T.C.
MALTEPE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

MOBİL SENSÖR CİHAZLARI KULLANILARAK ISI VE NEM DEĞERLERİNİN ÖLÇÜMÜ VE DEĞERLENDİRİLMESİ

Önder Mustafa GEREN

Yüksek Lisans Tezi

Tez Danışmanı
Yrd. Doç. Dr. Şenol Zafer ERDOĞAN

İSTANBUL – 2010

T.C.
MALTEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

MOBİL SENSÖR CİHAZLARI KULLANILARAK ISI VE NEM DEĞERLERİNİN ÖLÇÜMÜ VE DEĞERLENDİRİLMESİ

YÜKSEK LİSANS TEZİ

Önder Mustafa GEREN

Tez Danışmanı
Yrd. Doç. Dr. Şenol Zafer ERDOĞAN

İSTANBUL – 2010

Bu tez çalışması, Maltepe Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... / ..... / ..... tarih ve ..... / ..... sayılı kararıyla oluşturulan jüri tarafından ***Bilgisayar Mühendisliği Yüksek Lisansı Tezi*** olarak kabul edilmiştir.

JÜRİ

Yrd. Doç. Dr. Şenol Zafer ERDOĞAN
Danışman

Yrd. Doç. Dr. Turgay Tugay BİLGİN
Üye

Yrd. Doç. Dr. Birim Balcı Demir
Üye

ÖZET

Yüksek Lisans Tezi, Mobil Sensör Cihazları Kullanılarak Isı ve Nem Değerlerinin Ölçümü ve Değerlendirilmesi, T.C. Maltepe Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı.

Son yıllarda kablosuz teknolojiler ve kablosuz iletişim alt yapıları hem akademik hayatta hem de ticari hayatta giderek gelişmekte ve önemini arttırmaktadır. Kablosuz teknolojilerin içinde yer alan kablosuz sensör ağlar ise özellikle son yıllarda en hızlı ve en çok gelişen ve gelecek senelerde yoğun kullanımı olacak bir ağ teknolojisidir. Kablosuz sensör ağlar, askeri uygulamalar, sağlık uygulamaları, tabiat izlemesi, akıllı ev uygulamaları gibi pek çok değişik alanda uygulanmaktadır.

Kablosuz sensör ağlar, sensör düğümü olarak adlandırılan küçük fakat akıllı cihazların bir araya gelerek bir ağ oluşturmasından meydana gelmektedir. Sensör düğümleri üç ana modülden oluşmaktadır. Birinci modül sensör düğümün diğer sensör düğümleriyle haberleşmesini sağlayan radyo modülüdür. Kablosuz sensör ağlar haberleşme protokolü olarak IEEE 802.15.4 ZigBee protokolünü kullanmaktadır. ZigBee protokolü kablosuz sensör ağlarının düşük güçte ve düşük maliyette çalışma ilkesi göz önünde bulundurularak geliştirilmiş bir teknolojidir. Sensör düğümlerinin ikinci ana modülü sensör mödülüdür. Sensörler ısı, nem, ışık, ses, hareketlilik, gaz gibi çevresel değerleri ölçüp bunları sayısal yada analog elektriksel sinyallere çevirirler. Üçüncü ana modül ise kontrol modülür. Bu modül mikrodenetleyici, hafıza, güç kaynağı birimlerinden oluşur, sensör düğüm tarafından yapılması istenen fonksiyonlar için sensör düğümün hafızasına yüklenen komutları işler, sensör düğümünün üzerindeki sensörleri ve radyo modülünün çalışmasını denetler. Bu modül kısaca sensör düğümünün kontrolünden sorumludur.

Sensör düğümlerin en büyük sorunlarından biri sınırlı güç kaynağına sahip olmasıdır. Sınırlı güç kaynağı çalışma ömrünün de sınırlı olması anlamına gelmektedir ve sensör düğümün sınırlı çalışma ömrünün olması kablosuz sensör ağının da çalışma ömrünü sınırlamaktadır. Kablosuz sensör ağlarının çalışma ömürlerini uzatmak için sensör düğümlerin en düşük enerji tüketimi ilkesiyle çalışması sağlanmalıdır. Bu

nedenle kablosuz sensör teknolojilerinde üzerinde en çok çalışma yapılan konuların başında enerji-verimli (energy-efficient) algoritmalar gelmektedir. Sınırlı hafıza ve işlem yükü ise sensör düğümlerinin yeteneklerini kısıtlayan diğer unsurlardır. Kablosuz sensör ağlarının avantajları ise düşük maliyette olmaları, tasarısız ağlar oluşturarak istenilen verilerin ana merkeze ulaşmasını sağlamalarıdır. Bu sayede bir bölge içinde çok sayıda sensör düğüm rastgele serpilerek kablosuz sensör ağı oluşturmak mümkündür.

Tezin birinci bölümünde kablosuz sensör ağlarının genel özelliklerine, uygulama alanlarına ve kablosuz sensör ağının oluşturulmasında karşılaşılan zorluklara yer verilmektedir. İkinci bölümde sensör düğümlerin özellikleri, TinyOS işletim sistemi ve NesC programlama dilinin özellikleri anlatılmıştır. Üçüncü bölümde tezde veri toplamak için kullanılan Telosb sensör düğümü ve Cinterion XT65 cihazlarının özellikleri detaylı olarak anlatılmaktadır. Kablosuz sensör ağları kullanılarak mobil araçların üzerinde bulunan sensör düğümlerinin belirli bir bölgede farklı konumlarda ve farklı zamanlarda dolaştırılarak konum, ısı ve nem bilgilerinin toplanıp merkez istasyona gönderilmesi işlemlerini gerçekleyen sistemin mimarisi anlatılmaktadır. Tezin dördüncü bölümünde bilgi çıkarımı ve kümeleme analizi anlatılmaktadır. Sensörlerden gönderilen verilerin merkez istasyonda işlenip ilgili bölgenin ısı ve nem haritalarının çıkarılmasını sağlayan algoritmalar anlatılmaktadır. Sonuç bölümünde ısı ve nem haritalarının oluşturulması kapsamında yapılan çalışmalar ve deneylerden bahsedilmekte, kazanılan başarımlar açıklanmaktadır.

Bu tez 2010 yılında yapılmıştır ve 63 sayfadan oluşmaktadır.

Anahtar Kelimeler: Kablosuz Sensör Ağlar, TinyOS, NesC Programlama Dili, Veri Madenciliği, Kümeleme Analizi.

ABSTRACT

Master Thesis, Evaluation and Measurement of Temperature and Humidity Values Using Mobile Sensor Devices. T.C. Maltepe University, Graduate School of Natural and Applied Sciences, Department of Computer Engineering.

In recent years, wireless technologies and wireless communication infrastructures as well as commercial and academic life, are gradually developing and increasing their importance in life. Wireless technologies in wireless sensor networks especially in recent years, are the fastest and most developing network technology and will have an intensive use in the next few years. Wireless sensor networks, such as military applications, medical applications, habitat monitoring, smart home applications are applied in many different areas.

Wireless sensor networks consist of small but smart devices called as sensor node that create a network with coming together. The sensor nodes consist of three main modules. The first module is a radio module that provides the sensor node to communicate with other sensor nodes. Wireless sensor networks as the communication protocol uses IEEE 802.15.4 ZigBee protocol. ZigBee protocol is a technology that has been developed by considering the low-cost and low-power operation principle of wireless sensor network. The main module of sensor nodes is sensor module. Sensors measure environmental values such as heat, humidity, light, sound, movement, gas and convert them to digital or analog electrical signals. Third main module is controling module. This module consists of microcontroller, memory, and power supply units and operates the sensor node's memory installed commands to do the desired functions, supervises the operation of sensors on sensor node and radio module. Briefly this module is responsible of controling the sensor node.

One of the challenges of the sensor nodes is having limited power source. Limited power source means that a limited service life and sensor node that have a limited service life limits the service life of wireless sensor network. To prolong the service life of wireless sensor networks, the principle of low energy consumption should be provided for operation of the sensor nodes. Therefore, most work on wireless sensor

technologies, energy-efficient algorithms are at the beginning of the issues. Most of the subjects researching on wireless sensor technologies are energy-efficient algorithms. Limited memory and processing load are the other factors that restrict the capabilities of the sensor nodes. The advantages of wireless sensor networks are being low cost, creating ad-hoc networks to transport the desired data to the main server. In this way, wireless sensor network can be built with randomly scattering a large number of sensor node in a region.

In the first part of the thesis, general characteristics of wireless sensor networks, the application areas and the encountered difficulties in the creation of wireless sensor network are presented. In the second part features of the sensor nodes, TinyOS operating system and features of NesC programming language are explained. In the third part, feature of Telosb sensor node and Cinterion XT65 device which are used to collect data are described in detail. The architecture of the system that uses wireless sensor network and sensor nodes on vehicles that collects location, temperature and humidity datas in a certain region at different time , in different locations and sends them to the main server is described. The fourth section of the thesis knowledge extraction and clustering analysis are described. The algorithms which provide extraction of temperature and humidity maps of related region with data that are coming from sensor nodes and processed in main server. In the conclusion section, acquired successes, studies and experiments that are part of construction of temperature and humidity maps are explained.

This thesis has been completed in 2010 and consists of 63 pages.

Keywords: Wireless Sensor Networks, TinyOS, NesC Programming Language, Data Mining, Clustering Analysis.

TEŞEKKÜR

Tezime başlamamdan bitirmeme kadar bana devamlı yol gösteren ve destek veren tez danışmanım Sayın Yrd. Doç. Dr. Şenol Zafer Erdoğan'a, yardımlarından dolayı Sayın Veysel Karadağ'a ve bana her zaman destek olan aileme teşekkürü bir borç bilirim.

İÇİNDEKİLER

KISALTMALAR

Kısaltma	İngilizcesi	Türkçesi
CPU	Central Processing Unit	Merkezi İşlem Birimi
GPRS	General Packet Radio Service	Genel Paket Radyo Servisi
GSM	Global System for Mobile Communications	Mobil İletişim İçin Küresel Sistem
MEMS	Microelectromechanical systems	Mikro Elektronik ve Mekanik Sistemler
FIFO	First In First Out	İlk Giren İlk Çıkar
J2ME	Java 2 Platform, Micro Edition	Java 2 Mikro Versiyonu
GPS	Global Positioning System	Küresel Konumlama Sistemi
LEAP	Localized Encryption and Authentication Protocol	Yerelleştirilmiş Şifreleme ve Doğrulama Protokolü

ŞEKİLLER

Sayfa

TABLOLAR

Sayfa

DENKLEMLER

Sayfa

KODLAR

Sayfa

1. KABLOSUZ SENSÖR AĞLAR

Kablosuz sensör ağı, bulunduğu ortamdan sahip olduğu yetenekler neticesinde veriler toplayan ve bu verileri diğer sensör düğümleri üzerinden merkez istasyona ileten sensör düğümlerinin oluşturduğu bir ağ yapısıdır.

Son yıllarda Mikro Elektronik ve Mekanik Sistemler(MEMS) alanındaki gelişmeler kısa mesafede iletişim kurabilen sensör düğümlerinin gelişmesini, üretim maliyetlerinin düşmesini, güç tüketiminin azalmasını ve sensör yeteneklerinin artmasını sağlamıştır. Sensör düğümleri temel olarak veri toplama, veri üzerinde işlemler yapma ve iletişim kurma yeteneklerine sahiptirler. Sensör ağlarında bulunan sensör düğümleri birbirleri ile veri alış-verişi gerçekleştirebilirler ve sahip oldukları işlemci ve hafıza elemanlarının yetenek ve kapasiteleri doğrultusunda veri üzerinde işlem yapabilirler. Ebatları çok küçük olduğundan kullanılmak istenen bölgelere gelişi güzel atılabilirler. Sensör düğümler kendi aralarında haberleşerek, elde etmiş oldukları verileri işleyerek merkez istasyona iletirler. Sensör ağlarının bulundukları ortamda sıcaklık, basınç, nem, ses, gaz gibi değişiklikleri algılayıp ölçebilen birçok farklı yetenekte sensörlere sahip olmaları, sensör ağlarının çok çeşitli uygulama alanlarında kullanılabilmesini sağlar. Bu uygulama alanları aşağıdaki gibi sınıflandırılabilir:

- Askeri uygulamalar: Dost ve düşman kuvvetleri ve ekipmanı hakkında bilgi toplama, savaş alanı ve sınır bölgelerinin kontrolü, hedefleme, savaş zayiatı belirleme, nükleer, biyolojik veya kimyasal saldırı algılama gibi alanlarda kullanılmaktadır. [3]
- Çevre uygulamaları: Isı, nem ve biyolojik haritaların çıkarılması, canlıların hareketlerini izleme, kimyasal tehditleri önceden algılama, orman yangınlarını ve sel felaketlerini tespit etme fonksiyonlarını yerine getirirler. [3]
- Sualtı uygulamaları: Deniz tabanı hakkında bilgi toplama, deniz akıntılarını algılama/izleme, deniz canlılarını izleme. [3]
- Sağlık uygulamaları: Hastanedeki hasta ve doktorların izlenmesi, hastaların sağlık durumunu anlık izleme. [3]

- Ev uygulamaları: Isıtma, soğutma sistemlerinin kontrolü, ev içi hareketlilik izleme ve takip gibi akıllı ev otomasyonları. [3]
- Ticari uygulamalar: Ofis binalarında çevre kontrolü, etkileşimli müze, araba hırsızı yakalama ve görüntüleme, araç izleme ve algılama. [3]
- Bilimsel uygulamalar: Uzay keşif çalışmaları, kimyasal etkileşim araştırmaları, coğrafi çalışmalar. [3]

1.1. Kablosuz Sensör Ağlarının Özellikleri

Kablosuz sensör ağlarının başlıca özellikleri ve bu özelliklerin getirdiği avatajlar ve dezavantalar, kablosuz sensör ağları uygulamalarında karşılaşılan güçlükler aşağıdaki gibi sıralanabilir.

1.1.1. Güç Kaynağı

Her kablosuz sensör ağının en önemli dezavantajı sınırlı çalışma ömrüdür. Sensör düğümlerinin sınırlı enerji kaynaklarına sahip olmaları kablosuz sensör ağlarının ömürlerini sınırlamaktadır. Sensör düğümlerinin en çok enerji tüketen birimi haberleşme birimleridir. Sensör düğümleri arasındaki haberleşme ihtiyacı minimum olacak şekilde algoritmalar ve protokoller geliştirmek, kablosuz sensör ağlarının çalışma ömürlerini arttıracaktır. [1]

1.1.2. Haberleşme Mesafesi

Kablosuz sensör ağlarının en önemli özelliklerinden biri sensör ağının haberleşme mesafesidir. Sensör düğümlerin düşük maliyetli olması ve uygulamanın yapıldığı sahaya kolayca yerleştirilmesi sebebiyle kablosuz sensör ağları geniş kapsama alanına sahip olabilirler. Sensör düğümlerinin dinamik ağ topolojilerine uygun olması sebebiyle çalışma zamanında dahi sensör ağının kapsama alanı genişletilebilir veya daraltılabilir. Bununla beraber sensör ağın kapsamı alanı genişledikçe sensör düğümleri arasında haberleşme trafiği artabilir bu da enerji tüketimlerini arttırır dolayısıyla kablosuz sensör ağının çalışma ömrünü kısaltır. [1]

1.1.3. Güvenilirlik

Bir sistemin, belirli şartlar altında kabul edilebilen zaman içinde fonksiyonlarını gerçekleme olasılığına güvenilirlik(reliability) denir. [1] Sensör düğümleri, enerjilerinin tükenmesinden, çevresel etkilerden yada fiziksel zararlardan ötürü çalışamaz duruma gelebilirler. Sensör düğümlerin bir veya birkaç tanesinin çalışamaz duruma gelmesi kablosuz sensör ağının çalışmasını etkilememelidir. Bu durum kablosuz sensör ağının hatalara karşı güvenilirliğini göstermektedir. Sensörlerin güvenilirlikleri çoğunlukla Poisson dağılımına dayanan Denklem 1'e göre modellenir. [2]

$$R_k(t) = e^{-t\alpha_k}$$

Denklem 1. Poisson Dağılımı [1]

$R_k(t)$: Sensör düğümünün güvenilirlik değeri

$\propto_k$: k numaralı sensör düğümün hata yapma oranı

t: Sensör düğümün test edildiği zaman aralığı

Sensör düğümlerde kullanılacak algoritmalar hesaplanan güvenilirlik değerine göre belirlenir.

