T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MAKİNA MÜHENDİSLİĞİ ANABİLİM DALI

BUHARLAŞMALI SOĞUTUCULARIN PSİKROMETRİK ANALİZE DAYALI PERFORMANS BELİRLEMESİ

YÜKSEK LİSANS TEZİ

OĞUZ ÇALIŞKAN

DENİZLİ, TEMMUZ - 2015

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MAKİNA MÜHENDİSLİĞİ ANABİLİM DALI

BUHARLAŞMALI SOĞUTUCULARIN PSİKROMETRİK ANALİZE DAYALI PERFORMANS BELİRLEMESİ

YÜKSEK LİSANS TEZİ

OĞUZ ÇALIŞKAN

DENİZLİ, TEMMUZ - 2015

KABUL VE ONAY SAYFASI

OĞUZ ÇALIŞKAN tarafından hazırlanan "**BUHARLAŞMALI SOĞUTUCULARIN PSİKROMETRİK ANALİZE DAYALI PERFORMANS BELİRLEMESİ**" adlı tez çalışmasının savunma sınavı 22.07.2015 tarihinde yapılmış olup aşağıda verilen jüri tarafından oy birliği / ~~oy çokluğu~~ ile Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Makina Mühendisliği Anabilim Dalı Yüksek Lisans Tezi olarak kabul edilmiştir.

Jüri Üyeleri	İmza
Danışman Doç. Dr. Mehmet Fevzi KÖSEOĞLU	.....
Üye Prof. Dr. Nazım USTA Pamukkale Üniversitesi	.....
Üye Yrd. Doç. Dr. Şükrü Ulaş ATMACA Selçuk Üniversitesi	.....

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 12/08/2015..... tarih ve ..30/10..... sayılı kararıyla onaylanmıştır.

.....

Prof. Dr. Orhan KARABULUT

Fen Bilimleri Enstitüsü Müdürü

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

OĞUZ ÇALIŞKAN

ÖZET

BUHARLAŞMALI SOĞUTUCULARIN PSİKROMETRİK ANALİZE DAYALI PERFORMANS BELİRLEMESİ

YÜKSEK LİSANS TEZİ

OĞUZ ÇALIŞKAN

PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNA MÜHENDİSLİĞİ ANABİLİM DALI

(TEZ DANIŞMANI: DOÇ. DR. MEHMET FEVZİ KÖSEOĞLU)

DENİZLİ, TEMMUZ - 2015

Enerji verimliliği her alanda olduğu gibi soğutma sistemlerinde de gün geçtikçe önem kazanmaktadır. Bina soğutma yükünün tasarım aşamasında belirlenip uygun soğutma sisteminin seçilmesi önemli bir ihtiyaçtır. Direkt buharlaşmalı soğutucular yüksek verimleri nedeniyle özellikle kurak iklimlerde geniş bir kullanım alanına sahiptir. Bu çalışmada, Visual C# programlama dilinde bir binanın soğutma yükünü ve bu yükü karşılayacak direkt buharlaşmalı soğutucu kapasitesini hesaplayan bir program hazırlanmıştır. Soğutucu kapasitesini hesaplamak için gereken doyma verimi iki ayrı deneysel çalışma sonucunda saptanmıştır. Kullanıcı dostu arayüz ile hazırlanan program yapı elemanları, aydınlatma, insanlar ve cihazlardan kaynaklanan soğutma yüklerini ışınım zaman serileri yöntemi ile 24 saatlik dönem boyunca her saat için hesaplayarak maksimum yük ve maksimum yükün oluştuğu saati, bu yükü karşılayacak hava debisini, soğutucu adedini ve soğutuculara beslenmesi gereken su debisini belirlemektedir. Bu program sayesinde bir binanın soğutma yükü tasarım aşamasında hesaplanabilmekte, farklı yapı elemanlarının soğutma yüküne etkileri kolayca karşılaştırılabilmekte ve buharlaşmalı soğutma sistemlerinin girilen iklim koşulları için uygun olup olmadığı belirlenebilmektedir.

ANAHTAR KELİMELER: Buharlaşmalı soğutma, soğutma yükü, psikrometri

ABSTRACT

PERFORMANCE EVALUATION OF EVAPORATIVE COOLERS BASED ON PSYCHROMETRIC ANALYSIS
MSC THESIS
OĞUZ ÇALIŞKAN
PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE
MECHANICAL ENGINEERING
(SUPERVISOR: ASSOC. PROF. DR. MEHMET FEVZİ KÖSEOĞLU)

DENİZLİ, JULY 2015

Energy efficiency gains importance in cooling systems like all fields day by day. It is an important need to determine the cooling load of a building during design and select suitable cooling system. Direct evaporative coolers have a large usage area due to their energy efficiencies, especially in arid climates. In this study, a software which calculates the cooling load of a building and the capacity of direct evaporative cooler to satisfy the cooling load has been developed in Visual C# programming language. Saturation efficiency which is required to calculate the cooler capacity was determined as a result of two different experimental studies. The software has been developed with user-friendly interface and it calculates hourly cooling loads occurring from building components, lighting, occupants and devices with radiant time series method in a period of 24 hours, determines the peak load and the hour at which the peak load occurs, the air flow to satisfy the peak load, number of coolers and the water flow which needs to be supplied to coolers. With this software, one can calculate the cooling load of a building during design, easily compare effects of different building components to the cooling load and determine whether or not evaporative cooling systems are suitable for climate conditions given.

