve El Yapımı Patlayıcı Karşı Tedbirleri Test ve Araştırma Merkezinde elektronik devre tasarımı, testi ve tasarım geliştirme konularında araştırmacı olarak çalışmaktadır.

EKLER

Ek A: Tez Çalışması Kapsamında Yapılan Yayınlar

Çuğalır E., Duman M. E., Şuvak Ö., (2018), “Matlab/Mex/C++ Üzerinde Ngspice Bağlantı Aracı Tasarımı”, GTU Lisansüstü Araştırmalar Sempozyumu & Tanıtım Günleri, 54, Kocaeli/Gebze/TÜRKİYE, 14-16 Mayıs 2018.

Ek B: Octave Kodu ve Simülasyon Netlisti

Tablo B1.1: RC osilatörü octave kodu.

close all
clear all
clc
format long e
alib;
pkg load signal;
C1 = 'C1';
S1 = linspace(20*1e-9,50*1e-9,4);
for ee=1:numel(S1)
file_id = fopen('/home/gtuee/Documents/RC/part1.cir','w');
fprintf(file_id," %s %i %i %e", C1, 2, 6, S1(ee));
fclose(file_id);
str = '/home/gtuee/Documents/RC/RC.cir'
return_vectors = ngspice_so_iface_20171111_2226(str);
t = return_vectors(end).values;
indisis = find(t > 0.05);
indisis_graph = find(t > 0.05 & t < 0.15);
esp = (t(end)-t(1))/length(t);
y = (return_vectors(2).values);
yabs = abs(return_vectors(2).values);
ac = xcorr(t(indisis),y(indisis));
[pks,freqs] = findpeaks(ac);

Tablo B1.1: Devam.

z(ee) = mean(diff(freqs)*esp);
figure(ee)
plot(t(indisis_graph),y(indisis_graph),'LineWidth',4);
grid minor;
xlabel ("Second","fontsize", 18);
ylabel ("Voltage","fontsize", 18);
#saveas(1, "figure1.jpeg");
set(gca, "linewidth", 1, "fontsize", 18);
end
disp("")
for kk=1:numel(return_vectors)
disp (sprintf('%3d %s', kk, return_vectors(kk).str))
end
disp("")

Tablo B1.2: RC devresi netlisti.

R1 1 6 100k
R2 6 0 10k
R3 1 2 10k
R4 1 3 10k
R5 1 4 4k7
R6 7 0 390
C1 2 6 2e-08
C2 3 2 10n
C3 4 3 10n
C4 5 4 10μ
C5 0 7 100μ
V1 1 0 9
Q1 4 6 7 0 qnl
.model qnl npn(level=1 bf=120 rb=100 ccs=2pf tf=0.3ns tr=6ns cje=3pf cjc=2pf va=50)
.tran 0.1m 200m uic
.control
*tran 0.1m 200m uic
*run
*plot v(5)
.endc
.end

Tablo B1.3: Colpitts osilatörü octave kodu.

close all
clear all
clc
format long e
alib;
pkg load signal;
C1 = 'C1';
S1 = linspace(3*1e-9,9*1e-9,4);
for ee=1:numel(S1)
file_id = fopen('/home/gtuee/Documents/Colpits/part1.cir','w');
fprintf(file_id," %s %i %i %e", C1, 2, 3, S1(ee));
fclose(file_id);
str = '/home/gtuee/Documents/Colpits/Colpits.cir'
return_vectors = ngspice_so_iface_20171111_2226(str);
t = return_vectors(end).values;
indisis_graph= find(t > 1.5e-05 & t < 5e-05);
indisis = find(t > 0.5e-05);
esp = (t(end)-t(1))/length(t);
y = (return_vectors(8).values);
yabs = abs(return_vectors(8).values);
ac = xcorr(t(indisis),yabs(indisis));
[pks,freqs] = findpeaks(ac);
z(ee) = mean(diff(freqs)*esp);
figure(ee)
plot(t(indisis_graph),y(indisis_graph),'LineWidth',4);
grid minor;
xlabel ("Second","fontsize", 18);
ylabel ("Voltage","fontsize", 18);
set(gca, "linewidth", 1, "fontsize", 18);
end
disp("")
for kk=1:numel(return_vectors)
disp (sprintf('%3d %s', kk, return_vectors(kk).str))
end
disp("")

Tablo B1.4: Colpitts osilatör devresi netlisti.

r1	1 0	1
q1	2 1 3	qnl
vcc	4 0	5
rl	4 2	750
C1	2 3	5.000000e-10
c2	4 3	4500p
l1	4 2	10uH
re	3 6	4.65k
vee	6 0	dc -10 pwl 0 0 1e-9 -10
.model qnl npn(level=1 bf=80 rb=100 ccs=2pf tf=0.3ns tr=6ns cje=3pf cjc=2pf va=50)		
.tran 30n 50u		
.control		
*tran 30n 500u		
*run		
*plot v(3)		
.endc		
.end		

Tablo B1.5: Hartley osilatörü octave kodu.