1.1.4. Maliyet

Sensör ağların kurulum maliyetlerinin oldukça düşük olması sensör ağların avantajlarından biri olarak görülmektedir. Sensör düğümlerin uygulamanın yapılacağı sahaya yerleştirildikten sonra gerekli tüm işlemleri dışarıdan bir müdahaleye gerek duymadan kendileri yapabilmektedir. Bu şekilde kablosuz sensör ağların kurulum maliyetleri uygulamadan uygulamaya farklılık göstermekle birlikte düşük olmaktadır. Geliştirme maliyetleri kurulum maliyetlerine göre daha fazla olmaktadır. Geliştirme sürecinde sistemin güvenilirliğini arttırmak için çok sayıda test yapılmaktadır. Bu süreçde sistemin dayanıklılığı ve iletilen verilerin doğruluğu gibi bir çok parametre test edilmektedir. Gerçeklenen testler ve bu testlerde tespit edilen hataların çözüm işlemleri geliştirme maliyetini arttırmaktadır. Ancak bu hataların sistemin uygulamaya geçmesinden sonra tespit edilmesi toplam maliyeti daha fazla arttırabilmektedir. Üretim maliyetleri ise sensör düğümlerin algıladığı değişkenlerin sayısına ve çeşitliliğine, sensör düğümlerden beklenen veri işleme yeteneklerine göre değişmektedir. [1]

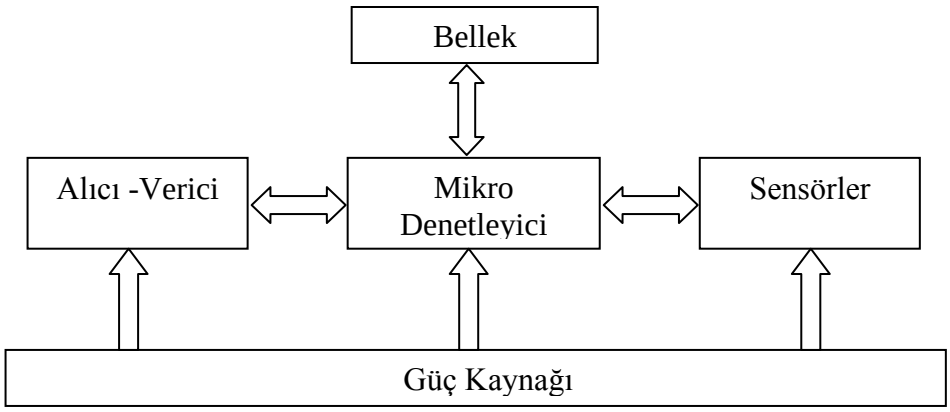
1.1.5. Güvenlik

Ticari ya da askeri uygulamalar dışında örneğin ısı, nem, ışık gibi çevresel verileri toplayıp işleyen kablosuz sensör ağlarında sensör düğümlerinin topladıkları verileri birbirlerine gönderirken verileri şifreleyip göndermelerine gerek olmayabilir. Ancak gizli bilgilerin toplanıp işlendiği bir ağda toplanan verilerin başkalarının eline

geçmemesi için veriler düğümler arasında şifrelenip gönderilir. Verilerin şifrelenmesi için birçok farklı kodlama ve şifreleme algoritmaları mevcuttur. Şifreleme aloritmalarına örnek olarak TinySec [9] ve LEAP [10] protokolleri gösterilebilir.Sensör düğümlerin veri işleme kapasitelerine göre uygun şifreleme tekniği seçilip ağın güvenliği sağlanabilir. Güvenliği arttırmak için kullanılacak güvenlik algoritmaları gönderilecek veri miktarını arttıracağı için ve sensör düğümünde daha fazla işlem yapılacağı için ağın enerji tüketimi artmaktadır dolayısıyla ağın çalışma ömrünü kısaltmaktadır. [1]

1.2. Kablosuz Sensör Düğümleri ve Özellikleri

Sensör düğümleri ortamdaki verileri toplayan, topladığı verileri kısmen işleyen ve diğer düğümlere ileten kablosuz sensör ağların en küçük birimidir. Sensör düğümün ana bileşenleri mikrodenetleyici, bellek, alıcı-verici, güç kaynağı ve bir veya daha fazla olabilen sensör bileşenidir. [5] Sensör düğümünün genel mimarisi Şekil 1'de gösterilmektedir.



Şekil 1. Sensör düğümü mimarisi

MikroDenetleyici: Sensöre yüklenen programla belirlenen komutları işler, sensör düğümü üzerindeki diğer bileşenlerin işlevselliğini denetler. [5]

Alıcı-Verici: Radyo dalgalarını kullanarak diğer düğümlere verileri ileten ve diğer düğümlerden gönderilen verileri alıp mikrodenetleyiciye ileten sensörün iletişim birimidir. Sensör düğümleri aralarında haberleşmek için 433 Mhz ile 2.4 Ghz arasındaki iletişim frekansını kullanırlar. [5]

Bellek: Mikro denetleyici üzerindeki bellek, enerji kullanımı açısından en uygun bellektir. Bunun dışında maliyetinin az olması ve depolama kapasitesinin mikro denetleyici üzerindeki belleğe göre daha büyük olmasından Flash bellekler kullanılmaktadır. İhtiyaç duyulan bellek gereksinimleri uygulamadan uygulamaya değişmektedir. [5]

Güç kaynağı: Sensör düğümünde gerçekleşen veri toplama, işleme ve düğümler arası iletişim nedeniyle güç kaynağına ihtiyaç duyulmaktadır. Sensör düğümleri mobil aygıtlar olduğu için üzerlerinde bulunan güç kaynakları sınırlı enerji ye sahip olmaktadır. Sensör düğümünde en fazla enerji tüketen bileşen alıcı-verici bileşenidir. Veri toplama ve işleme için enerji tüketimi daha azdır. [5] Gelişen enerji teknolojisiyle birlikte sensörler güneş enerjisi, ısı enerjisi gibi yenilenebilir enerji kaynaklarını da kullanabilecek yetenekte geliştirilmektedirler.

Sensörler: Sensörler sıcaklık, nem, basınç gibi fiziksel değerleri ölçüp sayısal yada analog olarak ölçüm sonucunu üreten bileşenlerdir. Alıcı-verici bileşeninden sonra en çok enerji tüketen bileşendir. [5]

2. TINYOS

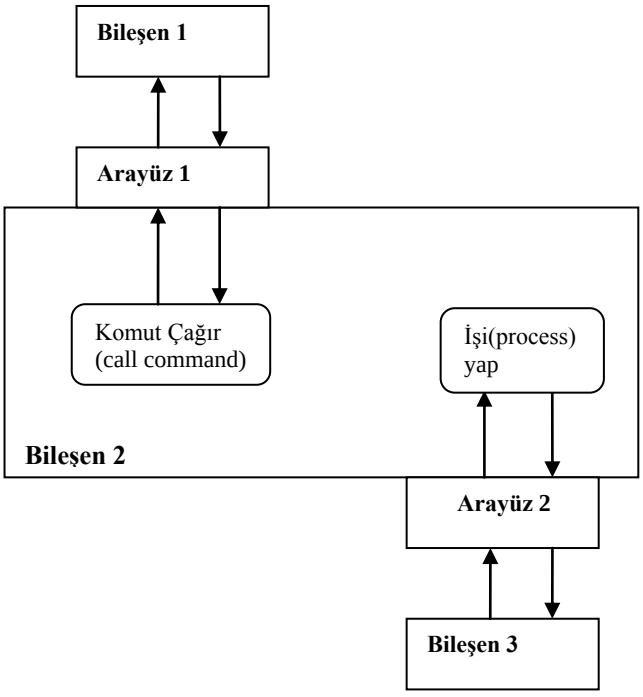
TinyOS, kablosuz sensör ağlarında kullanılmak üzere ücretsiz ve açık kaynak kodlu olarak dağıtılan gömülü bir işletim sistemidir. TinyOS işletim sistemi, kablosuz sensör ağlarının gereksinimlerini destekleyecek şekilde tasarlanmıştır. Berkeley Üniversitesi tarafından geliştirilmiştir. Kablosuz sensör ağlarının en büyük kısıtlarından olan düşük hafıza boyutu sorunu için uygun bir çözüm sunan, geliştirmeye ve uygulamaya elverişli, düşük miktarda hafıza gerektiren bileşen tabanlı bir mimariye sahiptir. TinyOS işletim sistemi, yapısı C programlama dili tabanlı olan NesC programlama dili ile yazılmıştır. TinyOS'in bileşen kütüphanesinde ağ protokolleri, dağıtılmış hizmetler, sensör sürücüleri ve veri toplama araçları bulunmaktadır. TinyOS'in olay bazlı işletim modeli daha detaylı güç yönetimine imkan sunduğu için de sensör ağlarının bir başka büyük kısıtı olan pil ömrünü uzatmaktadır. [1]

TinyOS işletim sisteminde, bilinen işletim sistemlerinden farklı olarak, çekirdek ve kullanıcı katmanı diye bir ayrım bulunmamaktadır. Uygulamalar donanımsal kaynaklara direk olarak erişir. Bu yapı, geliştirilen uygulamaların performasının arttrılmasına ve gerçek-zamanlı işlemler yapılmasına olanak sağlamaktadır. TinyOS bir dosya sistemine sahip değildir, statik bellek yönetimini kullanır. Böylece dinamik bellek yönetimlerinden oluşabilecek bellek yönetim problemlerinin de önüne geçilmiş olunu Tek bir evrensel bellek oluşturulur ve sanal bellek oluşturulmadan uygulamalar bu evrensel belleği kullanır. TinyOS işletim sistemi, sensör düğümün güç kaynağının daha verimli kullanılması ve çalışma ömrünün arttırılması için "acele et ve uyu" (hurry up - and-sleep) diye bilinen bir strateji izler. Bu strateji, enerji tüketimini azaltmak için mikrodenetleyicinin mümkün olduğunca uyuması prensibine dayanır. Mikro denetleyicinin yerine getireceği görevler, olabilecek en kısa sürede bitirilip, mikro denetleyici en az güç harcayacağı uyku modunda bekletilir. TinyOS'te kullanılan kesme ve zamanlayıcı fonksiyonları uygulamaların sonsuz döngüler için bekleme zorunluluğunu ortadan kaldırmaktadır. C programlam dilinde olduğu gib fonksiyon çağrıları yerine makrolar kullanılarak kod okunabilirliği arttırılabilir. Bunların

gerçekleştirilebilmesi için, TinyOS işletim sisteminde bileşen tabanlı fonksiyonel programlama yöntemi kullanılır. [1]

2.1. Bileşenler (Components)

TinyOS işletim sisteminin temel taşını oluşturan bileşenler, nesneye yönelik programlama mimarisini oluşturan nesne (object) yapısına benzer olarak içerisinde fonksiyonlara ve değişkenlere sahiptirler. Bileşenler, arayüzler(interfaces) sayesinde diğer bileşenlere bağlanırlar(wiring) ve tanımlanan olaylar(events) gerçekleştiğinde istenilen görevleri(tasks) başlatırlar. Bileşenler arayüzleri sağlarlar(provides) ya da kullanırlar(uses). TinyOS işletim sistemindeki sistem bileşenleri, kendilerine gelen olay çağırılarına cevap vererek çalışır. Bir olay çağrısı meydana geldiğinde, bu çağrı ile tetiklenen ilgili görev çalışır ve istenilenleri yerine getirir. Daha sonra diğer çağırıları beklemek için sisteme geri döner. Bileşenler konfigürasyon ve modül olmak üzere iki yapıdan oluşmaktadır. [1]



Şekil 2. Bileşen komut çağrımı gösterimi [1]

Şekil.2'de TinyOS bileşenlerinin arayüzler üzerinden diğer bileşenler ile olan iletişim yapısı gösterilmektedir. Bir bileşen başka bir bileşenden veya donanımdan bir sinyal(signal) aldığında, "command" ifadesiyle belirtilen ilgili fonksiyonda tanımlanan işleri(process) gerçekler ve varsa üretilen sonucu döndürür. Sinyaller, belirli bir olay gerçekleştiğinde, donanım veya bileşen tarafından gönderilen bildirimlerdir. "command" ifadesi diğer bileşenlerin kullanması için gerçeklenen fonksiyonları tanımlamak için kullanılır. Aynı şekilde, bir bileşen ilgili arayüz üzerinden başka bir bileşene "call" ifadesi ile komut çağrısı yapabilir. "call" ifadesi diğer bileşenler tarafından sağlanan fonksiyonları çağırmak için kullanılır. Komutlar, daha düşük seviyedeki bileşenlere iletilirken olaylar, komutlara veya siyallere cevaben daha yukarı seviyelere iletilirler. [1]

2.1.1. Konfigürasyon (Configuration)

Konfigürasyon, bir bileşenin görevleri yerine getirmesi için diğer hangi bileşenleri kullanacağını ve kullanılan diğer bileşenlerle olan ilişkileri belirler. İki bileşenin birbirleriyle ilişkilendirilmesi arayüzler üzerinden gerçekleşir. Bağlama olarak adlandırılan yazım şekli ile bir bileşenin yapısında bulunan diğer bileşenler birbirleri ile bağlanarak bir görevin yerine getirilmesini sağlayabilir. Konfigürasyon dosyaları, "configuration" fonksiyonu ile başlarlar. Bu fonksiyonda kullanılan ve sağlanan hizmetler belirtilir. Ardından "implementation" fonksiyonu ile birbirine bağlanacak bileşenler ve arayüzler tanımlanır. Burada herhangi bir uygulama kodu yazılmaz. Bağlama işlemini gerçeklemek için "–>, <–, =" operatörleri kullanılır. Bağlama operatörleri Tablo 1'de gösterildiği şekilde kullanılmaktadır. [1]

Tablo 1. Bağlama Operatörleri

Operatör	İfade
->	Kullanıcı.arayüz -> Sağlayıcı.arayüz
<-	Sağlayıcı.arayüz <- Kullanıcı.arayüz
=	Sağlayıcı1.arayüz = Sağlayıcı2.arayüz Kullanıcı1.arayüz = Kullanıcı2.arayüz

"->" operatörü, sol tarafında belirtilen arayüzün sağ tarafında belirtilen arayüzü kullandığını tanımlar. Benzer şekilde "<-" operatörü sağ tarafında belirtilen arayüzün sol tarafında belirtilen arayüzü kullandığını tanımlar. "=" ifadesi sağ tarafında tanımlanan bileşenin sağladığı arayüzün, sol tarafında tanımlanan bileşen tarafından da sağlandığını tanımlar. Aynı şekilde "=" ifadesinin sağ tarafında tanımlanan bileşenin kullandığı arayüzün, sol tarafında tanımlanan bileşen tarafından da kullanıldığını tanımlar. "=" operatörü bir bileşenin diğer bir bileşenin sağladığı ve kullandığı fonksiyonları farklı bir isimle tanımlamak için kullanılır.

Birbirine bağlancak bileşenlerin uygun yapıda olma gerekliliği vardır. İki bileşen benzer yapıdaki arayüzleri üzerinden birbirine bağlanabilir. Yada komutlar kendi aralarında ve olaylar da kendi aralarında birbirlerine bağlanabilirler. Örneğin, gönderme işlemi gerçekleyen "sending" arayüzü, alma işlemi gerçekleyen "receiving" arayüzüne bağlanamaz. Ancak "sending – sending" arayüzü birbirine bağlanabilir. Kod 1'de Blink bileşeninin konfigürasyon dosyası gösterilmektedir. "implementation" fonksiyonu içinde Blink bileşenin kullandığı diğer bileşenler belirtilmektedir. Kullanılan bileşenlerin arayüzlerinin diğer bileşenlerin arayüzlerine bağlama operatörleri ile nasıl bağlandığı gösterilmektedir. [1]

```
configuration Blink{
}
implementation {

components MainC, BlinkC, LedsC, TimerMilliC();
  BlinkC.Boot -> MainC.Boot;
  BlinkC.Timer -> Timer.Timer;
  BlinkC.Leds -> LedsC.Leds;
}
```

Kod 1. Blink konfigürasyon dosyası

Kod 1'e bakıldığı zaman BlinkC bileşeninin Boot arayüzü, MainC bileşeninin Boot arayüzüne bağlandığı görülmektedir. Bu bağlama sayesinde BlinkC uygulaması başlatıldığında MainC bileşeninin Boot fonksiyonunun çağırılması sağlanmaktadır. Aynı şekilde BlinkC bileşeninin Timer arayüzü TimerMillicC bileşeninin Timer arayüzüne, Leds arayüzü LedsC bileşeninin Leds arayüzüne bağlanmaktadır. Bu bağlama sayesinde TimerMilliC bileşeninin Timer arayüzü tarafından sağlanan fonksiyonlar BlinkC bileşeni tarafından kullanılabilmektedir. Aynı şekilde LedsC bileşeninin Leds arayüzü tarafından sağlanan fonksiyonlar BlinkC bileşeni tarafından kullanılabilmektedir.