KEYWORDS: Cooling load, evaporative cooling, psychrometry

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

SEMBOL LİSTESİ

A	:	Yüzey alanı, m^2
ab	:	Direkt hava kütlesi üssü
ad	:	Yayılan hava kütlesi üssü
AST	:	Belirgin güneş zamanı, ondalık saat
$\mathbf{c_n}$	:	n. saatteki iletim zaman faktörü
DST	:	Yaz saati, ondalık saat
$\mathbf{E_0}$	:	Dünya dışı ışınım akısı, W/m^2
$\mathbf{E_b}$	:	Direkt güneş ışınımı, W/m^2
$\mathbf{E_d}$	:	Yayılan güneş ışınımı, W/m^2
$\mathbf{E_t}$	:	Yüzeye gelen toplam güneş ışınımı, W/m^2
$\mathbf{E_{t,b}}$	:	Yüzeye gelen direkt güneş ışınımı, W/m^2
$\mathbf{E_{t,d}}$	:	Yüzeye gelen yayılan güneş ışınımı, W/m^2
$\mathbf{E_{t,r}}$	:	Yüzeye yerden yansıyan güneş ışınımı, W/m^2
ET	:	Zaman eşitliği, dak.
H	:	Saat açısı, °
h	:	Özgül entalpi, kJ/kg
$\mathbf{h_{dış}}$	:	Dış yüzeyde uzun dalga ışınım ve taşınım ısı transfer katsayısı, W/m^2K
L	:	Enlem, °
LON	:	Boylam, °
LPD	:	Birim alan başına aydınlatma gücü, W/m^2
LSM	:	Yerel standart meridyen boylamı, °
LST	:	Yerel standart zaman, ondalık saat
m	:	Bağıl hava kütlesi, kg/kg
P	:	Makine çıkış gücü, W
$\mathbf{P_{atm}}$	:	Atmosfer basıncı, Pa
$\mathbf{P_{buhar}}$	:	Su buharının kısmi basıncı, Pa
$\mathbf{P_{doyma}}$	:	Su buharının doyma basıncı, Pa
$\mathbf{P^*_{doyma}}$	:	Su buharının doyma basıncı (yaş termometre sıcaklığına göre), Pa
$\mathbf{q_\theta}$	:	θ. saatteki iletimsel ısı kazancı, W
$\mathbf{q_{direkt}}$	:	Direkt güneş ısı kazancı, W
$\mathbf{q_{em}}$	:	Makine ısı kazancı, W
$\mathbf{q_{giren,\theta-n}}$	:	(θ-n). saatteki ısı girişi, W
$\mathbf{Q_{ışınım,\theta-n}}$	:	(θ-n). saatteki ışınımsal soğutma yükü, W
$\mathbf{q_{ışınım,\theta-n}}$	:	(θ-n). saatteki ışınımsal ısı kazancı, W
$\mathbf{q_{iletim}}$	:	İletim ısı kazancı, W
$\mathbf{Q_{toplam}}$	:	Toplam duyulur soğutma yükü, W
$\mathbf{q_{yayılan}}$	:	Yayılan güneş ısı kazancı, W
$\mathbf{r_n}$	:	n. saatteki ışınım zaman faktörü
SHGC(θ)	:	θ geliş açısı için güneş ısı kazanç katsayısı
$\mathbf{\langle SHGC\rangle_D}$	:	Yayılan güneş ısı kazanç katsayısı
T	:	Mutlak kuru termometre sıcaklığı, K
T*	:	Mutlak yaş termometre sıcaklığı, K
t	:	Kuru termometre sıcaklığı, °C
t*	:	Yaş termometre sıcaklığı, °C
$\mathbf{t_{çıkış}}$	:	Buharlaşmalı soğutucu çıkış sıcaklığı, °C

