RSSI Tabanlı Konum Tespiti

17.02.2018

Mehmet Çağrı Yonca
Karadeniz Teknik Üniversitesi
Bilgisayar Mühendisliği

Giriş

Projemizin amacı konumu bilinen en az üç sinyal verici cihaz yardımı ile, bu üç cihazın radyo frekanslarının kesişim alanı içerisindeki konumu bilinmeyen dördüncü bir cihazın konumunu tespit edebilmek.

Proje Kapsamında Kullanılanlar

1. IOT cihazlar arasında bağlantı gerçekleştirebilmek için Contiki OS.
2. Kod üzerinde düzenlemeyi rahat bir şekilde uygulayabilmek için Sublime Text Editor.
3. Yazılan kodların gerçekte çalışıp çalışmayacağını test edebilmek için Cooja Simulator.

Proje Aşamaları

Projeye başlamadan önce ilk olarak kafamda proje taslağını oluşturup olay yönetimini kolaylaştırabilmek için kağıda geçirdim. Elimizde üç nokta vardı ve bu üç nokta arasındaki alanda rastgele bir nokta alıp bu noktanın konumunu bulmam gerekliydi. İnternette yaptığım küçük bir araştırma sonunda orta noktaları bilinen çember denklemlerinin ortak çözümüne giderek istediğim noktaları bulabileceğimi fark ettim.

Konumu bilinen 3 nokta olsun;

$$N_1 = (x_1 \ , y_1), N_2 = (x_2 \ , y_2), N_3 = (x_3 \ , y_3)$$

Aranan nokta ise

$$N_x = (x \ , y \ )$$

olsun.

O halde bu üç noktanın orta nokta kabul edildiği 3 farklı çemberin denklemi;

$$(x - x_1)^2 + (y - y_1)^2 = r^2_1$$

$$(x - x_2)^2 + (y - y_2)^2 = r^2_2$$

$$(x - x_3)^2 + (y - y_3)^2 = r^2_3$$

olarak bulunur. Bu denklemlerde r_1 , r_2 , r_3 çemberlerin yarıçaplarıdır.

Denklemlerin ortak çözümünden;

$$(-2x_1 + 2x_2)x + (-2y_1 + 2y_2)y = r^2_1 - r^2_2 - x^2_1 + x^2_2 - y^2_1 + y^2_2$$

$$(-2x_2 + 2x_3)x + (-2y_2 + 2y_3)y = r^2_2 - r^2_3 - x^2_2 + x^2_3 - y^2_2 + y^2_3$$

denklemleri gelir.

3 noktanın arasında kalan alandaki N_x noktası bu üç çemberin kesişim noktası olarak ayarlanırsa denklemlerin ortak çözümünden gelen $(x \ , y \ )$ noktaları aranan nokta olacaktır.

Cihazlardan gönderilen sinyaller sayesinde ortadaki nokta nerede olursa olsun 3 cihazın da kapsama alanına girdiği sürece bu kesişim noktalarında olur, çünkü her cihaz radyo frekanslarını küresel olarak gönderir, ve RSSI değeri ile bu kürenin yarıçapı temsil edilir.

Noktaların çözümünü bulmak için gereken noktaları, noktaların çember denklemlerini ve yarıçapları biliyoruz. Bu noktadan sonra yapılması gereken tek şey bu uygulamanın kodlanması.

Kodlamaya Giriş

Kodlamaya başlamadan önce taslak üzerinde belirlediğim yöntemleri bilgisayara nasıl geçirebilirim bunu düşünmeye başladım. Öncelikle bir insanın bu problemi nasıl çözdüğünü anlayıp onu makineye anlatmanın daha kolay olabileceğini fark edip problemi kağıt üzerinde kendim çözdüm. Her aşamayı önce kendim çözüp sonra bilgisayara çözdürdüm.

Kodlama Aşamaları

Öncelikle elimdeki 3 cihazın birbiriyle bağlantısı olması lazım. Bunun için UDP bağlantısı kurup IPv6 protokolü ile cihazları birbirine tanıtarak başladım. Yapmış olduğum uygulamada tek cihazın konumu bulunmak istendiği için ortaya sunucuyu konumlandırdım, bu sayede dışarıdaki 3 istemci (dayanak noktaları) ile sunucunun kodları kendilerine özel olacaktı.

Kabaca sistem şu şekilde çalışacaktı;

İstemciler uygun yere konumlandırılır, koordinatları bellidir, koordinatlarını çevredeki cihazlara gönderir, çevredeki cihaz sunucu ise (yani aranan cihaz ise) koordinatları alıp işlemcisi üzerinde denklemleri çözüp kendi koordinatlarını belirler.

(Bu işlem adımları ortada istemci olduğu varsayılarak da yapılabilirdi, değişmesi gereken tek kısım tüm istemcilerin çevresindeki istemcilerle etkileşimini sağlamak olacaktı fakat tek düğüm istendiği için gerekli olduğunu düşünmedim)

Bu nedenle kodlama kısmı iki kısımdan oluşuyor;

-Denklem değişken katsayılarının belirlenmesi

-Katsayıları belirlenen denklemin Gauss Yok Etme Metodu yardımı ile çözümü

Denklem Değişken Katsayılarının Belirlenmesi

Öncelikle dayanak noktalarının (konumları bilinen istemciler) koordinatlarını simulasyon üzerinde görebilmek için konum ayarlama(setleme) fonksiyonu yazmam gerekiyordu. Bu aşamayı istemciler üzerine setx ve sety adında 2 ayrı giriş koşulu ile tamamladım.