2.1.2. Modül (Modul)

Modül bileşenin yerine getireceği işlerin uygulama kodlarını içerir. Modül sahip olduğu değişken ve fonksiyonları ile sağladığı arayüzlerin komutlarını(commands) ve kullandığı arayüzlerin olaylarını(events) tanımlar. Bileşenin tanımladığı olaylar gerçekleştiğinde başlatılacak görevlerin uygulama kodları modülün içerisinde tanımlanır. Bir bileşenin modülü tanımlanmak istendiğinde modülün isminin önüne "**module**" ifadesi yazılır. Modül ifadesinin tanımladığı alan(scope) içerisinde bileşenin kullanacağı ve sağlayacağı arayüzler tanımlanır. Bileşenin kullanacağı arayüzü tanımlamak için arayüzün isminin önüne "**uses interface**" ifadesi, sağlayacağı arayüzün önüne "**provides interface**" ifadesi getirilir. Aşağıda yer alan Kod 2'de PeriodicReaderC bileşenin sağladığı ve kullandığı arayüzler gösterilmektedir. PeriodicReaderC örneğinde, bileşenin StdControl arayüzünün komutlarını tanımladığı, Timer ve Read arayüzlerinin olaylarının kullanıldığı belirtilmektedir.

```
module PeriodicReaderC
{
  provides interface StdControl;
  uses interface Timer<TMilli>;
  uses interface Read<uint16_t>;
}
```

Kod 2. PeriodicReaderC modülü arayüz tanımları [4]

Modülün uygulama kodları uygulama(implementation) alanının içerisinde tanımlanır. Uygulama alanı "**implementation**" ifadesi ile belirtilir. Kod 3'de PeriodicReaderC bileşeninin modül kodlarının tamamı gösterilmektedir.

```
module PeriodicReaderC  {

     provides interface StdControl;
     uses interface Timer<TMilli>;
     uses interface Read<uint16_t>;
}
implementation  {

     uint16_t lastVal = 0;

     command error_t StdControl.start() {
          return call Timer.startPeriodic(1000);
     }
     command error_t StdControl.stop() {
          return call Timer.stop();
     }
     event void Timer.fired() {
          call Read.read();
     }
     event void Read.readDone(error_t err, uint16_t val)
{
          if (err == SUCCESS) {
               lastVal = val;
          }
     }
}
```

Kod 3. PeriodicReaderC modülü [4]

PeriodicReaderC bileşeninin modülünde, bileşenin sağladığı StdControl arayüzünün start(), stop() komutları ve. kullanılan Timer arayüzünün fired olayı ve Read arayüzünün readDone olayları tanımlanmaktadır. Komut fonksiyonlarının çağırmak için **call** komutu kullanılmaktadır. PeriodicReaderC bileşeni, konfigürasyon dosyasında

bağlanan sensörün ürettiği verileri periyodik olarak örnekler ve aldığı örnekleri yerel bir değişkende saklar. StdControl arayüzünün start komutu bileşene gönderildiğinde PeriodicReaderC bileşeni Timer arayüzünün startPeriodic fonksiyonunu 1000 değerli parametre ile çağırmaktadır. Sensör düğümün zamanlayıcısı 1000 ms. periyodunda Timer.fired olayını çağırılmasını sağlayan sinyali(signal) gönderir ve Timer.fired olayı çağırılır. Timer.fired olayı çağırıldığında Read.read() fonksiyonu çağırılacaktır. Read.read() fonksiyonunun çağırılmasıyla sensörde okuma işlemi gerçekleşecek ve okuma işlemi bittiğinde Read.readDone() olayı sensörün okuduğu veri ile birlikte çağırılacaktır. Read.readDone fonksiyonunda parameter olarak gönderilen veri, lastVal isimli değişkene yazıldıktan sonra bileşen görevini tamamlamış olacak ve bir sonraki Timer.fired olayı için sinyal gönderilene kadar uykuda bekleyecektir.

2.2. Arayüzler (Interfaces)

Arayüzler, bileşenler arasında iki yönlü haberleşmeyi sağlayan yapılardır. Arayüzler komut(command) ve olay(event) fonksiyonlarından oluşmaktadır. Arayüz dosyaları "**interface**" fonksiyonu içinde tanımlanırlar. Arayüzleri kullanan bileşenler arayüzün olay fonksiyolarını, arayüzleri kullanan bileşenler olay fonksiyonlarını tanımlamak zorundadırlar. Kod 4'de TinyOS standart kütüphanesinde bulunan Timer arayüzü gösterilmektedir.

```
interface Timer {

     command void startPeriodic(uint32_t interval );
     command void  stop ();
     event void fired ();
}
```

Kod 4. Timer Arayüzü

Timer arayüzünü kullanan bileşenler komut fonksiyonu olan "startPeriodic" fonksiyonu ile zamanlayıcıyı başlatabilirler. "startPeriodic" fonksiyonu parametre olarak zamanlayıcının çalışma periyodunu belirleyen milisaniye cinsinden tamsayı değeri alır.

“stop” komut fonksiyonu ile başlatılan zamanlayıcıyı durdurulur. Timer arayüzünü kullanan bileşenin ilgili arayüz üzerinden gönderdiği komutların görevlerini yapabilmesi için bileşenin ilgili arayüzün komut fonksiyonlarını tanımlaması gerekmektedir. Zamanlayıcı başladıktan sonra “fired” sinyali (signal) gönderilir ve bu sinyal Timer arayüzü üzerinden bu arayüzü kullanan bileşene gönderilir. Sinyali alan bileşen fired olayını tanımladığı için fired fonksiyonu çağırılır ve ilgili bileşende tanımlanan görevler icra edilir.

2.3. Görevler (Tasks)

Fonksiyonel programlamanın en önemli problemlerinden biri, işleyen bir fonksiyonun çalışma süresinin çok uzun sürmesinden dolayı gerçek-zamanlı yapılması gereken diğer fonksiyonların çalışmasının gecikmesine neden olmasıdır. Benzer şekilde, icra edilen fonksiyonun içinde oluşan bir hatadan dolayı fonksiyon sonsuz bir döngüye girerse yada çalışması hiçbir zaman bitmezse, diğer sistem çağrılarını dinleyen fonksiyonların ve oluşacak kesmelerin durmasına neden olabilecektir. Hesaplama süresi uzun olan fonksiyonlar için, TinyOS görev(task) adı verilen bir çalışma mekanizmasına sahiptir. Görevler, arka planda çalışarak, diğer fonksiyonları başlatan olaylara ve kesmelere engel olmazlar. Görevler, mikro-denetleyicinin boş olduğu zaman aralıklarında iş düzenleyici(scheduler) tarafından başlıtılarak icra edilmeleri sağlanır. Görevlerin çalışması, daha düşük seviyeli sistem çağırıları oluştuğunda durdurulabilirler. Böylece yeni bir olay çağrısı oluştuğunda çalışan görevler bekleme durumuna alınır ve sistemin yeni olayları gecikme olmadan mikro-denetleyicide işlemesi sağlanmış olunur. Görevler arasındaki öncelik iş düzenlyeci tarafından atanma sırasına göre belirlenmektedir. İş düzenleyici tarafından ilk atanan görev bitirilmeden diğer görevler başlatılmaz. İş düzenleyicinin kuyruğunda başlatılacak herhangi bir görev yoksa mikro-denetleyicinin saati haricindeki diğer birimleri uyku modunda bekletilir. Böylece güç kaynağının verimli bir şekilde kullanımı sağlanır. [1] Görevler başka görevleri başlatamaz. Olaylar veya komutlar görevleri başlatabilir. Bir fonksiyonu görev olarak çalışması için

fonksiyonun başına **task** ifadesi eklenir. Görevlerin başlatılması için görev fonksiyonunun çağırılacağı satırda fonksiyon isminin önüne post ifadesi eklenmelidir. Kod 5'de örnek bir görev ve görevi başlayan bir komut gövdesi gösterilmektedir.

```
command error_t Read.read() {
     post readDoneTask();
     return SUCCESS;
}
task void readDoneTask() {
     signal Read.readDone(SUCCESS, filterVal);
}
```

Kod 5. Görev fonksiyonu ve görev başlatılması [4]

Kod 5'de gösterildiği üzere, Read bileşeninde task ifadesiyle belirtilen "readDoneTask" isminde bir görev fonksiyonu tanımlanıyor. "readDoneTask" görev fonksiyonunda okuma işleminin tamamlandığını belirten readDone sinyali gönderiliyor. Benzer şekilde command ifadesiyle "read" komut fonksiyonu tanımlanıyor. "read" fonksiyonu çağırıldığında "readDoneTask" görevi başlatılıyor.

2.4. İş Düzenleyici (Scheduler)

TinyOS işletim sisteminde herhangi bir anda sadece bir işlem (process) çalışabilmektedir. Bu yüzden oldukça basit bir yöntem olan FIFO (İlk giren ilk çıkar) işlem yöntemi kullanılmaktadır. TinyOS'in yapısında 2 seviyeli bir zamanlama (scheduling) yöntemi kullanılmaktadır. Bu yöntemdeki birinci grubu görevler (tasks), ikinci grubu ise olaylar (events) oluşturmaktadır. Görevler hesaplamaların yapılmasında, olaylar ise veri akışının kontrol edilmesinde kullanılmaktadır. Geliştirilen yapıda olaylar, başka olayları veya görevleri oluşturabilirler. Ancak görevler, başka görevleri veya olayları oluşturamazlar. Olaylar donanımsal kesmeler (interrupts) tarafından oluşturulan düşük seviyeli olaylar ve yazılımlar tarafından oluşturulan yüksek seviyeli olaylar olarak ikiye ayrılmaktadır. Olaylar, fonksiyon çağırılması

(function calls) ile düşük seviyeden yüksek seviyeye doğru yayılabilirler. TinyOS'in yapısında yüksek seviyedeki bileşenler daha düşük seviyedeki bileşenlere komutlar (commands) aracılığıyla istekte bulunurlar. [1]

2.5. Atomik Yapılar (Atomics)

TinyOS işletim sistemi, çalışma zamanında kesmelerle bölünmemesi gereken yapılar için atomikleştirme adında bir çalışma mekanizma tanımlamaktadır. Fonksiyonel programlamada hatalara yol açan durumlardan biri farklı modüllerin aynı bellek alanına eş zamanlı olarak erişmeye çalışmasıdır. Bu durum yarışma (racing) durumu olarak adlandırılmaktadır. Örneğin bir görevin içerisinde yapılan bir işlem ortak bir bellek alanı kullanıyorsa ve ilgili işlemler tamamlanmadan başka bir görev tarafından bu ortak alana herhangi bir veri yazılmaması garanti altına alınak istenirse ilgili kod bloğu atomik kod olarak tanımlanır. Atomik kod blokları içerisinde yürütülen işlemlerin çalışma zamanı uzun sürerse sistemde oluşan kesmelerin işlenmesinde gecikmeler yaşanabilir. Dolayısıyla atomik kod blokları mümkün olduğunca kısa tasarlanmalıdır. Atomik kodlar genel olarak, diğer görevler tarafından ortak olarak kullanılan bellek bölgeleri olduğunda kullanılmaktadır. [1]. Atomik olarak çalışması istenen kod bloğunun başına **atomic** ifadesi eklenerek ilgili bloğun kapsadığı işlemler herhangi bir kesmeye uğramadan çalışması tamamlanır. Kod 6'da örnek bir atomik kod bloğu gösterilmektdir. Kod 6'da increment komutu çağırıldığında a değişkeninin değeri bir arttırılır ve b değişkeninin değeri a değişkeninin değerinin 1 fazlasına eşitlenir.

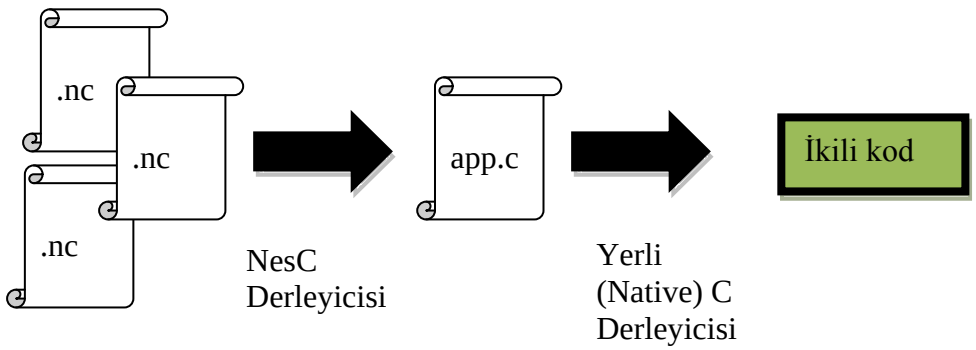
```
command bool increment() {
     atomic {
          a++;
          b = a + 1;
     }
}
```

Kod 6. Atomik kod bloğu [4]

2.6. NesC Programlama Dili

TinyOS işletim sistemi, bileşen tabanlı uygulamarı geliştirmek için tasarlanmış C tabanlı bir programlama dili olan NesC ile geliştirilmiştir. NesC, kablosuz sensör ağların düşük hafıza ve sınırlı enerji kısıtı göz önüne alınarak olay tabanlı çalışabilme ve modüler programlama özelliklerine sahiptir. NesC programlama kodu, öncelikle NesC önderleyicisi ile standart C koduna dönüştürülür. Dönüştürülen NesC kodu C derleyicisi kullanılarak ikili düzende mikrodenetleyiciye yüklenmeye hazır dosya haline getirilir. Şekil 3'de NesC programlama kodunun mikrodenetleyiciye yüklenme aşamasına getirilme adımları gösterilmektedir. [1]

TinyOS işletim sistemi statik bellek yönetimine sahip olduğu için geliştirilen programda kullanılan değişkenlerin bellekteki adresleri derleme zamanında belirlenir. Dolayısıyla NesC kodunda yapılacak optimizasyon işlemi derleme sürecinde gerçekleştirilir. Derleme zamanında yapılan bellek kullanım kontrolleri sayesinde, aynı bellek bölgesine aynı anda yazma (racing) gibi hatalar tespit edilebilir. [1]



Şekil 3. NesC derleme modeli [7]

NesC programlama dilinde Modül ve Konfigürasyon olmak üzere iki tip dosya bulunur. Modüller, bileşenlerin sağladığı görevleri ve diğer bileşenlerin sağladığı görevleri kullanan arayüz tanımlarını içerirler. Diğer bileşenlere veri gönderen veya diğer bileşenlerden veri alan bileşen, bu işlemleri modül yapısında gerçekleştirir. Bileşenin yerine getireceği işlemlere ait C kodları modüller içerisinde tanımlanır.

Konfigürasyonlar ise, bileşenlerin birbirleri ile olan ilişkilerini tanımlarlar. Konfigürasyon dosyasında bileşenin kullandığı diğer bileşenlerin uygun arayüzleri birbirine bağlanarak istenilen görevlerin gerçeklenmesi sağlanır. Bir bileşen hem konfigürasyon hem modül dosyalarından oluşabilir. Bu şekilde tanımlanan bileşenler istenilen görevleri yerine getirmek için, hem başka bileşenleri kullanırlar hem de kendi içinde tanımlanan görevleri kullanılarlar. Arayüzler NesC programlama dilinde, bileşenler arasında gerçekleşen iletişimin yapısını düzenler. Bu iletişimler bileşenlerin sağladığı hizmetler yada diğer bileşenlerden beklenen hizmetler şeklinde olabilir.

NesC programlama dilinde modüller öncelikle bileşenin sağladığı ve kullandığı hizmetlerin sırasıyla "provides" ve "uses" anahtar kelimeleriyle tanımlandığı "module" fonksiyon tanımı ile başlarlar. "module" fonksiyonundan sonra , "implementation" fonksiyon tanımı ile bileşenin yerine getireceği hizmetleri uygulama kodunun yazımı gelir. "implementation" fonksiyonda yerel değişkenler ve fonksiyonlar tanımlanmaktadır. [1]

2.6.1. Örnek Uygulama

Powerup programı çalışmaya başladıktan sonra 0 numaralı ledi yakar ve uykuya geçer. Powerup programının C dilinde uygulanması Kod 7'de gösterilmektedir.

```
#include "mote.h"

int main ()
{
     mote_init ();
     led0_on ();
     sleep ();
}
```

Kod 7. Powerup C dili uygulaması [7]

“mote.h” başlık dosyası sensör düğümün donanımını başlatan (mote_init), Led’i yakan (led0_on) ve düğümü uykuya geçiren (sleep) fonksiyonların ve sensör düğüm için diğer temel fonksiyonların deklarasyonunu içerir. Main fonksiyonu sensör düğümü başlatıldığında otomatik olarak çağırılır.

Powerup programının NesC ile uygulanması ‘Modül’ ve ‘Konfigürasyon’ olmak üzere iki bölümden oluşur. İlk bölümü olan PowerupC modülü Powerup uygulamasının ve PowerupC bileşeninin gerçekleyeceği işleri içerir. PowerupC modülü Kod 8’de gösterilmektedir.

```
module PowerupC {
      uses interface Boot;
      uses interface Leds;
}

implementation {
      event void Boot.booted () {
      call Leds.led0On ();
      }
}
```

Kod 8. PowerupC Modülü [7]

PowerupC modülü Boot ve Leds arayüzlerini kullanarak sistem ve donanım ile etkileşime girer. Modülün ‘implementation’ kısmında Boot arayüzünün booted olayı(event) tanımlanmaktadır. Sensör düğümü başlatıldığında booted olayı gerçeklenecek ve booted olayı gerçeklendiğinde Leds arayüzünün led0On fonksiyonu çağırılacaktır. Farklı platformlarda farklı sayıda ve renkte led bulunduğu için Leds arayüzünün ledXOn foksiyonları led0On, led1On ve led2On olarak numaralanmaktadır. PowerupC modülü ile Powerup programının C dili ile uygulanmasını karşılaştırdığımızda, NesC uygulamasında Boot.booted olayının C’de main’e karşılık geldiğiniz görürüz. C’de programlar fonksiyonlardan oluşur, NesC’de programlar bileşenlerden oluşur. C’de fonksiyonlar direk olarak birbiriyle haberleşebilir ancak NesC’de bileşenler arayüzler üzerinden haberleşirler. NesC arayüzleri Java arayüzlerine benzemektedir. Java arayüzlerine ek olarak, gelen çağrıları (callbacks) ayırt edebilmek için ‘command’ ve ’event’ ifadelerini içerir. Kod 9 ve Kod 10’da sırasıyla Boot ve Leds arayüzlerinin tanımları gösterilmektedir.

```
interface Boot {

     event void booted ();
}
```

Kod 9. Boot arayüzü

```
interface Leds {
     command void led0On ();
     command void led0Off ();
     command void led0Toggle ();

     command void led1On ();
     command void led1Off ();
     command void led1Toggle ();

     command void led2On ();
     command void led2Off ();
     command void led2Toggle ();
}
```

Kod 10. Leds arayüzü

Powerup programının ikinci bölümü olan PowerupAppC konfigürasyonu PowerupC modülünün TinyOS servislerine nasıl bağlanacağını belirler. Kod 11'de PowerupAppC konfigürasyon dosyası gösterilmektedir.