$t_{çiy}$	:	Çiy noktası sıcaklığı, °C
$t_{dış}$	:	Dış ortam sıcaklığı, °C
t_e	:	Eşdeğer sıcaklık, °C
$t_{e,\theta-n}$	:	(θ-n). saatteki eşdeğer sıcaklık, °C
$t_{giriş}$	:	Buharlaşmalı soğutucu giriş sıcaklığı, °C
$t_{iç}$	:	İç ortam sıcaklığı, °C
TZ	:	Zaman dilimi
U	:	Isı transfer katsayısı, W/m^2K
ν	:	Buharlaşmalı soğutucu hava hızı, m/s
$\dot{V}_{hava}$	:	Buharlaşmalı soğutucu hava debisi, m^3/h
v_{hava}	:	Nemli havanın özgül hacmi, m^3/kg
$\dot{V}_{su}$	:	Buharlaşmalı soğutucu su debisi, m^3/h
v_{su}	:	Sıvı fazda suyun özgül hacmi, m^3/kg
w	:	Özgül nem, kg su buharı/kg kuru hava
w^*	:	Özgül nem (yaş termometre sıcaklığına göre), kg su buharı/kg kuru hava
$w_{çıkış}$	:	Buharlaşmalı soğutucu çıkışında özgül nem, kg su buharı/kg kuru hava
w_{doyma}	:	Doyma durumunda özgül nem, kg su buharı/kg kuru hava
w^*_{doyma}	:	Doyma durumunda özgül nem (yaş termometre sıcaklığına
$w_{giriş}$	:	Buharlaşmalı soğutucu girişinde özgül nem, kg su buharı/kg kuru hava
Y	:	Dikey ışınımın yatay ışınıma oranı
z	:	Yükselti, m

Yunan sembolleri

α	:	Yüzeyin güneş ışınımı emiciliği
β	:	Güneş irtifa açısı, °
γ	:	Yüzey-güneş azimut açısı, °
δ	:	Sapma açısı, °
ΔR	:	Dış hava sıcaklığındaki siyah cisim tarafından yapılan ışıma ile yüzeye gökyüzü ve çevresinden gelen uzun dalga ışınım arasındaki fark, W/m^2
ε	:	Yüzeyin yarı küresel yayınımı
ε	:	Buharlaşmalı soğutucu doyma verimi
η	:	Makine verimi
θ	:	Geliş açısı, °
ρ_g	:	Zemin yansıtıcılığı
Σ	:	Yüzey eğim açısı, °
τ_b	:	Direkt optik derinlik
τ_d	:	Yayılan optik derinlik
ϕ	:	Bağıl nem göre), kg su buharı/kg kuru hava
ϕ	:	Güneş azimut açısı, °
$\phi_{çıkış}$	:	Buharlaşmalı soğutucu çıkışında bağıl nem
$\phi_{giriş}$	:	Buharlaşmalı soğutucu girişinde bağıl nem
ψ	:	Yüzey azimut açısı, °

KISALTMA LİSTESİ

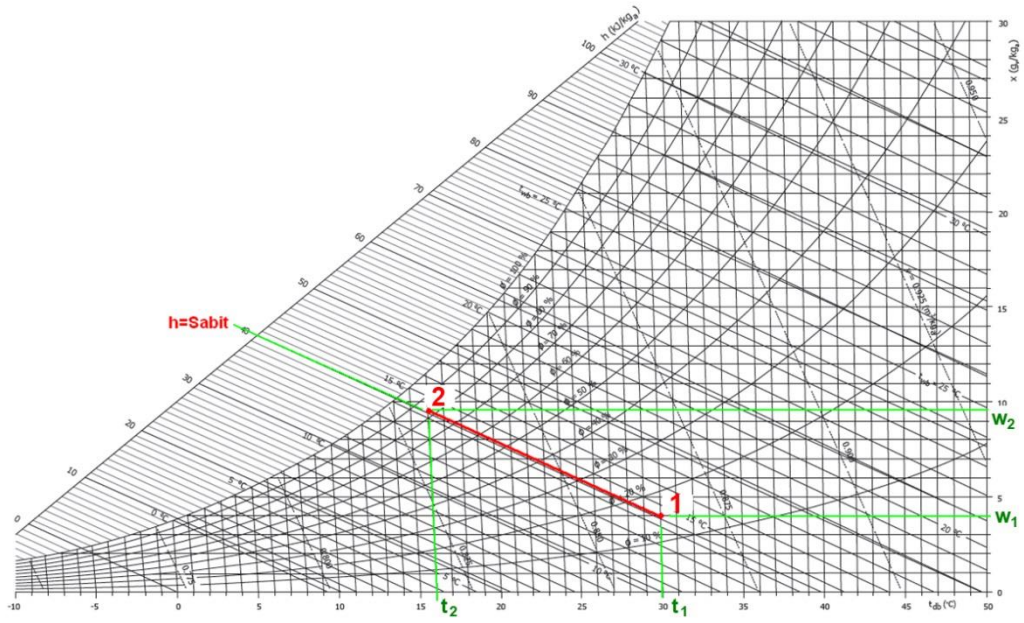
ASHRAE	**:**	American Society of Heating, Refrigerating and Air-Conditioning Engineers
COP	**:**	Coefficient of performance
EER	**:**	Energy efficiency ratio
EPS	**:**	Expanded polystyrene
NTC	**:**	Negative temperature coefficient
TETD/TA	**:**	Total equivalent time differential/time averaging
TÜMAS	**:**	Türkiye Meteorolojik Veri Arşiv Sistemi
RTS	**:**	Radiant time series
UTC	**:**	Universal Coordinated Time
VDI	**:**	Verein Deutscher Ingenieure