Kabaca setx ve sety fonksiyonları kendinden sonra gelen karakterleri char dizisi olarak alıp integere çevirip sayı değerinden ‘0’ karakterini çıkartarak gönderilmek istenen sayıyı elde ediyorlar ve bu sayı değerlerini x ve y değişkenlerinde tutuyorlar.

Ardından dayanak noktaları ile ortadaki nokta arasındaki uzaklıkları yani noktaların çemberlerinin yarıçaplarını bulmam gerekliydi, bu aşamada RSSI (Alınan Sinyalin Güç Değeri) kullanılacaktı. Dayanak noktalarından gönderdiğim radyo sinyallerinin ortadaki noktaya erişim gücü bu projede dayanak noktalarının kapsama alanlarının yani çemberlerinin yarıçapı olacaktı. Her dayanak noktasından gelen RSSI değerlerini ortadaki cihazda bir dizide tuttum bu sayede denklem çözümünde istenildiği zaman erişilebilecekti.

Bu aşamada denklem için gerekli bütün bilgiler elimdeydi ve bu andan sonra denklemi çözme fazına geçtim.

Gauss Yok Etme Metodu

Elimizde daha önce bulduğumuz bütün değişken katsayılarını kullanarak 3 çember denklemi yazıp bu denklemleri matris formunda çözmek için Gauss Elimination (Gauss Yok Etme) metodu kullanılması gerektiğini hemen anlamıştım. Bu nedenle internetten uzun süre araştırma yapıp bunu bilgisayara nasıl uygulatacağımı anlamaya çalıştım fakat internetteki hiçbir kod yeterli değildi. Bu nedenle okulda tuttuğum notların bir kısmını tekrar edip küçük bir özet ile kodlama aşamasına geçtim. Bu yöntemi uygularken matrisin köşegenlerini 1 yapıp, alt ve üst üçgenleri nasıl 0 yapabileceğimi kağıt üzerinde tasarladım ve bunu aynı şekilde koda uyguladım.

Kod kabaca şu şekilde çalışıyor:

Matrisi alır, ilk satır ve ilk sütun 0 olup olmadığına bakar, 0 ise satırı en alta atar, 0 yada 1 değilse 1 yapar ve uyguladığı işlemleri aynı satırdaki tüm sütunlara uygular, bunu tekrar ederek son satıra kadar gider ve son satıra geldiğinde alt üçgeni 0 yapmış olur, son satıra geldiğinde aynı işlemleri tersten yapar ve bu sayede üst üçgeni de sıfır yapmış olur.

En son elimizde kalan matriste 1.satırın en sonundaki değer ilk değişkenin sonucu, 2.satırın en sonundaki değer ise son değişkenin sonucudur.

Bu aşamada matrisin ilk satırının son sütunundaki eleman x'e setlenir, ikinci satırın son sütunundaki eleman y'ye setlenir ve bu şekilde istenen değerler bulunmuş olur. İstenirse değerler, bu projede olduğu gibi, ekrana yazdırılır.

Projenin Çalıştırılması

Coojayı açın ve File > Open Simulation > Browse yolunu izleyip açılan pencerede projenin kaynak kodunun bulunduğu rpl-udp.csc dosyasını seçin. Proje yüklendikten sonra açılan ekranda alttaki 3 kutucuk dayanak noktalarımızın koordinatlarını setleyeceğimiz kutular ve Mote Output yazan kısım ağı izleyebileceğimiz alan.

Dayanak noktalarının koordinatlarını setlerken sinyal uzaklıklarının ortadaki düğümle dayanak noktaları arasındaki mesafeden büyük olmamasına dikkat edin. Mesela dayanak noktalarını (-20,0), (0,20), (20,0) olarak seçerseniz ve bu dayanak noktaları ile ortadaki düğüm arasındaki uzaklık 20'den büyükse ortadaki noktalar gerçek değerini veremez, çünkü yarıçapı ortadaki düğümle dayanak noktası arasında olan bir çember bulunduğu konumla orantılı olmalıdır.

Örneğin 1.noktayı (-40,0), 2.noktayı (0,40), 3.noktayı (40,0) olarak setleyelim

İlk dayanak noktasının seri girişine setx -40 komutunu girip enterleyin ve ardından sety 0 komutunu girin. İkinci dayanak noktasının seri girişine setx 0 komutunu girip enterleyin ve ardından sety 40 komutunu girin.Son olarak üçüncü dayanak noktasının seri girişine de setx 40, sety 0 komutlarını girin.

Bu sayede dayanak noktalarımızın konumları belirlendi ve şimdi herhangi bir dayanak noktasından hesapla komutu ile ortadaki sunucunun konumun hesaplanması isteğini gönderin. Mote output penceresinde ortadaki cihazın dayanak noktalarına uzaklıklarını ve cihazın konumunu 'İstenen nokta: (x , y)' olarak görebilirsiniz.

Network penceresinden 1 numaralı düğümü(sunucu) 3 dayanak noktasının kapsadığı herhangi bir yere götürün ve ardından herhangi bir dayanak noktasının seri girişinden hesapla komutunu girin, Mote Output penceresinde ortadaki cihazın yeni konumunu göreceksiniz.

Kaynakça

-Fundamentals of Wireless Sensor Networks: Theory and Practice

Waltenegus Dargie and Christian Poellabauer © 2010 John Wiley & Sons Ltd.

-https://math.stackexchange.com/questions/884807/find-x-location-using-3-known-x-y-loca tion-using-trilateration

-http://contiki.sourceforge.net/docs/2.6/

-https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja

-https://sourceforge.net/p/contiki/mailman/message/34243244/

-https://en.wikipedia.org/wiki/Gaussian_elimination