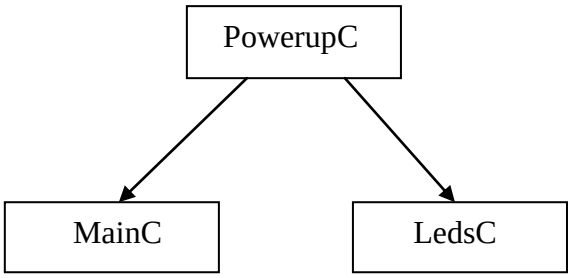
```
configuration PowerupAppC {
 }
implementation {
     components MainC , LedsC , PowerupC;

     MainC.Boot -> PowerupC.Boot;
     PowerupC.Leds -> LedsC.Leds;
}
```

Kod 11. PowerupAppC konfigürasyonu

PowerupAppC konfigürasyonu Powerup programının üç bileşenden oluştuğunu gösteriyor. Bu bileşenler MainC (sistemin başlatılması), LedsC (led kontrolü), PowerupC (Powerup programının modülü) bileşenleridir. PowerupAppC konfigürasyonu, kullanılan bileşenlerin sağladığı (provides) ve kullandığı (uses) arayüzlerin birbirleriyle nasıl bağlanacağını tanımlar. MainC bileşeni sensör düğümün başlatılması işlerini bitirdikten sonra Boot arayüzünün booted olayını çağıran bir sinyal

(signal) gönderir. PowerupC bileşeninin Boot arayüzü MainC bileşeninin Boot arayüzüne bağlandığı için PowerupC bileşeninin Boot arayüzne ait ‘booted’ olayı çağırılır. “booted” olayı Leds arayüzünün led0On komutunu çağırır. PowerupC bileşeninin Leds arayüzü LedsC bileşeninin Leds arayüzüne bağlandığı için 0 numaralı led yanar. 0 numaralı led yandıktan sonra sensör düğümü uyku moduna geçer ve Powerup programı görevini icra etmiş olur. Powerup programının bileşenler arası bağlama(wiring) şeması Şekil 4’de gösterilmektedir.



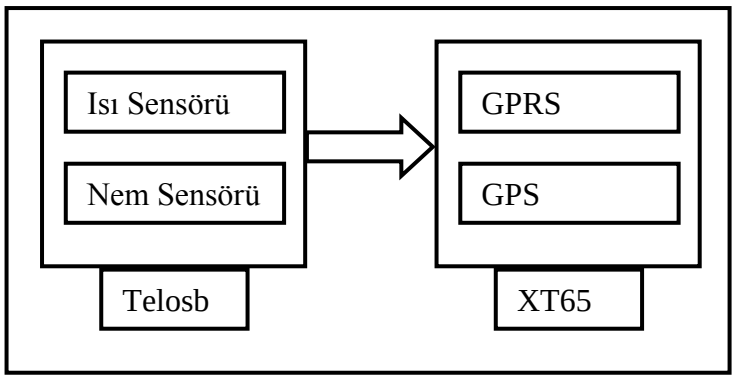
Şekil 4. Powerup bileşeni bağlantı şeması

3. MOBİL SENSÖR CİHAZLARI KULLANILARAK ISI VE NEM DEĞERLERİNİN ÖLÇÜMÜ

Bu bölümde, ısı ve nem değerlerinin ölçümünü gerçekleştiren sensör düğümünü meydana getiren donanımların özellikleri, bu donanımlarda çalışan yazılımların önemli kod blokları ve sistemin genel mimarisi anlatılacaktır.

3.1. Kullanılan Donanımlar ve Özellikleri

Isı ve nem değerlerinin ölçümü için geliştirilen sensör düğümü, Telosb sensörü ve Cinterion XT65 cihazından meydana gelmektedir. Cinterion XT65 cihazı tezin bundan sonraki bölümlerinde XT65 kablosuz ajanı olarak adlandırılacaktır. Sensör düğümü mimarisi Şekil 5'de gösterilmektedir.



Şekil 5. Sensör Düğümü

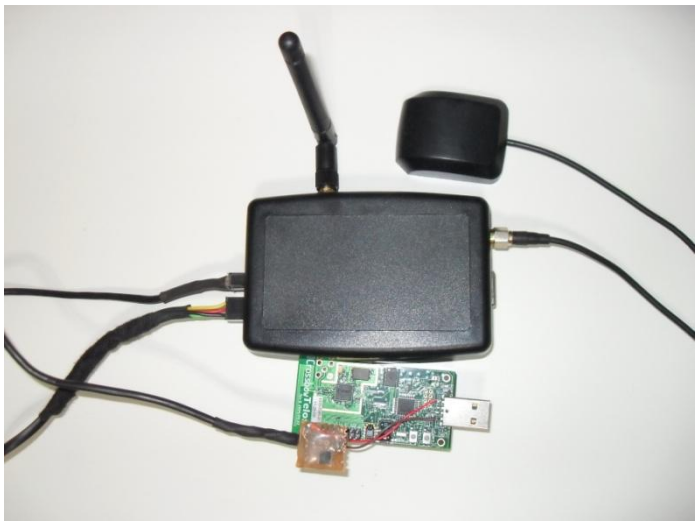
Şekil 5'de gösterildiği üzere Telosb sensörü üzerindeki ısı ve nem sensörleri kullanılarak ısı ve nem değerleri Telosb üzerindeki genişleme (expansion) portu üzerinden XT65 kablosuz ajanına gönderilir. XT65 kablosuz ajanı, üzerinde bulunan GPS alıcısı ile sensör düğümünün konum bilgisini ve konum bilgisinin alındığı GPS zamanını elde eder. Elde edilen konum ve zaman bilgileri Telosb sensöründen

gönderilen ısı ve nem değerleri ile Şekil 6'da gösterilen paket veri yapısına göre birleştirilir.

Tarih	Zaman	Enlem	Boylam	Isı	Nem

Şekil 6. Paket Veri Yapısı

Veri paketi XT65 kablosuz ajanı üzerindeki GPRS modülü sayesinde internet ağı üzerinden sunucu bilgisayara gönderilir. Şekil 7'de Telosb ile XT65 kablosuz ajanın genişleme portu üzerinden bağlantılı hali gösterilmektedir.

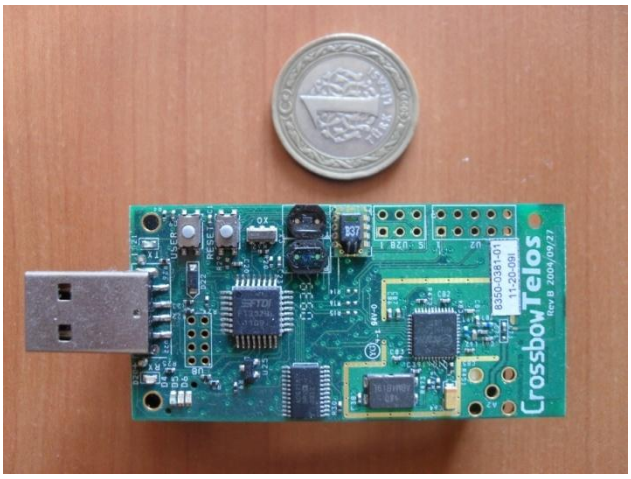


Şekil 7. Telosb ve XT65 bağlantısı

Sistemin tasarımında, sensör düğümleri motorlu araçlar üzerinde bulunması ve enerjilerini araçlar üzerinden sağlayacakları sebebiyle bu tez çalışmasında kablosuz sensörlerin enerji kısıtlamalarından ve enerji kısıtlamalarına bağlı sorunlardan bahsedilmemektedir.

3.1.1. TelosB Sensörü ve Özellikleri

Tez çalışmasında amaçlanan ısı ve nem değerlerinin ölçümü için TelosB sensörü kullanılmaktadır. Berkeley Üniversitesi tarafından üretilen TelosB sensörü Şekil 8'de gösterilmektedir.



Şekil 8. Telosb sensörü

TelosB sensörün donanımsal özellikleri Tablo 2'de gösterilmektedir.

Tablo 2. TelosB Sensör Düğümünün Özellikleri [6]

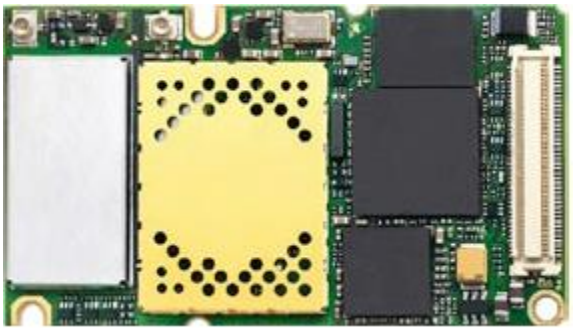
Mikro Denetleyici	8 MHz TI MSP430 16 bit
ROM	48 Kb
RAM	10 Kb
Flash Bellek	1 MB
RF Frekans Bandı	2400 – 2483.5 MHz
RF Veri Gönderim Hızı	250 kbps
RF Standartı	IEEE 802.15.4
RF Kapsama Alanı(Açık Alan)	75 - 100m
RF Kapsama Alanı(Kapalı Alan)	20 - 30m
Enerji Tüketimi	Aktif: 1.8 mA, Uyku: 5.1 µA
Sensörler	Isı, Nem, Işık
Boyutlar(mm)	65 x 31 x 6

Telosb sensör üzerinde RF alıcı-verici bulunmaktadır ve bu alıcı-verici sayesinde Telosb sensörler maksimum 100m çapındaki alan içerisinde birbirleriyle haberleşebilmektedir. Ancak sistem tasarımı gereğince sensör düğümleri arasındaki mesafeler 100 metreden daha fazla olacağı için sensör düğümlerin merkezle arasındaki haberleşme XT65 cihazı üzerinde bulunan GPRS modülü vasıtasıyla GSM altyapısı üzerinden gerçekleştirilmektedir.

3.1.2. XT65 Kablosuz Ajanı

XT65 kablosuz ajanı Telosb'den gönderilen ısı,nem değerlerini ve sensör düğümün bulunduğu GPS konum,zaman bilgilerini sunucu bilgisayara göndermektedir. Üzerinde bulunan işlemci j2me programlama dili ile yazılmış java programlarını çalıştırabilmektedir. J2me(Java Micro Edition), Sun Microsystems tarafından mobil cihazlar için geliştirilen java tabanlı bir kütüphanedir. j2me kütüphanesi temel Java sınıflarından ve j2me için java dilinde yazılmış mobil cihazlarda çalışabilen sınıflardan oluşmaktadır.

XT65 cihazı j2me programlarının yanında AT komutlarınıda işleyebilmektedir. AT ifadesi ATtention(dikkat) kelimesinin ilk iki harfini temsil etmektedir. Modem cihazlarının programlanmasında ve birbirleriyle haberleşmesinde kullanılan bir protokol komut setidir. [12] Şekil 9'da XT65'in işlemci,hafıza ve diğer elemanlarını taşıyan anakart devresi gösterilmektedir.



Şekil 9. XT65 anakart devresi[20]

Şekil 10'da XT65 kablosuz ajanın koruma kutusu ve GPS,GPRS antenlerinin entegre edilmiş hali gösterilmektedir.

Şekil 10. XT65 kablosuz ajanı

XT65 kablosuz ajanın donanımsal özellikleri Tablo 3'de gösterilmektedir.

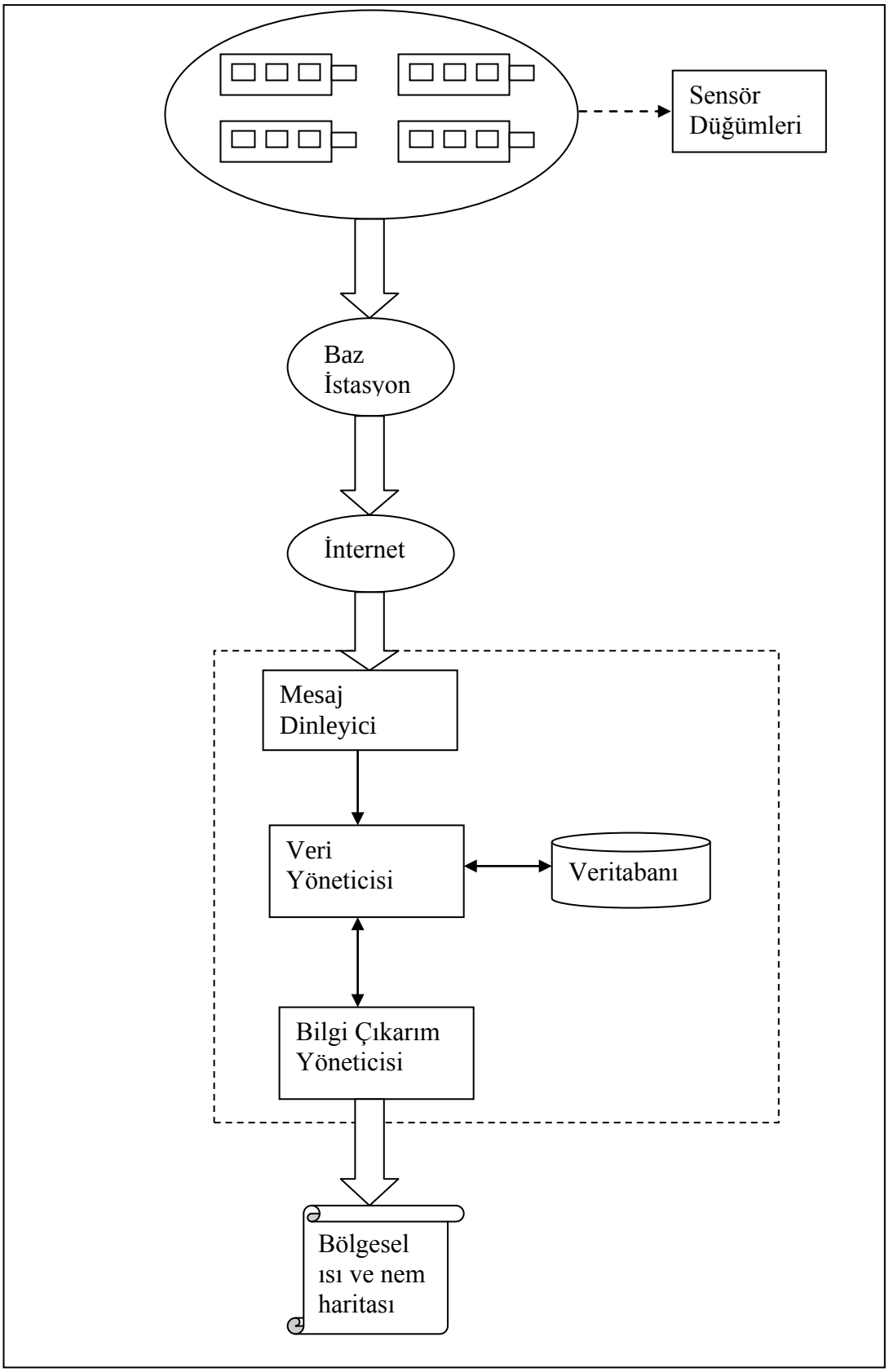
Tablo 3. XT65 Donanımsal Özellikler [20]

CPU	ARM© Core, Blackfin© DSP
RAM	400 Kb
Flash Bellek	1.2 MB
GSM Frekans Bandı	GSM850/900/1800/1900 MHz
GPRS Veri Gönderim Hızı	86 kbps
İnternet servisleri	TCP, UDP, HTTP, FTP, SMTP, POP3

Algılayıcılar	GPS
Boyutlar(mm)	34 x 59 x 3.5

3.1.3. Sistem Mimarisi

Mobil sensör cihazları kullanılarak ısı ve nem değerlerinin ölçümü sisteminde, ölçümü gerçekleyecek sensör düğümleri motorlu araçlar üzerinde bulunacaktır. Sensör düğümlerinin bulunduğu araçlar ölçüm yapılmak istenen coğrafi bölgede istenilen süre boyunca gezdirilerek ihtiyaç duyulan veriler toplanır. Araçlar üzerinden sürekli enerji elde edilebileceği için gerçeklenen sistem tasarımında sensör düğümlerin enerji sorunu bulunmamaktadır. Sensör düğümleri topladıkları verileri sürekli olarak belirli periyotlarda sunucu bilgisayar gönderirler. Verilerin sunucu bilgisayara gönderilmesi GSM alt yapısı üzerinden gerçekleşmektedir. Veriler XT65 kablosuz ajanının GSM modülü üzerinden baz istasyonuna gönderilmektedir. Baz istasyonu kendisine gelen verileri internet ağı üzerinden sunucu bilgisayara göndermektedir. Sunucu bilgisayar kendisine gelen verileri alır ve veritabanına kaydeder. Veritabanında yeterli veri oluştuğunda bu veriler bilgi çıkarımı ve kümeleme algoritmaları kullanılarak ölçüm yapılan bölgelerde zamana bağlı ısı ve nem harita oluşturulmaktadır. Gerçeklenen sistem mimarisi Şekil 11'de gösterilmektedir.