ÖNSÖZ

Bu çalışma, zorlu bir hesaplama ve programlama sürecinin eseridir. Bu zorlu süreçte desteklerini esirgemeyen değerli danışmanım Doç. Dr. Mehmet Fevzi KÖSEOĞLU'na, Selçuk Üniversitesi Makine Mühendisliği Bölümü değerli hocalarına, her zaman yardımıma koşan yüksek lisans arkadaşım Arş. Gör. Osman YELER'e, deneysel çalışmalarda bana kapılarını açan Basri KABAETLİ ve Yaşam Mühendislik Ltd. Şti. personeline şükranlarımı sunarım.

1. GİRİŞ

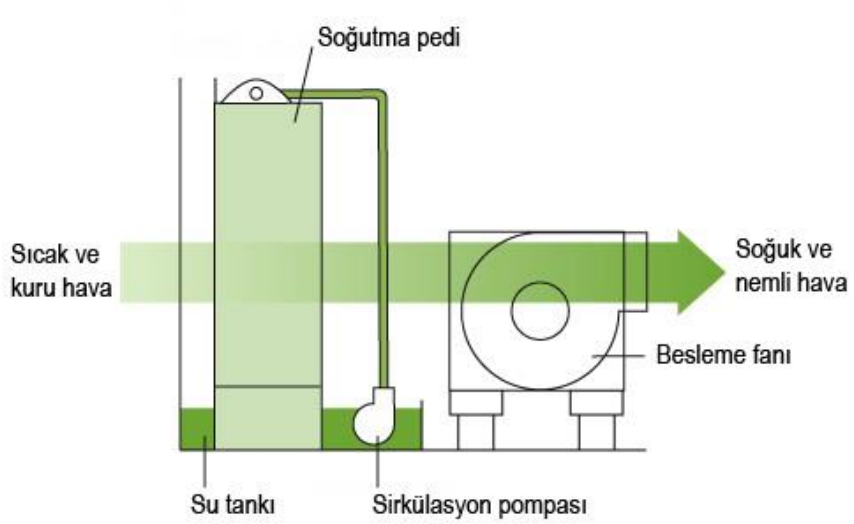
Buharlaşmalı soğutma işlemi basit bir ilkeye dayanır. Sıvı fazdaki su, buharlaşma gizli ısısını çevresindeki havadan çekerek buhar fazına geçer ve böylece havanın entalpisini düşürür. İdeal gaz olarak kabul edilen havanın entalpisi sadece sıcaklığın fonksiyonudur (Owen 2009), bu nedenle entalpisi düşen havanın sıcaklığı da düşmüş olur. Soğuyan havanın nem miktarında buharlaşmadan dolayı artış meydana gelir. Buharlaşmalı soğutma işlemi sırasında nemli havanın toplam entalpisi ve yaş termometre sıcaklığı yaklaşık olarak sabittir (Şekil 1.1). Bu nedenle bu işlem adyabatik olarak kabul edilir. Bu prensibe dayanarak soğutma işlemi gerçekleştiren buharlaşmalı soğutucuların direkt, indirekt ve yarı direkt olmak üzere üç tipi vardır.



Şekil 1.1: Buharlaşmalı soğutma işleminin psikrometrik diyagram üzerinde gösterimi.

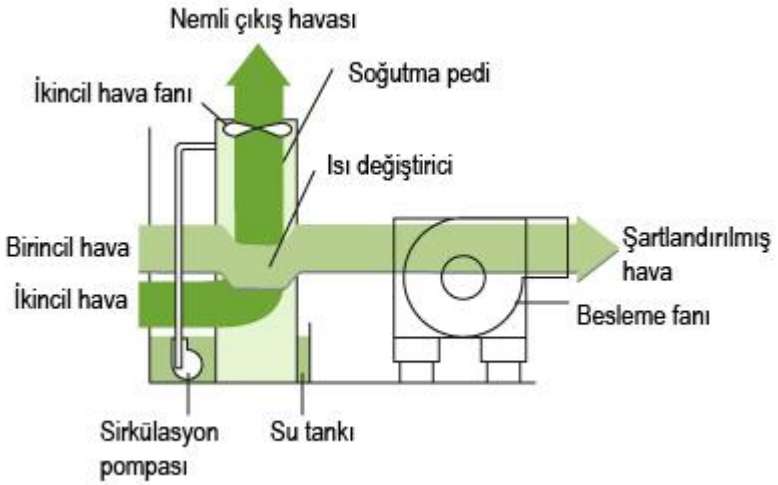
Direkt buharlaşmalı soğutucular besleme fanı, soğutma pedi, sirkülasyon pompası ve su tankından oluşan basit sistemlerdir (Şekil 1.2). Soğutma pedleri genellikle oluklu kağıt malzemeden üretilir. Sirkülasyon pompası vasıtasıyla soğutma pedine su beslenerek pedin sürekli ıslak kalması sağlanır. Fan vasıtasıyla