close all
clear all
clc
format long e
alib;
pkg load signal;
C1 = 'C1';
S1 = linspace(0.1*1e-6,0.4*1e-6,4);
for ee=1:numel(S1)
file_id = fopen('/home/gtuee/Documents/Hartley/part1.cir','w');
fprintf(file_id," %s %i %i %e", C1, 4, 6, S1(ee));
fclose(file_id);
str = '/home/gtuee/Documents/Hartley/Hartley.cir'
return_vectors = ngspice_so_iface_20171111_2226(str);
t = return_vectors(end).values;
indisis_graph= find(t > 0.0025);
indisis = find(t > 0.0025);
esp = (t(end)-t(2))/length(t);

Tablo B1.5: Devam.

y = (return_vectors(8).values);
yabs = abs(return_vectors(8).values);
ac = xcorr(t(indisis),yabs(indisis));
[pks,freqs] = findpeaks(ac);
z(ee) = mean(diff(freqs)*esp);
figure(ee)
plot(t(indisis_graph),y(indisis_graph),'LineWidth',4);
grid minor;
xlabel ("Second","fontsize", 18);
ylabel ("Voltage","fontsize", 18);
set(gca, "linewidth", 1, "fontsize", 18);
end
disp("")
for kk=1:numel(return_vectors)
disp (sprintf("%3d %s", kk, return_vectors(kk).str))
end
disp("")

Tablo B1.6: Hartley osilatör devre netlisti.

Q1 2 3 5 BC547B
V1 1 0 12
R1 1 2 2.2k
R2 1 3 47k
R3 3 0 10k
R4 5 0 680
C2 3 4 10u
C3 2 6 22u
C4 5 0 22u
L1 4 0 0.1m
L2 0 6 1m
C1 4 6 0.1u

Tablo B1.6: Devam.

.model BC547B NPN(IS=2.39E-14 NF=1.008 ISE=3.545E-15 NE=1.541 BF=294.3 IKF=0.1357 VAF=63.2 NR=1.004 ISC=6.272E-14 NC=1.243 BR=7.946 IKR=0.1144 VAR=25.9 RB=1 IRB=1.00E-06 RBM=1 RE=0.4683 RC=0.85 XTB=0 EG=1.11 XTI=3 CJE=1.358E-11 VJE=0.65 MJE=0.3279 TF=4.391E-10 XTF=120 VTF=2.643 ITF=0.7495 PTF=0 CJC=3.728E-12 VJC=0.3997 MJC=0.2955 XCJC=0.6193 TR=1.00E-32 CJS=0 VJS=0.75 MJS=0.333 FC=0.9579 Vceo=45 Icrating=100m)
.tran 1u 5m
.control
*tran 1u 200m
*run
*plot v(6)
.endc
.end

Tablo B1.7: Armstrong osilatörü octave kodu.

close all
clear all
clc
format long e
alib;
pkg load signal;
C1 = 'C1';
S1 = linspace(30*1e-9,120*1e-9,4);
for ee=1:numel(S1)
file_id = fopen('/home/gtuee/Documents/Armstrong/part1.cir','w');
fprintf(file_id," %s %i %i %e", C1, 0, 1, S1(ee));
fclose(file_id);
str = '/home/gtuee/Documents/Armstrong/Arms.cir'
return_vectors = ngspice_so_iface_20171111_2226(str);
t = return_vectors(end).values;
indisis = find(t > 0.0025);
esp = (t(end)-t(1))/length(t);
y = (return_vectors(10).values);
yabs = abs(return_vectors(10).values);
ac = xcorr(t(indisis),yabs(indisis));
[pks,freqs] = findpeaks(ac);

Tablo B1.7: Devam.

z(ee) = mean(diff(freqs)*esp);
figure(ee)
plot(t(indisis),y(indisis),'LineWidth',4);
grid minor;
xlabel ("Second","fontsize", 18);
ylabel ("Voltage","fontsize", 18);
set(gca, "linewidth", 1, "fontsize", 18);
end
disp("")
for kk=1:numel(return_vectors)
disp (sprintf('%3d %s', kk, return_vectors(kk).str))
end
disp("")

Tablo B1.8: Armstrong osilatör devre netlisti.

C1 0 1 10n
L1 1 0 1m
C2 1 4 220p
L3 +v 8 1m
Q1 8 4 6 BC547C
R1 +v 4 110k
R2 4 0 680k
V1 +v 0 12
R3 6 0 1k
C4 6 0 220p
V2 2 0 sine(0 2 100)
R4 3 2 100
S1 3 4 5 0 SW
V3 5 0 pulse(-1 1 0 1n 1n 5n)
.model BC547C NPN(IS=4.679E-14 NF=1.01 ISE=2.642E-15 NE=1.581 BF=458.7 IKF=0.1371 VAF=52.64 NR=1.019 ISC=2.337E-14 NC=1.164 BR=11.57 IKR=0.1144 VAR=364.5 RB=1 IRB=1.00E-06 RBM=1 RE=0.2598 RC=1 XTB=0 EG=1.11 XTI=3 CJE=1.229E-11 VJE=0.5591 MJE=0.3385 TF=4.689E-10 XTF=160 VTF=2.828 ITF=0.8842 PTF=0 CJC=4.42E-12 VJC=0.1994 MJC=0.2782 XCJC=0.6193 TR=1.00E-32 CJS=0 VJS=0.75 MJS=0.333 FC=0.7936)
K1 L1 L3 1.
.tran 0.01m 4m
.model SW SW()
.control

Tablo B1.8: Devam.

*run
*plot v(5)
*plot v(2)
*plot v(8)
.endc
.end