Şekil 11. Sistem mimarisi

Sensör Düğümleri: Araçlar üzerinde konumlanırlar. Belirlenen periyotlarda ortamdan ısı, nem, konum ve zaman bilgilerini ölçerler. Ölçülen verileri baz istasyonu üzerinden sunucu bilgisayara gönderirler.

Mesaj Dinleyici: İnternet ağı üzerinden sensör ajanları tarafından gönderilen veri paketlerini dinler ve veri yöneticisine iletir.

Veri Yöneticisi: Mesaj Dinleyici modülünden gelen ısı, nem, konum ve zaman bilgilerini veritabanına kaydeder. Isı ve nem haritaları çıkarılması için bilgi çıkarım yöneticisine gelen sorguları veritabanı üzerinde işletir ve sorgu sonuçlarını bilgi çıkarım yöneticisine gönderir.

Bilgi Çıkarım Yöneticisi: Veritabanından elde edilen verileri bilgi çıkarım ve k-orta(k-means) kümeleme algoritmaları ile işleyerek, zamana bağlı ısı ve nem haritaları oluşturmaktadır.

3.2. XT65App Uygulaması

XT65App uygulaması j2me sınıfları kullanılarak java programlama dilinde geliştirildi. XT65App uygulaması XT65 cihazı üzerinde bulunun GPS modülünden sensör ajanının anlık konum bilgisini ve Telosb sensörünün gönderdiği ısı ve nem değerlerini okur. Okuduğu değerleri konfigürasyon dosyasında belirtilen ip adresinin yine dosyada belirtilen port numarasına GPRS bağlantısı üzerinden göndermektedir. XT65App uygulaması verileri okuma ve gönderme işlemlerini konfigürasyon dosyasında belirtilen süre periyodunda tekrarlamaktadır.

XT65App uygulaması üç sınıftan oluşmaktadır.

- Connection sınıfı: Verileri sunucu uygulamasına gönderen fonksiyonları tanımlar

- ReadSensor sınıfı: Telosb sensörü tarafından gönderilen verileri okuyan fonksiyonları tanımlar.
- XT65App sınıfı: XT65App uygulamasının temel(main) sınıfıdır. Uygulamanın yönetimini içeren fonksiyonları tanımlar.

3.2.1. Connection Sınıfı

Connection sınıfı Telosb'den gönderilen ölçüm değerlerinin XT65 kablosuz ajanı üzerinden baz istasyonuna gönderilmesi için geliştirildi. Connection sınıfı "sendData" fonksiyonu ile verileri sunucu uygulamaya gönderir. "sendData fonksiyonu" Kod 12'de gösterilmektedir.

```
public void sendData(String ip, String port, String data)
{
 try
 {
  client = (SocketConnection)Connector.open ("socket://"
+               ip + ":" + port+ ";" + connProfile);
  OutputStream os = client.openOutputStream();
  os.write(data.getBytes());
  os.flush();
  os.close();
  client.close();
 }
 catch (IOException e)
 {
     System.out.println(e);
 }
}
```

Kod 12. Connection sınıfının sendData fonksiyonu

"sendData" fonksiyonunun parametreleri sırasıyla verinin gönderileceği sunucunun ip adresi, port numarası ve gönderilecek veridir. "sendData" fonksiyonu içerisinde "Connector.open" fonksiyonu ile sunucu bilgisayara bir soket bağlantısı açılır. Açılan soketin "openOutputStream" fonksiyonu ile soket üzerinde yazma akımı(output stream)

açılır. Açılan yazma akımının "write" fonksiyonu ile sunucuya parametre olarak iletilen veri gönderilir. Veri gönderme işlemi tamamlandıktan sonra açılan yazma akımı ve soket kapatılır.

3.2.2. ReadSensor Sınıfı

ReadSensor sınıfı Telosb'den gönderilen ölçüm değerlerinin XT65 kablosuz ajanı tarafından okunması için geliştirildi. ReadSensor sınıfı "readSensor" fonksiyonu ile Telosb sensörü tarafından gönderilen 2 baytlık ısı ve nem değerlerini okur ve çağırıldığı fonksiyona string tipinde değer döndürür. "readSensor" fonkiyonu Kod 13'de gösterilmektedir.

```
public String readSensor()
{
 String str;
 str = "";
 int temp = 0;
 int hum=0;
 try
 {
  CommConnection cc = (CommConnection) Connector.open
("comm:com0; baudrate=115200", Connector.READ_WRITE,
true);
  InputStream is = cc.openInputStream();
  int counter = 0;
  do
  {
     int a = is.read();
     char b = (char)a;
     if(counter == 0)
     {
       temp = a;
     }
     else if(counter == 1)
     {
       hum= a;
     }
      counter++;
```

```
   }
   while(counter<2);
   is.close();
   cc.close();
   }
   catch(Exception e)
   {
     System.err.println(e);
   }
   str = temp + "," + hum;
   return str;
}
```

Kod 13. ReadSensor sınıfının readSensor fonksiyonu

"readSensor" fonksiyonu "Connector.open" fonksiyonu ile "com0" portuna bir soket açar. Açılan soket üzerinde "openInputStream" fonksiyonu ile okuma akımı(input stream) açılır. Açılan okuma akımının "read" fonksiyonu ile Telosb sensöründen gönderilen 2 baytlık ısı ve nem değerleri okunur. Telosb sensöründen gönderilen ilk bayt ısı değerini, ikinci bayt ise nem değerini temsil etmektedir. "read" fonksiyonu çağırıldığında Telosb sensörden ölçüm değerleri henüz gönderilmemiş ise XT65 kablosuz ajanının ölçümleri ile Telosb sensörün ölçümleri arasında senkronizyon sağlanması açısından XT65App uygulaması bu noktada bloklanır. Telosb sensör ölçüm değerlerini gönderdikten sonra uygulamaya kaldığı yerden çalışmaya devam eder. Okunan ölçüm değerleri "Isı değeri, Nem değeri" formatına getirilerek çağırılan fonksiyona geri döndürülür.

3.2.3. XT65App Sınıfı

XT65App sınıfı XT65 kablosuz ajanında gerçekleştirilen işlermlerin kontrolü için geliştirildi. XT65App sınıfı XT65App uygulamasının "main" sınıfıdır ve uygulama başlatıldığında yaratılan ve çağırılan ilk sınıftır. XT65App sınıfı "Config.txt" isimli konfigürasyon dosyasından sunucu bilgisayarın ip adresini, port numarasını ve verilerin okuma-gönderilme periyodunu belirleyen değeri okur. Şekil 12'de "Config.txt" dosyasının içeriği gösterilmektedir.

```
212.168.2.1
34123
30
```

Şekil 12. Config.txt dosyası

Şekil 12'de gösterilen "Config.txt" dosyasının ilk satırında sunucu bilgisayarın ip adresi, ikinci satırında sunucu bilgisayarın dinlediği port numarası, üçüncü satırında ölçüm periyodunu belirleyen değer tanımlanmaktadır. Ölçüm periyodunu belirleyen değer saniye cinsinden tanımlanmaktadır.

XT65App uygulaması başlatıldığında XT65App sınıfının "startApp" fonksiyonu çağırılır. "startApp" fonksiyonu Kod 14'de gösterilmektedir.

```
protected void startApp() throws
MIDletStateChangeException
{
  System.out.println("Application starting...");
  try
  {
    ATCmd = new ATCommand(false);
    sendAT("AT^SGPSS=1,0");
    Thread.sleep(2000);
    while(true)
    {
      try
      {
       //Zaman ve GPS değerlerini oku
       String gpsStr =   ATCmd.send("AT^SGPSR=0" + "\r");
       String gpsStr1 = gpsStr.substring(21, 51);
       String gpsStr2 = gpsStr.substring(53, 64);
       //Isı ve Nem değerlerini oku
       String sensorStr = Sensor.readSensor();
       //Verileri birleştir
     String wholeData = gpsStr1 + gpsStr2 +","+ sensorStr;
     System.out.println(wholeData );
     //Verileri gönder
     DataConnection.sendData(dstIp, dstPort, wholeData);
          Thread.sleep(timePeriod);
      }
    }
  catch (ATCommandFailedException e)
  {
```

```
        System.out.println(e);
        destroyApp(true);
    }
    catch(Exception e)
    {
       e.printStackTrace();
       System.out.println(e);
    }
    destroyApp(true);
}
```

Kod 14. XT65 sınıfının startApp fonksiyonu

"startApp" fonksiyonu içerisinde öncelikle "AT^SGPSS=1,0" at komutu gönderilirek XT cihazının GPS modülü aktif konuma getirilir. GPS modülü aktif edildikten sonra döngü içerisinde "AT^SGPSR=0" at komutu ile güncel gps konum ve zaman bilgisi okunur. Okunan gps bilgileri Sensor nesnesinin "readSensor" fonksiyonundan döndürülen ısı ve nem bilgileri ile "tarih,zaman,konum bilgisi,ısı,nem" formatına getirilir. Uygun formata getirilen veri DataConnection nesnesinin "sendData" fonksiyonu ile sunucuya gönderilir. Döngü içerisinde yapılan tüm işlemler "timePeriod" değişkeninde belirtilen periyot boyunca tekrarlanır.

3.3. TelosbApp Uygulaması

TelosbApp uygulaması periyodik aralıklarla ısı ve nem değerlerini ölçüp, "expansion" port üzerinden XT65 kablosuz ajanına göndermektedir. TelosbApp uygulaması TelosbAppC konfigürasyon dosyası, TelosbApp modül dosyası ve MsgStruct.h başlık dosyasından oluşmaktadır.

3.3.1. TelosbAppC Konfigürasyon Dosyası

TelosbAppC konfigürasyon dosyası, Telosb sensörün istenilen ölçümleri yapabilmesi için ihtiyaç duyulan bileşenlerin uygulamaya dahil edilebilmesi için oluşturuldu. TelosbAppC konfigürasyon dosyası TelosbApp bileşeninin kullanacağı bileşenleri tanımlamaktadır. Konfigürasyon dosyası Kod 15'de gösterilmektedir.

```
configuration TelosbAppC
{
}
 implementation
 {
  components MainC,LedsC;
  components new TimerMilliC() as SendTimer;
  components new SensirionSht11C();
  components TelosbApp as App;
  components PlatformSerialC as Serial;
  App.UartSend -> Serial;
  App.Control -> Serial;
  App.Boot -> MainC.Boot;
  App.SendTimer -> SendTimer;
  App.ReadTemp ->SensirionSht11C.Temperature;
  App.ReadHum ->SensirionSht11C.Humidity;
  App.Leds -> LedsC;
}
```

Kod 15. TelosbAppC konfigürasyon dosyası

Konfigürasyon dosyasında, TelosbApp bileşeninin ısı ve nem değerlerini okuması için SensirionSht11C bileşeni tanımlanmaktadır. XT65 kablosuz ajanına verileri göndermek için PlatformSerialC bileşeni ve tanımlanan görevleri belirlen periyotta gerçeklemesi için Timer bileşenleri tanımlanmaktadır. Tanımlanan bileşenler TelosbApp bileşenin ilgili fonksiyonlarına bağlanmaktadır(wired).

3.3.2. TelosbApp Modül Dosyası

TelosbApp modül dosyası, Telosb sensörün istenilen ölçümleri gerçekleyebilmesi için geliştirilen TelosbApp bileşeni için oluşturuldu. TelosbApp bileşeni çalışmaya başladığında ilk önce TelosBApp modül dosyasında tanımlanan Boot bileşeninin "booted" fonksiyonu çağırılır. "booted" fonksiyonu uygulama içerisinde kullanılan değişkenleri başlangıç değerlerine eşitler, Timer bileşeninin startPeriodic fonksiyonu ile belirlenen çalışma periyodunu tanımlar. "booted" fonksiyonu Kod 16'da gösterilmektedir.

```
event void Boot.booted()
{
    tempreading = 0;
    humreading = 0;
    call Control.start();
    call SendTimer.startPeriodic(5000);
}
```

Kod 16. Booted fonksiyonu

"Booted" fonksiyonu icra edildikten sonra Timer bileşeninin sayacı işlemeye başlar. Belirlenen periyotlarda Timer bileşeninin "fired" fonksiyonu çağırılır. "fired" fonksiyonu Telosb sensör üzerinde bulunan zamanlayıcı (timer) sayacının programlanan periyotlar boyunca oluşturduğu kesmeler (interrupt) tarafından çağırılır. Telosb sensör tarafından gerçeklenen ölçümlerin, XT65 kablosuz ajanına gönderilmesi için XT65'in ölçüm periyotlari ile senkronize olması gerekmektedir. Senkronizasyonun sağlanabilmesi için ölçümün düzenli aralılarla gerçeklenmesi gerekmektedir. Düzenli aralıklarla ölçümün yapılabilmesi için ölçüm işlemleri zamanlayıcı tarafından çağırılan "fired" fonksiyonunda gerçeklenmiştir. "fired" fonksiyonunda öncelikle tempreading ve humreading bayrakları kontrol edilerek güncel ısı ve nem değerlerinin okunma durumu kontrol edilir. Eğer güncel değerler okunmuş ise PlatformSerialC bileşeninin "send" fonksiyonu ile gönderme işlemi gerçekleşir."fired" fonksiyonu Kod 17'de gösterilmektedir.

```
event void SendTimer.fired()
{
    if (tempreading == 1 && humreading == 1 )
    {
      call UartSend.send((char *)&dataMsg,
sizeof(MsgStruct));
      humreading = 0;
      tempreading = 0;
     }
    call ReadTemp.read();
    call ReadHum.read();

}
```

Kod 17. Fired fonksiyonu

Güncel değerler okunmamış ise gönderme işlemi yapılmaz ve Temperature ve Humidity bileşenlerinin "read" fonksiyonları çağırılarak ısı ve nem değerleri okunur. Isı değerinin okuma işlemi tamamlandığında ReadTemp bileşeninin "readDone" fonksiyonu ve nem değerinin okuma işlemi tamamlandığında ReadHum bileşeninin "readDone" fonksiyonu çağırılır. "readDone" fonksiyonlarında parametre olarak iletilen ısı ve nem değerleri ölçüme dayalı elektriksel voltaj değerleri olduğu için istenen santigrat ve yüzdelik değerlere dönüşümleri gerçekleşir. Isı değerinin santigrat karşılığını bulmak için Denklem 2'deki eşitlik kullanılır.

Isı(^{0}C) = (-39.6 + 0.01 * (okunan_ısı_değeri))

Denklem 2. Isı dönüşüm eşitliği[11]

Nem değerinin yüzdelik karşılığını bulmak için Denklem 3 ve Denklem 4'deki eşitlikler kullanılır. Denklem 3'de verilen eşitlik, nem değerinin lineer yüzdelik değerini döndürür. Denklem 4'de verilen eşitlik, ısı değerinin etkisine bağlı olarak gerçek nem yüzdelik değerini döndürür.

Lineer_nem(%) = (-2.0468 + (0.0367 * okunan_nem_değeri) + (0.0000015955 * okunan_nem_değeri * okunan_nem_değeri))

Denklem 3. Lineer nem yüzdelik dönüşüm eşitliği[11]

Gerçek_Nem(%) = ((mTemperature - 25) * (0.01 + 0.00008 * okunan_nem_değeri)) + (lineer_nem_değeri);

Denklem 4. Gerçek nem yüzdelik dönüşüm eşitliği[11]

Denklem 2 ve Denklem 4'den elde edilen değerler "dataMsg" yapısının(struct) ilgili değişkenlerine yazılır. Isı ve nem değerlerinin okunduğu "readDone" fonksiyonları Kod 18'de gösterilmektedir.

```
event void ReadTemp.readDone(error_t result, uint16_t data)
{
    if (result != SUCCESS)
      {
          data = 0xffff;
          call Leds.led1Toggle();

      }
    mTemperature =(int8_t)(-39.6 + 0.01 * data);
    dataMsg.temperature = mTemperature;
    tempreading = 1;

  }

event void ReadHum.readDone(error_t result, uint16_t data) {
    if (result != SUCCESS)
    {
          data = 0xffff;
          call Leds.led2Toggle();
    }
    dataMsg.humidity = (uint8_t)(-2.0468 + (0.0367 * data) +
(0.0000015955 * data *    data));
    dataMsg.humidity = ((mTemperature - 25) * (0.01 +
0.00008 * data) ) +
    dataMsg.humidity;
    humreading = 1;
 }
```

Kod 18. readDone fonksiyonları

Isı ve nem değerlerinin okuma işlemi tamamlandığında tempreading ve humreading bayrakları işaretlenir ve "fired" fonksiyonu çağırıldığında en son okunan ısı ve nem değerleri XT65 kablosuz ajanına gönderilir. Gönderilen ısı ve nem değerlerinin oluşturduğu veri paketi yapısı Şekil 13'de gösterilmektedir.