sisteme beslenen hava, soğutma pedinin olukları arasından geçerek içerideki su damlalarını buharlaştırır. Bu sayede hava soğuk ve nemli olarak sistemden çıkar.



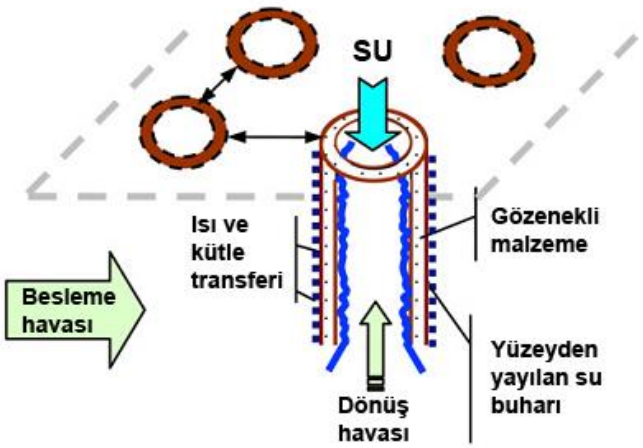
Şekil 1.2: Direkt buharlaşmalı soğutucu şematik gösterimi (url-1).

İndirekt buharlaşmalı soğutucularda besleme fanı, soğutma pedi, sirkülasyon pompası ve su tankına ek olarak bir ısı değiştirici bulunur (Şekil 1.3). Bu sistemlerde birincil ve ikincil olmak üzere iki hava akımı vardır. Birincil akımda sıcak hava ısı değiştiriciye gelerek ısısını, soğutma pedinden geçerek soğuyan ve nemlenen ikincil hava akımına verir. Nemli olan ikincil hava dışarı atılırken, nemlendirilmeden soğutulan birincil hava ortama verilir. İndirekt buharlaşmalı soğutucuların verimleri, direkt tipe göre düşüktür.



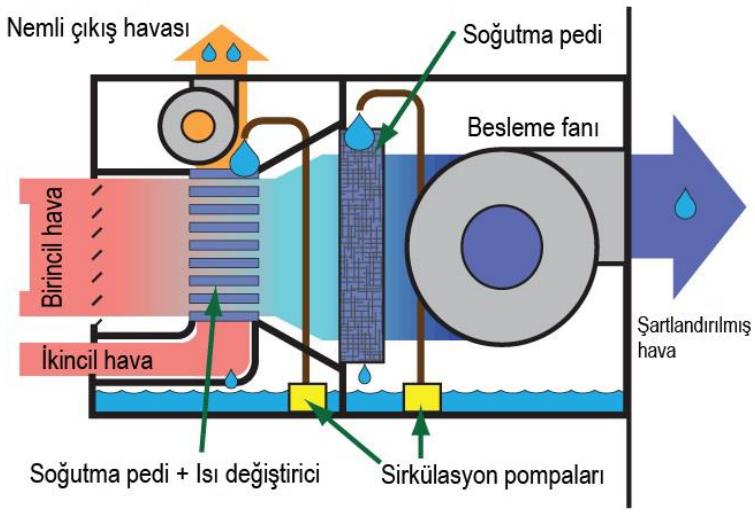
Şekil 1.3: İndirekt buharlaşmalı soğutucu şematik gösterimi (url-1).

Yarı direkt buharlaşmalı soğutucular, besleme havasının nem miktarına göre direkt veya indirekt olarak çalışan cihazlardır. Seramik gibi gözenekli malzemeden oluşan soğutma elemanına sahiptirler (Şekil 1.4).



Şekil 1.4: Yarı direkt buharlaşmalı soğutma elemanı (Martin 2009).

Bu üç tip haricinde direkt ve indirekt buharlaşmalı soğutmanın kombine edildiği sistemler de vardır (Şekil 1.5). Bu sistemlerde hava önce indirekt buharlaşmalı soğutucudan geçerek nemlendirilmeden bir miktar soğutulur, daha sonra direkt buharlaşmalı soğutucuya gönderilerek nemli ve daha soğuk biçimde sistemden çıkar. Çıkış havasının nem miktarı, sadece direkt buharlaşmalı soğutma yapan sistemlere göre daha düşüktür.