1 byte	1 byte
Isı değeri	Nem değeri

Şekil 13. Telosb veri paketi yapısı

Şekil 13'de gösterildiği şekilde Telosb sensöründen XT65'e gönderilen veri paketinin ilk iki baytı ısı değerinden, son iki baytı nem değerinden oluşmaktadır.

3.3.3. MsgStruct.h Başlık Dosyası

MsgStruct.h başlık dosyası Telosb sensöründen XT65 kablosuz ajanına gönderilen veri paketini tanımlamak için oluşturuldu. XT65App uygulaması ile TelosbApp uygulaması arasında gerçeklenen veri alış-verişinde kullanılan veri paketi yapısı MsgStruct.h başlık dosyasında tanımlanmaktadır. MsgStruct.h başlık dosyası Kod 19'da gösterilmektedir.

```
typedef nx_struct MessageStruct{
  nx_int8_t temperature;
  nx_uint8_t humidity;
} MsgStruct;
```

Kod 19. MsgStruct.h başlık dosyası

MsgStruct yapısında ısı değeri, 8 bitlik işaretli tam sayı tipinde olan temperature değişkeninde saklanır. Nem değeri 8 bitlik işaretsiz tam sayı tipinde olan humidity değişkeninde saklanır.

4. BİLGİ ÇIKARIMI VE DEĞERLENDİRME

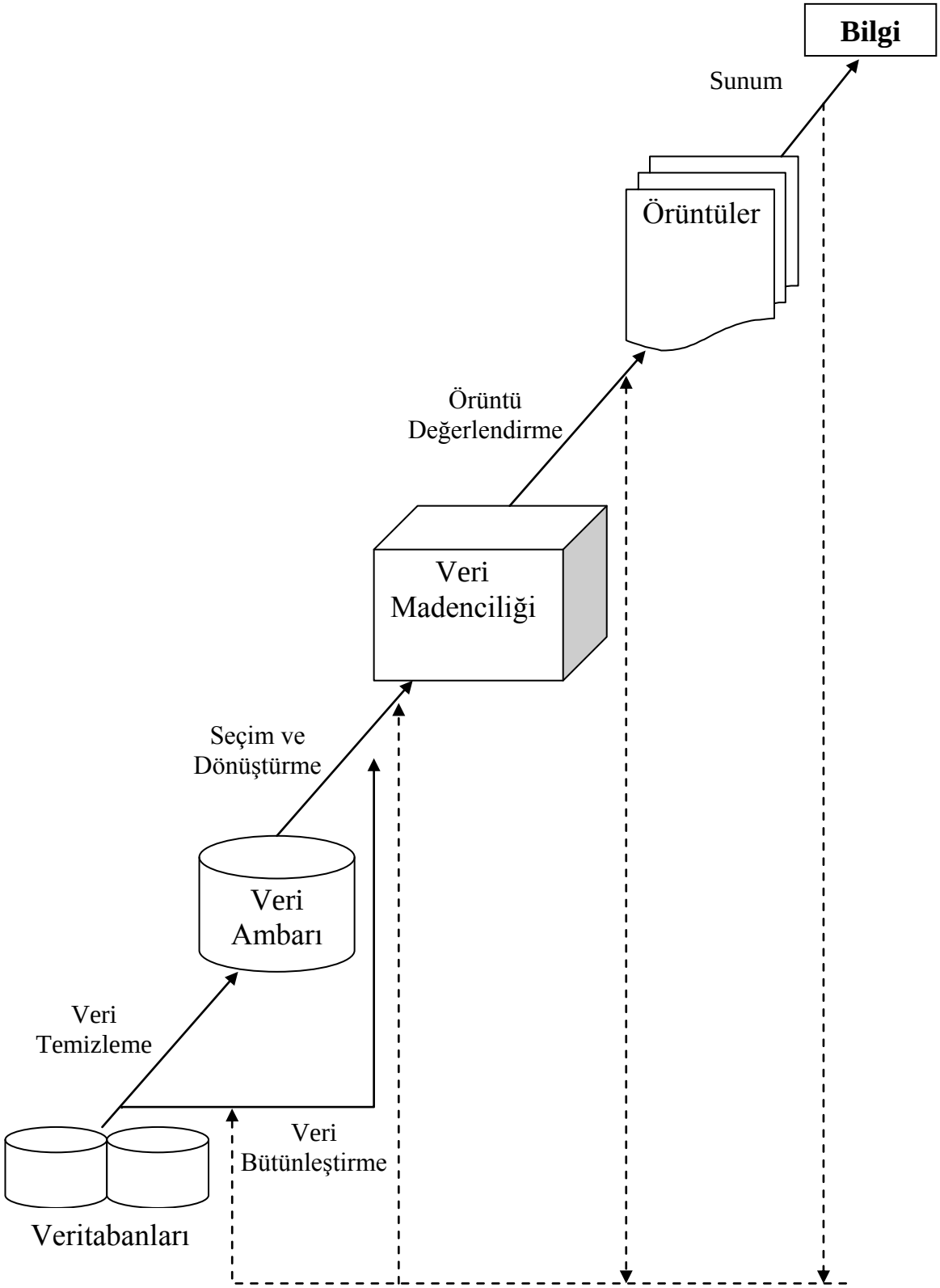
Farklı mobil sensör cihazları kullanılarak ısı ve nem değerlerinin ölçümü ve ölçülen değerlerin veritabanına kaydedilmesi büyük bir veri yığını oluşturmaktadır. Ancak bu veri yığını bölgesel ısı ve nem haritalarının oluşturulması için yeterli değildir. Çünkü veritabanına kaydedilen veriler farklı bölgelerden ve farklı zamanlardan rastgele kaydedilebilirler. İstenen ısı ve nem haritalarının oluşturulabilmesi için bu veri yığınındaki veriler bilgi çıkarım ve veri madenciliğinde kullanılan kümeleme algoritmalarıyla değerlendirilmedirler.

4.1. Bilgi Çıkarımı ve Veri Madenciliği

Veritabanlarında bilgi çıkarımı terimini çok büyük hacimdeki veri yığınları içerisinden yararlı bilgileri keşfetme ve çıkarma sürecinin tamamı olarak yorumlayabiliriz. Veri madenciliği ise bu sürecin önemli bir adımını oluşturmaktadır. [14]

Veri madenciliği, büyük miktardaki veri yığınları içerisindeki veriler arası ilişkilerin, istatistiklerin, değişimlerin otomatik veya yarı otomatik sistemlerle analiz edilerek önemli olan bilgilerin, veri desenlerinin, modellerinin veya biçimlerinin elde edilmesi tekniğidir. Veri madenciliği genel olarak, veri kümeleri arasındaki ilişkilerin, benzerliklerin anlamlı bilgilere dönüştürülmesini sağlayan belirli analiz algortimaları ve yazılım tekniklerinin geliştirilmesini kapsar. [16] Veri yığınları içerisinde bulunan ham veriler tek başlarına anlamlı bir bilgi oluşturmayabilir. Ancak bilgisayar teknolojilerinden faydalanarak, ham veriler üzerinde gerçekleştirilen veri madenciliği ve bilgi çıkarımı analizleriyle yeni ve anlamlı bilgiler keşfedilebilmektedir. [15]

Şekil 14'de bilgi keşfi sürecinin adımları ve veri madenciliğinin bilgi keşfindeki yeri gösterilmiştir. Gerçekte veri madenciliği bilgi keşfi sürecinin bir parçasını oluşturmaktadır ancak pratikte veri madenciliği ve bilgi keşfi benzer süreçler olarak yorumlanmaktadır.



Şekil 14. Sistem Mimarisi[13]

Şekil 14'de gösterildiği üzere bilgi keşfi süreci, verilerin toplanması, veri temizleme, bütünleştirme, seçme ve dönüştürme, veri madenciliği işlemleri, elde edilen örüntülerin değerlendirilmesi ve bilginin sunumu adımlarından oluşmaktadır. Verilerin toplanması temizlenmesi ve bütünleştirilmesi veri önişleme adımları olarak adlandırılırlar. Kaliteli veriler kaliteli çıktılar üretilmesini sağlarlar. Dolayısıyla veri önişlem adımları bilgi çıkarım sürecinde büyük önem taşımaktadır.

4.1.1. Verilerin Toplanması (Data Collection)

İstenen amaca uygun olarak gerekli olan verilerin ve bu verilerin elde edileceği kaynakların belirlenmesi adımıdır. [14] Veri ön işleme sürecinin ilk aşaması olan veri toplama, güvenilir kaynaktan, verilerin tespit edilip toplanması süreci olarak da tanımlanabilir. [16] Gerçeklenen tez çalışmasında mobil sensör cihazlarından gönderilen verilerin veritabanlarına kaydedilme süreci veri toplama sürecine karşılık gelmektedir.

4.1.2. Veri Temizlenmesi (Data Cleaning)

Aşırı uç değerler, yanlış girilmiş yada bozulmuş (gürültülü) verilerin analiz edilecek veri yığınları arasından çıkarılması olarak tanımlanan bu süreçte veriler arasındaki tutarsız durumlar giderilir. [15] Veri temizleme işlemleri yapılmayan veriler üzerinde bilgi çıkarım işlemlerinin yapılması hedeflenen anlamlı bilgilerin elde edilme olasılığını ve doğruluğunu olumsuz yönde etkilemektedir.

4.1.3. Veri Bütünleştirme (Data Consolidation)

Veri kaynaklarından gelen verilerde kayıplar oluşabilir yada birbirleriyle ilişkili veriler farklı veritabanlarında bulunabilirler. Kayıplar mümkün olduğu ölçüde çözülmeye çalışılır ve tüm veriler bir veritabanında toplanır. [14]

4.1.4. Veri Şeçimi Ve Dönüştürme (Data Selection And Transformation)

Verilerin veri madenciliğine uygun yapıya dönüştürülmesi işlemi olan bu süreçte verilerin seçilen veri madenciliği algortimasına uygun veri yapısına dönüştürülme işlemleri bulunmaktadır. [14] Örneğin bir algoritmanın uygulanmasında değişken değerlerinin evet/hayır olması, başka algoritmada da değişken değerlerinin yüksek/orta/düşük olması kullanılan algoritmanın etkinliğini arttırabilir. [15]

4.1.5. Veri Madenciliği (Data Mining)

Veri madenciliği adımı bilgi çıkarım sürecinin en önemli aşamasıdır. Bu aşamada elde edilmesi planlanan bilgiler üzerinden tasarlanan model veya modeller ve bu modellere dayalı uygun veri madenciliği analiz yöntemleri belirlenir. Hedefler doğrultusunda en iyi sonucu veren model ve analiz yöntemleri bu süreçte değerlendirilir ve belirlenir. [14] Veri önişleme aşamasından sonra, analiz edilmeye uygun hale getirilen veriler üzerinde, belirlenen analiz yöntemleri uygulanarak verilere ait örüntüler elde edilmektedir. [15]

4.1.6. Örüntü değerlendirme (Pattern Evaluation)

Veri madenciliğinde seçilen algoritmanın ürettiği örüntülerin değerlendirilmesi ve ilginç örüntülerin tanımlanması bu aşamada gerçekleşmektedir. Örüntü değerlendirmede, ilginçlik (interestingness) ölçüm yöntemleri kullanılarak bulunan verilerin ne derecede yararlı oldukları tespit edilir. [14]

4.1.7. Bilgi Sunumu (Knowledge Presentation)

Örüntü değerlendirme aşamasının çıktısı olan bilgilerin kullanıcının istediği formatta düzenlenmesi bu aşamada gerçekleştirilir. [16] Kullanıcı ile bilgi çıkarım sistemi arasında arayüz görevi görür. Bilginin düzenlenmesinde farklı görselleştirme ve raporlama araçları kullanılabilir. [14]

4.2. Kümeleme Analizi

Veri madenciliğinde yaygın olarak kullanılan yöntemlerden biri kümeleme analizidir. Kümeleme işlemi veri analizlerinin örüntü oluşturma aşamasında, veri kaynağındaki tüm verilerin kullanılması yerine, benzer özellik gösteren verilerin kullanılması için verileri belirli özelliklere göre kümelere ayırma sürecidir. [17] Kümeleme işlemi aynı karakteriktiğe sahip verilerinsaptanması ve benzerliklerine göre sınıflandırılmasını sağlayan çok değişkenli bir analiz yöntemidir. Ortaya çıkan grupların her birine küme adı verilir. Bir küme aynı küme içndeki diğer veriler ile benzer özellikler gösteren veriler toğluluğudur. Benzer özellikleri göstermeyen diğer veriler de farklı kümelerde bulunmaktadır.

Kümeleme analizi sürecinde birçok farklı kümeleme metodu bulunmaktadır. Bu meteodların kullanılmasında verilerin türü ve uygulamanın amacı önemli bir unsur olmaktadır. En çok kullanılan kümeleme metotları hiyerarşik ve bölümleme metodlarıdır.

4.2.1. Hiyeraşik Metodlar (Hierarchical methods)

Bu süreçte veri kümeleri önceden belirlenmiş parametrelere göre hiyeraşik olarak sınıflandırılırlar**.** [18] Hiyerarşik kümeleme metodları, verileri ağaç yapısı şeklindeki gruplar içerisinde kümeleyerek çalışırlar. Hiyerarşik metodlar, küme sayısını belirten k değerine ihtiyaç duymazlar, ancak ağaç yapısının tamamlandığını belirten eşik değerine ihtiyaç duyarlar. [14]

Hiyeraşik kümeleme metotları, sağlam ve basit olmaması ve az güvenilir olması gibi dezavantajlara sahip olmasına rağmen, yorumlama ve okumada kolaylık sağlamasından dolayı diğer hiyerarşik olmayan kümeleme metotlarına göre daha fazla kullanılmaktadır. [14]

4.2.2. Bölümleme Metotları (Partitioning methods)

Bölümleme metotlarında veriler gruplara bölünerek, her grup belirlenmiş bir kritere göre değerlendirilir. Bölümleme yöntemlerinde, n veritabanındaki veri sayısı ve k bölümleme sonucunda oluşacak küme sayısı olarak kabul edilir. Bölümleme algoritması n adet verinin, k sayıda kümeye bölünmesini gerçekler ($k \leq n$). Her bölüm bir kümeyi temsil etmektedir. Kümeler belirli kriterler doğrultusunda ayrıştırılan verilerden

oluşması sebebiyle aynı kümedeki veriler birbirleryle ortak özellikler taşırken, farklı kümedeki veriler farklı özelliklere sahiptirler. [18] En yaygın kullanılan bölümleme yöntemleri arasında K-means ve KNN yöntemleri bulunmaktadır.

4.2.2.1. KNN (K en yakın komşu) yöntemi

Sınıf belirleme işleminin başarımı, daha önceden sınıflandırılan örneklerin sayısı ile doğru orantılıdır. Ancak, bazı durumlarda tek uzaklıktaki komşuluklar istenen başarımın sağlanmasında yeterli olmazlar. Bir noktanın k komşusuna bakılarak sınıfının belirlenmesi yöntemi k en yakın komşu yöntemi olarak adlandırılır. [19] K en yakın komşu yönteminde, komşuluk dereceleri bulunacak verilerin öğrenme kümesindeki normal davranış verilerine benzerlikleri hesaplanarak, en yakın olduğu hesaplanan k verinin ortalamasıyla, belirlenen eşik değerine göre sınıflandırmaları yapılır. [16] Sınıflandırmadaki hatayı azaltmak için k sayısı büyük seçilebilir yada k sayısı küçük seçilerek yakın komşuların etkileri artırılabilir. [19]

KNN algoritması Denklem 5'de gösterildiği gibi matematiksel olarak ifade edilebilir.

$D(xi) = j$ ($j \in [1,m]$)

$D(x) = D(min (dist (x,x_i)))$ ($i \in [1,n]$) $\forall$ xi için

Denklem 5. KNN matematiksel gösterimi[19]

KNN algoritmasının adımlarını Tablo 5'de gösterildiği şekildedir.