Şekil 1.5: Direkt ve indirekt kombine tip buharlaşmalı soğutucu şematik gösterimi (url-2).

Buharlaşmalı soğutucuların başta tekstil fabrikaları, cam üretim tesisleri, fırınlar, tavuk çiftlikleri, seralar, açık hava alanları olmak üzere geniş bir kullanım alanı vardır. Ayrıca alternatif olarak iklimlendirme ünitelerinde yoğuşturucu sıcaklığını düşürmek için, güç santrallerinde türbin soğutmasında ve klima cihazlarıyla kombine olarak kullanılmaktadır.

Buhar sıkıştırmalı soğutma sistemleriyle karşılaştırıldığında buharlaşmalı soğutucuların avantajları şunlardır:

- Yapıları basittir.
- Yatırım ve işletme maliyetleri düşüktür.
- Sadece fan ve pompa tarafından güç tüketildiği için verimleri yüksektir.
- Çevreye zararlı akışkan kullanmazlar.
- Ortama sürekli taze hava beslerler.

Tüm bu avantajlarına rağmen buharlaşmalı soğutucuların, buhar sıkıştırmalı soğutma sistemlerine karşı bazı dezavantajları vardır:

- Sürekli su beslemesine ihtiyaç duyarlar.
- Nemli iklimlerde ve fazla nemin istenmediği yerlerde kullanımı uygun değildir.
- Verimli çalışabilmeleri için besleme suyunun saflığı iyi olmalıdır. Bunun için su yumuşatma ünitelerine ihtiyaç duyulabilir. Bu da ek maliyet yaratır.
- Hassas bir şekilde nem kontrolü yapılamaz. Özellikle elektronik ekipmanların bulunduğu ortamlarda fazla nem tehlikeli olabileceği için buharlaşmalı soğutucular önerilmez.

2. LİTERATÜR ARAŞTIRMASI

Buharlaşmalı soğutucularla ilgili deneysel, simülasyon ve matematiksel modelleme olarak birçok çalışma mevcuttur. Buradaki literatür taramasında, tez konusu olan direkt buharlaşmalı soğutucular üzerine ağırlık verilmiştir.

Camargo ve diğ. (2005) direkt buharlaşmalı soğutucuda basit bir kontrol hacmi için enerji korunumunu yazıp su ve nemli hava arasındaki ısı ve kütle transferini analiz ederek bir matematiksel model geliştirmiş, bu modelle belirledikleri doyma verimi ve ısı taşınım katsayısını Brezilya'nın Taubate kentinde elde ettikleri deneysel verilerle karşılaştırmışlardır. Fouda ve Melikyan (2011) benzer bir çalışmayla suyun buharlaşma gizli ısısını ısı kaynağı, buharlaşan suyun kütlesini de kütle kaynağı olarak alıp bunları enerji ve kütle korunum denklemlerine yerleştirerek direkt buharlaşmalı soğutucular için basitleştirilmiş bir matematiksel model geliştirmiş ve matematiksel hesaplamalar ile deneysel sonuçlar arasında tutarlılık gözlemlemiştir. Yazarlar ayrıca giriş hava hızı, ped kalınlığı, giriş kuru termometre sıcaklığı faktörlerinin soğutma performansına etkilerini hesaplayıp analiz etmişlerdir. Halasz (1998) ise tüm buharlaşmalı soğutucu tiplerini kapsayan bir matematiksel model geliştirerek sabit parametreler içeren boyutsuz lineer diferansiyel denklemler türetmiştir. Yazar bu sayede sadece birkaç parametre ve diyagram ile soğutucuların performansının ifade edilebileceğini göstermiştir. Elmetenani ve diğ. (2011) deneysel çalışmanın aksine METEONORM yazılımını kullanarak Cezayir'in Bechar kenti için Haziran, Temmuz ve Ağustos aylarına ait meteorolojik verileri üretip TRNSYS yazılımı üzerinde oluşturdukları simülasyon ortamında direkt buharlaşmalı soğutucuların EER değerlerini hesaplamışlardır. Yazarlar buharlaşmalı soğutucuların düşük güç tüketimi üzerinde durarak, fotovoltaik paneller ile çalıştırılabileceğini göstermişlerdir.

Hajidavalloo (2007) pencere tipi klima kondenserinin her iki tarafına da buharlaşmalı soğutma pedi koyarak su enjekte etmiş ve güç tükeminde yaklaşık %16, COP değerinde ise yaklaşık %55 artış gözlemlemiştir. Yazar buharlaşmalı soğutma destekli klimaların konvansiyonel klimalara göre kendini yaklaşık 1 yılda amorti edebileceğini belirlemiştir. Benzer bir çalışma Wang ve diğ. (2014) tarafından split

klimalar için yapılmıştır. Yazarlar split klima kondenserinin girişine buharlaşmalı soğutma ünitesi koyarak kompresör güç tüketiminde %14,3'e kadar azalma, evaporatöre giren sıvı fazdaki soğutucu akışkan debisinde artış, COP değerinde %6,1-18 arasında artış gözlemlemiş ve buharlaşmalı soğutma sisteminin kendini 2 ila 3 yılda amorti edebileceğini belirlemiştir.

Lazzarin (2007) psikrometrik diyagramı farklı bölgelere ayırarak tüm iklim koşulları için direkt ve indirekt buharlaşma tekniklerini analiz etmiş ve bu tekniklerin belirtilen iklim koşulları için uygun olup olmadığını belirlemiştir.

Guo ve Zhao (1998) plakalı eşanjörlü indirekt buharlaşmalı soğutucuların ısıl performansını nümerik olarak analiz ederek birincil ve ikincil hava akımındaki hız, kanal genişliği, girişteki bağıl nem, plakanın ıslatılabilirliği gibi faktörlerin ısıl performansa etkilerini incelemişlerdir. Yazarlar küçük kanal genişliği, ikincil hava akımının girişinde düşük bağıl nem, plakanın ıslatılabilirliğinin yüksek olması ve ikincil hava akımında yüksek hız durumunda daha yüksek performans gözlemlemiştir. Riffat ve Zhu (2004) indirekt buharlaşmalı soğutuculardaki ısı transfer elemanları üzerinde durarak, soğutma elemanı olarak gözenekli seramik, ısı değiştiricisi olarak ısı borusu kullanmış ve bu soğutucunun özelliklerini simüle edebilmek için bir matematiksel model geliştirerek elde ettikleri verileri deneysel sonuçlarla doğrulamışlardır. Yazarlar elde ettikleri bulgularda kuru ve rüzgarlı iklim ile giriş havasının 0,6 m/s civarında sağlanmasının soğutucuda yüksek performans gösterdiğini, performansı daha da artırmak için ısı borusu ile seramik yüzey arasındaki ısı iletiminin artırılması gerektiğini gözlemlemişlerdir. Tulsidasani ve diğ. (1998) ise borulu eşanjörlü indirekt buharlaşmalı soğutucuların performansını belirlemek için çevre faktörü ve soğutma faktörü olmak üzere iki adet boyutsuz parametre belirleyerek ısı transferi temelli bir matematiksel model geliştirmişlerdir. Yazarlar belli boyutlardaki ortamdan maksimum ısıyı atabilecek optimum soğutucu kapasitesi için doğrusal bir ilişki bulmuşlardır.

Steeman ve diğ. (2009) dönüş havasının adyabatik nemlendirme ile soğutulup hava/hava ısı değiştiricisi vasıtasıyla besleme havasının sıcaklığının düşürüldüğü indirekt buharlaşmalı soğutmalı iklimlendirme ünitelerinin ısıl performansı ile bu sistemin uygulandığı binaların ısı ve kütle dengesi arasındaki ilişkiyi belirlemek için TRNSYS ortamında bir simülasyon metodolojisi geliştirmiştir. Yazarlar sistem

performansının hava giriş koşullarından bağımsız olduğunu, iç ortamdaki nemin artması ve havalandırma hızının azalması durumunda sistem performansının düştüğünü gözlemlemiştir. Joudi ve Mehdi (2000) ise taze havalı ve karışım havalı olmak üzere iki tip indirekt buharlaşmalı soğutmalı iklimlendirme ünitesini Irak'ın Bağdat kentine ait iklim koşullarında simüle ederek değişken soğutma yükleri altında incelemiş ve indirekt buharlaşmalı soğutmanın, çalıştırılma süresinin büyük kısmında konfor koşullarını sağladığını, sadece fan ve pompa tarafından güç tüketildiği için yüksek performans gösterdiğini belirlemiştir.

Martin (2009) iklimlendirme sistemlerinde enerji geri kazanımı amaçlı kullanılan gözenekli seramik borulu yarı direkt buharlaşmalı soğutucuların ısıl performans karakteristiklerini deneysel olarak incelemiş, besleme havasının düşük nem ve yüksek sıcaklıkta olması durumunda seramik boruların yüzeyindeki ana etkinin buharlaşma olduğunu, hem nem hem de sıcaklığın yüksek olması durumunda ise nem alma ve dolayısıyla yoğuşma meydana geldğini ve duyulur ile gizli ısının geri kazanıldığını gözlemlemiştir.