Tablo 4. KNN algoritması adımları[16]

```
Ögrenme kümesi D = {d1, d2, d3, …., dm);
foreach   kümele işlemi yapılacak X verileri (i =1…n), n: X verilerinin sayısı
   if xi verisi beklenen sınır değerleri arasında değilse then
      xi = anormal;
```

```
else
   foreach   D kümesi (j =1…n)
        yj =  yakınlık_hesapla(xi, dj)
   yj değerlerini büyükten küçüğe sırala;
   S  = sırala(yj)
   S kümesinin ilk k tane  elemanı için yakınlık ortalaması hesapla(S) ;
   if yakınlık_ortalaması(xi) > eşik değer then
      xi =normal;
   else
      xi =anormal;
```

Tablo 5'de gösterildiği üzere X test kümesi, D öğrenme kümesi olsun. Öncelikle X kümesindeki her verinin, D kümesindeki verilere olan yakınlıkları hesaplanır ve sıralanır. Sıralanmış yakınlık bilgileri üzerinden en büyük k tane değer alınarak ortalamaları hesaplanır. Son olarak ortalama değerleri, belirlenen eşik degerinden büyük X kümesi verileri normal, küçük olan veriler ise anormal olarak sınıflandırılır. [16]

KNN algoritmasının performansını etkileyen kriterler [16],

- Benzerlik ölçümü için seçilen k komşu sayısı
- K en yakın komşuya olan benzerliklerinde hesaplanan ortalama değerin kıyaslanmasında kullanılan eşik değeri
- Benzerlik ölçümü
- Öğrenme kümesindeki normal davranışların yeterli sayıda olması

4.2.2.2. K-means yöntemi

En iyi bilinen ve en çok kullanılan K-means yöntemi 1967 yılında J.B. MacQueen tarafından geliştirilmiştir. K-means yönteminde k adet küme oluşmaktadır. Her küme içerisinde bulunan verilerin ağırlıklı ortalamları sonucu bir karşılaştırma değeri ortaya

çıkmaktadır. Küme içerisinde bu değere en yakın olan nokta değeri küme merkezi olarak kabul edilmektedir. [14] Kümeleme sonucunda oluşan kümelerin elemanları (intra-cluster) arasındaki benzerlikler, kümeler arası (inter-cluster) elemanları arasındaki benzerlikler göre çok daha fazladır. [18]

K-means algoritmasının genel mantığı, öncelikle k sayıda rastgele nokta belirlenmesidir. Bu noktalar ilk küme merkezlerini temsil etmektedir. Daha sonra gelen her yeni veri bu merkez noktalara en yakın olduğu noktanın kümesine dahil edilir ve eklendiği kümenin ortalaması tekrar hesaplanarak yeni merkez nokta bulunmaktadır. [14] Bu işlemler her yeni veri geldiğinde tekrarlanmaktadır. Tablo 5'de K-means algoritmasının adımları gösterilmektedir.

Tablo 5. K-means algortiması adımları[16]

K tane rastgele küme merkezi belirle
foreach kümele işlemi yapılacak X verileri (i =1…n), n: X verilerinin sayısı
a = x_i
foreach kümelenmiş verilerin bulunduğu C kümeleri (j= 1…k)
b = yakınlık_hesapla(a, merkez_nokta(c_j))
index = j
x_i verisini en küçük b değerinin elde edildiği c_j kümesine ekle
ekle(x_i,c_{index})
foreach kümelenmiş verilerin bulunduğu C kümeleri (j= 1…k)
küme merkezlerini hesapla;
merkez_nokta(c_j) = toplam(d_i)/eleman_sayısı(c_j) ($d_i \in c_j$);

K-means yönteminde hata ölçütü olarak hata-kareleri ölçütü (square-error criterion) kullanılır. Hata-kareleri ölçütü Denklem 6'da gösterilmektedir.

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2$$

Denklem 6. Hata-kare ölçütü[13]

Denklem 6'da ifade edilen denklemde eşitliğin sol tarafındaki E ifadesi veritabanındaki tüm nesnelerin hata-kare değerlerinin toplamına eşittir. p ifadesi hangi kümeye ait

olduğu hesaplanacak yeni verinin değeri, m_i ifadesi ise i. kümenin hesaplanan son merkez noktası olan ortalama değeridir.

K-means algoritmasının zayıf yönlerini aşağıdaki gibi sıralayabiliriz. [14]

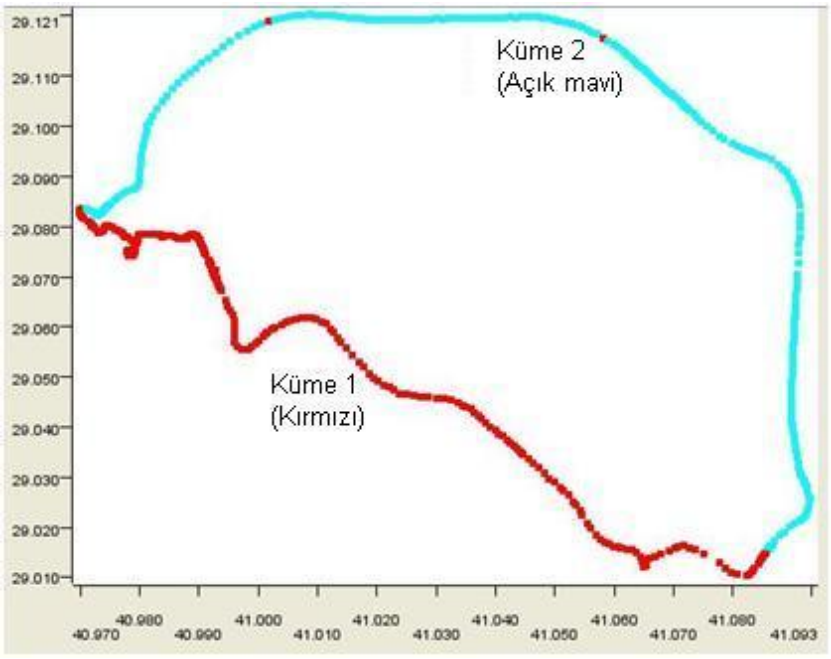
- Algoritmanın başında giriş parametresi olarak bir k sayısına ihtiyaç vardır. Elde edilecek olan sonuçlar k sayısına göre değişkenlik gösterebilmektedir.
- Aşırı gürültü ve istisna veriler algoritmayla hesaplanan ortalama değeri değiştirdiği için k-means algoritması gürültü ve istisnaya karşı aşırı duyarlıdır.
- K-means algoritması sadece sayısal veriler ile kullanılabilmektedir. Kategorik verilerin kümelenmesi için k-means algortiması bir çözüm sunamamaktadır.

4.3. Değerlendirme

Bu tez çalışmasında geliştirilen sensör düğümü, bir araç üzerinde İstanbul ilinde belirli bir güzergah üzerinde dolaştırılarak anlık konum,ısı ve nem değerleri toplanmıştır. Şekil 15'de sensör düğümün verileri topladığı güzergah gösterilmektedir.

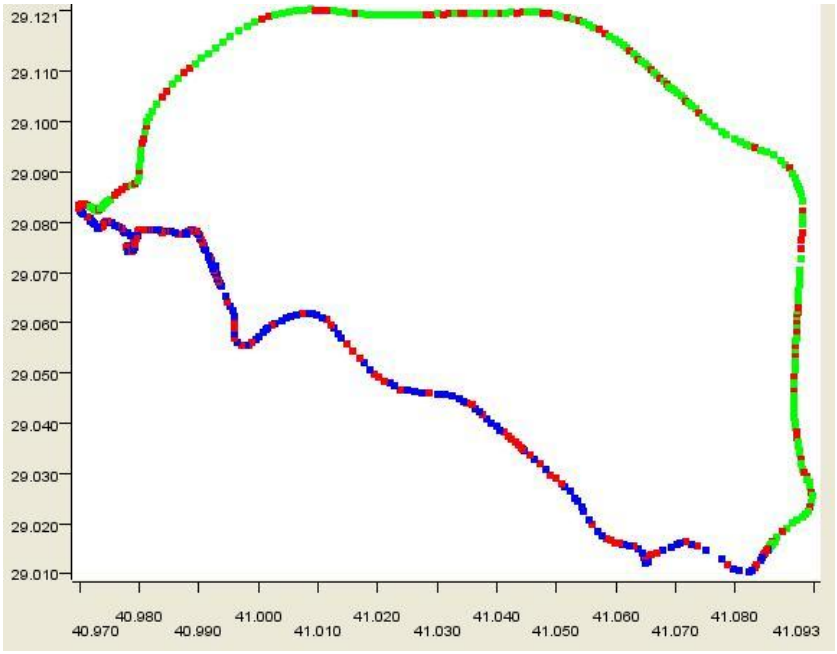
Şekil 15. Veri toplama güzergahı

Şekil 15'de gösterilen güzergah üzerinde 10 sn. aralıklarla toplam 501 adet konum,ısı,nem verileri toplanmıştır. Toplanan 501 adet veri üzerinde K-means algoritması kullanılarak kümeleme analizi yapılmıştır. İlk yapılan analizde k =2 değeri verilerek iki adet küme oluşması sağlanmıştır. K = 2 değeri ile çalıştırılan k-means algoritması Şekil 16'da gösterilen kümeleri oluşturmaktadır.



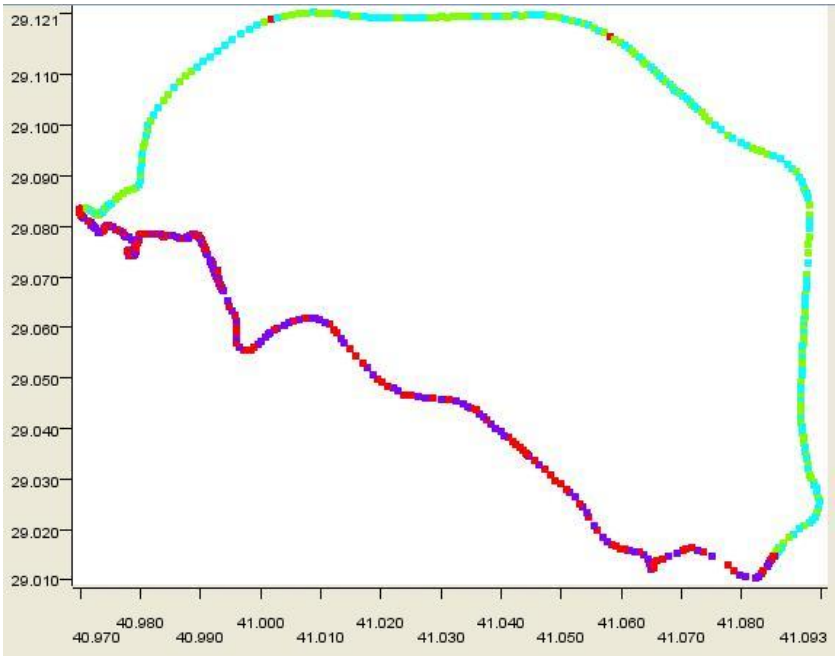
Şekil 16. 2'li kümeleme

Şekil 16'da gösterildiği gibi kırmızı ile gösterilen noktalar birinci kümeye, açık mavi ile gösterilen noktalar ikinci kümeye ait olmaktadır. Şekil 16'da gösterilen noktalar Şekil 15'de gösterilen güzergah üzerine iz düşürüldüğünde kırmızı ile belirtilen noktaların verilerin toplandığı coğrafi bölgenin güney- güneybatı bölgesinde kümelendiği, açık mavi ile belirtilen noktaların bölgenin kuzey-kuzeydoğu kesiminde kümelendiği görülmektedir. Kırmızı bölgenin denize daha yakın olduğu, açık mavi bölgenin iç kesimlere daha yakın olduğu da görülmektedir. K-means algoritması k = 3 ve k = 4 olacak şekilde elde edilen veriler üzerinde tekrar çalıştırıldı ve sırasıyla Şekil 17 ve Şekil 18'deki sonuçlar elde edildi. Şekillerde gösterilen her farklı renk bir kümeyi temsil etmektedir.



Şekil 17. 3'lü kümeleme

Şekil 17'de görüldüğü üzere 3 adet küme oluşturulduğunda mavi renkli noktalar verilerin toplandığı bölgenin güney- güneybatı kesiminde, yeşil noktalar kuzey-kuzeybatı kesiminde kümelenmektedir. Kırmızı ile belirtilen noktalar ise bölgenin hem kuzey hem de güney kesiminde kümelendiği görülmektedir.



Şekil 18. 4'lü kümeleme

Şekil 18'de K-means algoritmasının k = 4 olduğu kümeleme çıktısı görülmektedir. Kırmızı ve mavi noktaların güney-güneybatı bölgesinde kümelendiği, açık mavi ve yeşil noktaların kuzey-kuzeydoğu bölgesinde kümelendiği görülmektedir.

K=2 değeri ile oluşturulan 2 kümeye ait ısı ve nem verilerinin ortalama değerleri Tablo 6'da gösterilmektedir.

Tablo 6. Ortalama değerler

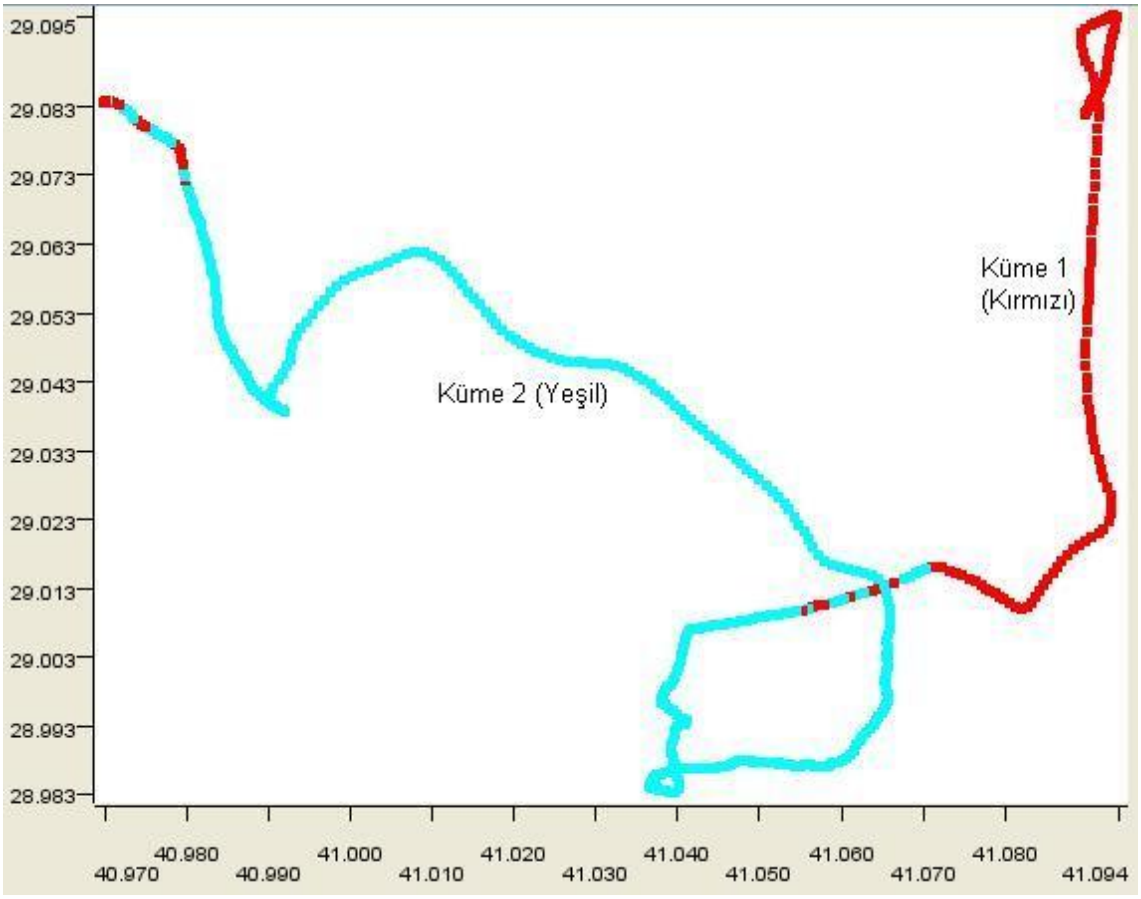
	Küme 1 (Kırmızı)	**Küme 2 (Açık Mavi)**
Ortalama ısı	13.893	14.03
Ortalama nem	52.179	57.235
Toplam örnek sayısı	245	256

Tablo 6'da görüldüğü üzere her iki kümeye ait örnek sayıları birbirine yakın olmakla birlikte ortalama ısı değerleri arasındaki fark 0.137 dir. Ortalama nem değerleri arasındaki fark ise 5,056 dır. Hesaplanan fark değerlerine göre iki kümenin belirlenmesinde nem değerlerinin ısı değerlerine göre daha fazla ayırt edici olduğu görülmektedir. Denize yakın bölgeyi temsil eden küme 2'nin nem değerlerinin yüksek olması, iç kesimleri temsil eden küme 1'in nem değerlerinin düşük olması nem değerinin ayırt etmede önemli rol oynadığını göstermektedir. 2'li, 3'lü ve 4'lü kümeleme analiz sonuçlarından görüldüğü üzere en doğru kümeleme k = 2 değeri ile oluşturulan 2'li kümeleme analiz sonucudur.

Veri toplama çalışmaları daha geniş coğrafi alanı kapsayacak şekilde yeniden uygulanmıştır. Veri toplama çalışması Kavacık, Levent, Taksim, Beşiktaş, Şişli, Kadıköy (Hasanpaşa – Erenköy) bölgelerini kapsamaktadır. Verilerin toplandığı güzergah Şekil 19'da gösterilmektedir.

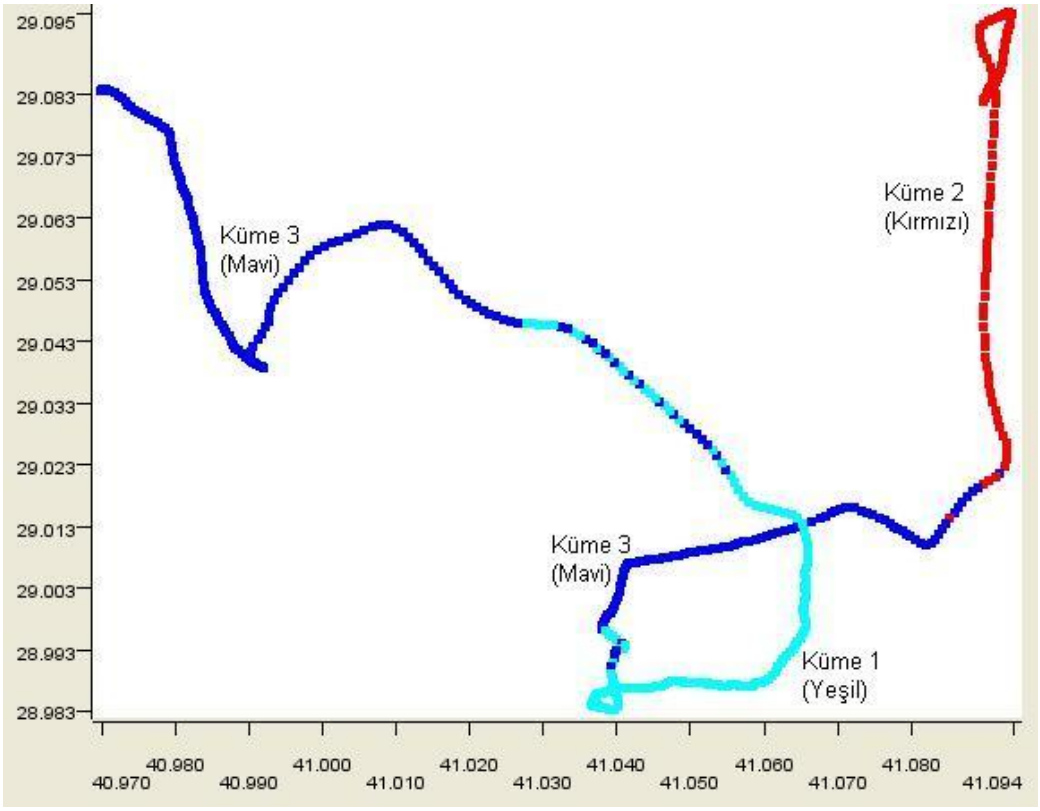
Şekil 19. Veri toplama güzergahı-2

Şekil 19'da gösterilen güzergah üzerinde 5 sn. aralıklarla toplam 543 adet konum, ısı ve nem verileri toplanmıştır. Toplanan 543 adet veri üzerinde K-means algoritması kullanılarak kümeleme analizi yapılmıştır. İlk yapılan analizde k = 2 değeri verilerek iki adet küme oluşması sağlanmıştır. K = 2 değeri ile çalıştırılan k-means algoritması Şekil 20'de gösterilen kümeleri oluşturmaktadır.

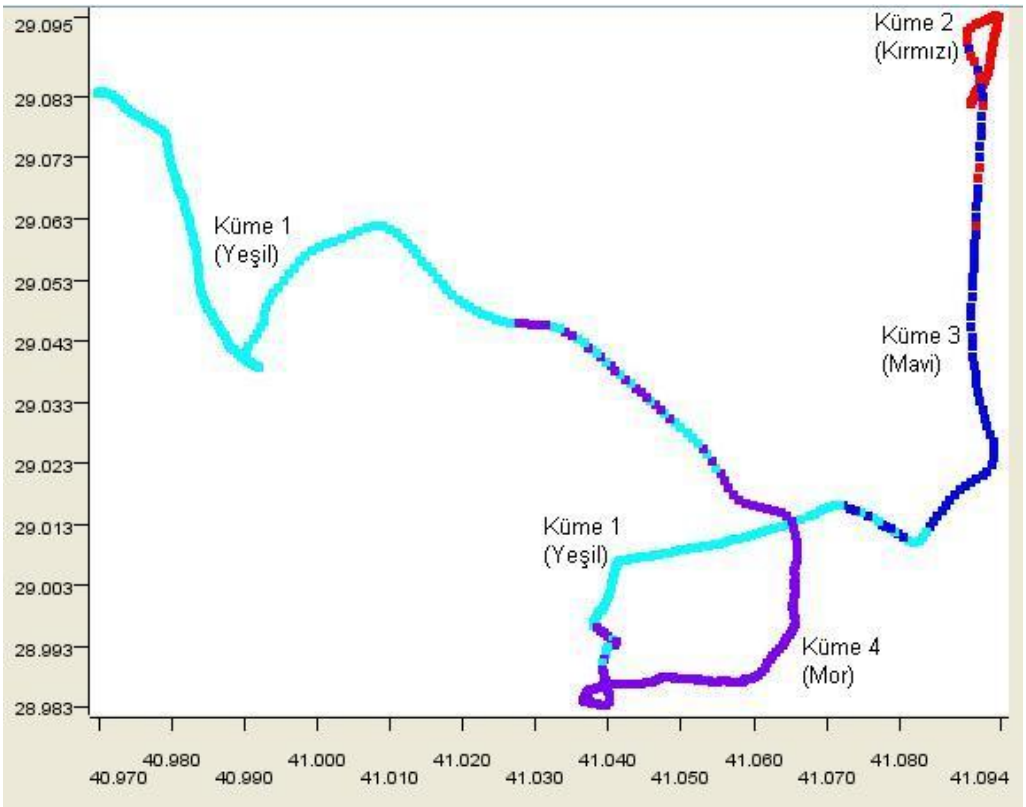


Şekil 20. 2'li kümeleme sonucu

K-means algoritması k = 3 ve k = 4 olacak şekilde elde edilen veriler üzerinde tekrar çalıştırıldı ve sırasıyla Şekil 21 ve Şekil 22'deki sonuçlar elde edildi. Şekillerde gösterilen her farklı renk bir kümeyi temsil etmektedir.



Şekil 21. 3'lü kümeleme sonucu



Şekil 22. 4'lü kümeleme sonucu

Kümeleme işlemi sonucu 3'lü ve 4'lü kümelemede, kümelerin çok net bir şekilde ayrıldığı görülmektedir. Bu nedenle 3'lü ve 4'lü kümeleme üzerinde analiz çalışması yapılmıştır. 3'lü kümele işlemine göre elde edilen ortalama ısı ve nem değerleri aşağıdaki Tablo 7'de görülmektedir. Tablo 7'ye göre ortalama nem değeri en yüksek olan küme kırmızı ile renklendirilmiş olan Küme 2'dir. Şekil 19'da gösterilen harita üzerinde bakıldığı zaman bu bölgenin Kavacık-Levent arası ve Fatih Sultan Mehmet köprüsünün üzeri ve yakını olduğu görülmektedir. Bu bölgenin denize yakın olması nem değerinin yüksek olmasını göstermektedir. Küme 3(Mavi renkli) içerisinde yer alan iki farklı coğrafi bölgenin (Beşiktaş (Barbaros bulvarı) ve Kadıköy (Hasanpaşa - Erenköy)) ısı ve nem değerleri bakımından benzerlik gösterdiği Şekil 21'de görülmektedir.

Tablo 7. Üçlü kümelemenin ortalama değerleri

	Küme 1(Yeşil)	**Küme 2(Kırmızı)**	**Küme 3(Mavi)**
Ortalama ısı	7.56	5.04	6.23
Ortalama nem	64.43	73.74	68.48

K-means algoritması k = 4 olacak şekilde çalıştırıldığında elde edilen sonuçlar Şekil 22'de gösterilmektedir. Tablo 8'de ise 4'lü kümeleme için elde edilmiş olan ortalama ısı ve nem değerleri görülmektedir. 3'lü kümeleme sonucunda elde edilen iki farklı coğrafi bölgenin (Beşiktaş(Barbaros bulvarı) ve Kadıköy(Hasanpaşa-Erenköy)) ısı ve nem değerleri bakımından benzerliği 4'lü kümeleme işleminde de görülmektedir. Tablo 8'de görüldüğü gibi nem değerleri en yüksek olan iki küme Küme-3 ve Küme-4'dür. Bu iki küme ise 3'lü kümelemede de görüldüğü gibi Kavacık-Levent arasını ifade eder. Küme-2 ve Küme-3'ün ortalama ısı ve nem değerlerinin birbirine yakın olmasından dolayı bu iki küme, 3'lü kümeleme işleminde tek bir küme (Şekil 21'deki kırmızı renkli küme) altında toplanmaktadır

Tablo 8. Dörtlü kümelemenin ortalama değerleri

	Küme 1 (Yeşil)	**Küme 2 (Kırmızı)**	**Küme 3 (Mavi)**	**Küme 4 (Mor)**
Ortalama ısı	6.8	5.0	5.39	7.49
Ortalama nem	66.7	74.5	72.0	64.8

3'lü ve 4'lü kümeleme sonuçlarının her ikisinde de Şişli bölgesi ayrı bir küme olarak yer almaktadır. Bu bölge Şekil 21'de yeşil, Şekil 22'de mor renkle gösterilmektedir. Her iki kümeleme işlemi sonucunda Şişli bölgesinin ayrı bir küme olarak gösterilmesi, bu bölgenin diğer bölgelere göre belirgin faklılıklar içerdiğini göstermektedir.

5. SONUÇ

Son yıllarda elektronik ve enerji depolama teknolojilerinde sağlanan ilerlemeler sensör teknolojilerinin gelişmesini hızlandırmada, sensör maliyetlerinin düşmesinde ve buna bağlı olarak kullanım alanlarının artmasında etkili olmuştur. Sensörlerin yetenekleri arttıkça ve maliyetleri düştükçe kullanım alanları da artmaktadır. Önceleri askeri uygulamalarda kullanılan sensörler, giderek çevre izleme, coğrafi haritalar oluşturma gibi akademik alanlarda ve akıllı ev, sağlık sektörü gibi ticari alanlarda da kullanılmaktadır.

Bu tez çalışmasında geliştirilen sensör düğümünün belirlenen bir bölgede dolaştırılmasıyla konum, ısı ve nem değerleri anlık olarak toplanmaktadır. Toplanan veriler GSM altyapısı kullanılarak GPRS protokolü üzerinden sistemde internet adresi tanımlanan sunucu bilgisayara anlık olarak gönderilmektedir. Tez çalışmasında geliştirilen sistem ile elimizde yer alan sensör tiplerine göre ısı ve nem değerleri toplanmıştır. Farklı tipte sensör (CO sensörü, ışık sensörü, CO2 sensörü) kartları kullanılarak farklı çevresel değerlerde algılanabilir ve merkezi sunucuya gönderilebilir. Bu şekilde bölgenin karbon dioksit, karbon monoksit gibi çevresel değerler de anlık olarak toplanabilir. Bölgenin hava kirlilik haritaları da zamana bağlı olarak çıkarılabilir.

Günün hangi saatleri, haftanın hangi günleri, hangi aylar gibi farklı periyotlar için bu değerlere ve haritalara sahip olunabilir.

Geliştirilen sistemle toplanan veriler bölgenin haritaları üzerine işlenmesi sonucu o bölgenin ısı ve nem haritaları ortaya konabilir. Bu şekilde mevsimsel olarak veya aylık olarak belirli alt bölgelerinde ısı ve nem değerlerini elde edebiliriz. Benzer çevresel özelliklere sahip farklı coğrafi bölgeler tespit edilebilir. Gelecek yıllar için ısı ve nem değeri tahminleri yapılabilmesi için bu toplanan veriler kullanılabilir.

Tez çalışmasında sunulan sistem ile ortaya çıkarılacak ısı ve nem haritaları ve elde edilen veriler, bölgenin iklim politikalarının belirlenmesinde önemli rol oynayabilir.

KAYNAKLAR

1. Erboral, S., Kablosuz Duyarga Ağlarında Veri Birleştirilmesi Ve Değerlendirilmesi, İstanbul Teknik Üniversitesi, Yüksek Lisans Tezi, İstanbul, Eylül 2008

2. Akyildiz, Ian F., Su W., Sankarasubramaniam, Y., Cayirci, E., A Survey On Sensor Networks, IEEE Communication Magazine, Volume: 40, Issue: 8, 2002

3. Aslan, Ali U., The Effects Of Hierarchy On Mobile Wireless Sensor Network Coverage, Middle East Technical University, Yüksek Lisans Tezi, Nisan 2006

4. Levis, P., TinyOS Programming, 2006

5. Kalaycı, T. E., Kablosuz Sensör Ağlar Ve Uygulamaları, Akademik Bilişim'09 - XI. Akademik Bilişim Konferansı Bildirileri, Harran Üniversitesi, Şanlıurfa, Şubat 2009

6. http://www.xbow.com , Crossbow Technology, Inc. TelosB Datasheet, (Erişim tarihi: 01.12.2010)

7. Levis, P., Gay, D., TinyOS Programming, Cambridge University Press, ISBN-13 978-0-511-50730-4, 2009

8. Gay, D., Levis, P., Culler, D., Brewer, E., NesC 1.1 Language Reference Manuel

9. Karlof, C., Sastry, N., Wagner, D., TinySec: a Link Layer Security Architecture for Wireless Sensor Networks, http://www.cs.berkeley.edu/~daw/papers/tinysec-sensys04.pdf, 2004, (Erişim tarihi: 10.10.2010)

10. Zhu, S., Setia, S., Jajodia, S., LEAP: Efficient Security Mechanisms for Large-scale Distributed Sensor Networks, http://www.cs.ucsb.edu/~ravenben/papers/sensors/LEAP-ccs03.pdf, 2003, (Erişim tarihi: 09.10.2010)

11. http://www.sensirion.com , Sensirion Company, Datasheet SHT1x., 2010, (Erişim tarihi: 01.12.2010)

12. http://www.developershome.com/sms/atCommandsIntro.asp, (Erişim tarihi: 21.11.2010)

13. Han, J., Kamber, M., Data Mining Concepts and Techniques, Academic Press, ISBN: 1-55860- 489-8, 2001

14. Erdoğan, Ş. Z., Veri Madenciliği ve Veri Madenciliğinde Kullanılan K-means Algoritmasının Öğrenci Veri Tabanında Uygulanması, İstanbul Üniversitesi, Yüksek Lisans Tezi, İstanbul, 2004

15. Esen, M. F., Veritabanlarından Bilgi Keşfi: Veri Madenciliği ve Bir Sağlık Uygulaması, İstanbul Üniversitesi, Yüksek Lisans Tezi, İstanbul, 2009

16. Çalışkan, K. S., K'KNN: Kümeleme ve K En Yakın Komşu Yöntemi ile Ağlarda Nüfuz Tespiti, Gebze Yüksek Teknoloji Enstitüsü, Yüksek Lisans Tezi, Gebze, 2008

17. Demiralay, M., Hiyerarşik Kümeleme Metotları ile Veri Madenciliği Uygulamaları, Marmara Üniversitesi, Yüksek Lisans Tezi, İstanbul, 2005

18. Kaya, H., Köymen, K., Veri Madenciliği Kavramı ve Uygulama Alanları, Doğu Anadolu Bölgesi Araştırmaları, 2008

19. Karabağ, R., Kullanıcı Davranış Analizi ile Nüfuz Tespiti, Gebze Yüksek Teknoloji Enstitüsü, Yüksek Lisans Tezi, Gebze, 2006

20. http://www.cinterion.com , Cinterion Company**,** Wireless Modules XT65 datasheet, 2008, (Erişim tarihi: 01.12.2010)

ÖZGEÇMİŞ

1983 yılında Manisa'da doğdu. İlköğretimini Manisa Sakarya İlköğretim okulunda, ortaöğretimini Manisa Dündar Çiloğlu Anadolu Lisesi'nde tamamladı.

2001 yılında Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Bölümünü kazandı ve 2005 yılında mezun oldu. 2006 - 2008 yılları arasında TÜBİTAK MAM Malzeme Enstitüsü'nde araştırmacı olarak görev yaptı. 2008-2009 yılları arasında askerlik görevini yedek subay olarak tamamladı. Nisan 2009'dan bu yana TÜBİTAK BİLGEM Bilişim Teknolojileri Enstitüsü'nde araştırmacı olarak görev yapmaktadır.

2006 yılında, Maltepe Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Yüksek Lisans programında yüksek lisans öğrenimi yapmaya hak kazandı. 2008 yılında askerlik görevi sebebiyle ara verdiği eğitim hayatına 2009 yılında tekrar başlamıştır.

Evli olan Önder Mustafa GEREN, İngilizce bilmektedir.