Dai ve Sumathy (2002) çapraz akışlı direkt buharlaşmalı soğutucuda soğutma elemanı olarak petek yapıda kağıt malzemeyi matematiksel modelleme ile incelemiş ve hava sıcaklığının yaklaşık 9 °C düşürülebildiğini, bağıl nemin yaklaşık %50 artırılabildiğini saptamışlardır. Pires ve diğ. (2011) farklı yapı ve tekstil malzemelerinin buharlaşmalı soğutma kabiliyetlerini deneysel olarak inceleyerek petek yapıda, polyester ara parçalı kumaş malzemenin en iyi performansı gösterdiğini belirlemişlerdir. Zhao ve diğ. (2008) ise benzer bir çalışmayla metal, elyaf, seramik, zeolit ve karbon malzemelerin indirekt buharlaşmalı soğutma sistemlerindeki ısı ve kütle transferine etkilerini incelemiş ve malzemelerin termal özelliklerinden çok biçim, dayanıklılık, su geçirmez kaplama ile uyumluluk ve kirlenme riski faktörlerinin soğutma performansında önemli olduğunu belirlemişlerdir. Yazarlar fitilden elde edilmiş alüminyum levhanın en uygun yapı ve malzeme olduğunu saptamışlardır.

Oranlıer ve Eyriboyun (2009) ASHRAE tarafından geliştirilen "*Toplam Eşdeğer Sıcaklık Farkı/Zaman Ortalaması*" (*Total Equivalent Time Differential/Time Averaging – TETD/TA*) yöntemini kullanarak Visual Basic 6.0 programla dilinde kullanıcı dostu arayüze sahip bir hesap yazılımı geliştirmiştir. Yazarlar bu yazılımda

24 saatlik dönem boyunca anlık duyulur ve gizli ısı kazançlarını her saat için hesaplayarak maksimum yükün oluştuğu saati belirlemişlerdir. Özgören ve diğ. (2011), İzmir ili için 1997-2008 yılları arasındaki meteorolojik verileri kullanarak yıllık ortalama sıcaklık, güneş ışınımı ve ortalama rüzgar hızı değerlerini analiz edip, daha güncel olan "*Işınım Zaman Serileri*" (*Radiant Time Series – RTS*) yöntemiyle MATLAB ortamında model bir konutun saatlik ısı kazancı ve soğutma yükünü hesaplamıştır. Bulut ve diğ. (2006) ise ASHRAE ve VDI tarafından geliştirilen soğutma yükü hesap yöntemleri ile piyasada kullanılan basit hesap ve bilgisayar programlarıyla yapılan hesap yöntemlerini örnek bir binaya uygulayarak karşılaştırmıştır. Yazarlar değişik yöntemlerle buldukları soğutma yükleri arasında %5-45 arasında fark tespit etmiş ve bu farkın sebepleri olarak yöntemlerde kullanılan farklı katsayıları, binanın konumu için uygun olmayan tablo değerlerini ve yapı malzemeleri için yöntemlerde verilen tablolarda uygun değerlerin olmamasını göstermişlerdir.

3. TEORİK YÖNTEM VE HESAPLAMALAR

Buharlaşmalı soğutucuların performansı, giriş ve çıkıştaki kuru termometre sıcaklıkları ile yaş termometre sıcaklığına göre belirlenir. Yaygın olarak kullanılan elektronik ölçüm aletleri kuru termometre sıcaklığı ve bağıl nemi ölçebilmekte fakat yaş termometre sıcaklığını ölçememektedirler. Yaş termometre sıcaklığının belirlenmesinde iterasyon yöntemine ihtiyaç vardır. İterasyonun yakınsaması için sınır değerlerinin iyi belirlenmesi gerekir.

3.1 Psikrometrik Hesaplamalar

Soğutucu giriş ve çıkış havasının psikrometrik özelliklerinin hesaplanmasında ASHRAE Temel El Kitabı 2009 basımında yer alan formüller kullanılmıştır (Owen 2009).

Verilen yükseklik değeri için atmosfer basıncı (3.1) denklemi ile hesaplanır.

$$P_{atm} = 101325\left(1 - 2{,}25577 \times 10^{-5} z\right)^{5{,}2559} \tag{3.1}$$

Verilen mutlak kuru termometre sıcaklığı için doyma basıncı (3.2) denklemi ile hesaplanır.

$$\begin{aligned} \ln P_{doyma} = & -5{,}8002206 \times 10^{3} \,/\, T + 1{,}3914993 - 4{,}8640239 \times 10^{-2} T \\ & + 4{,}1764768 \times 10^{-5} T^{2} - 1{,}4452093 \times 10^{-8} T^{3} \\ & + 6{,}5459673 \ln T \end{aligned} \tag{3.2}$$

Verilen mutlak yaş termometre sıcaklığı için doyma basıncı (3.3) denklemi ile hesaplanır.

$$\begin{aligned} \ln P^{*}_{doyma} = & -5{,}8002206 \times 10^{3} \,/\, T^{*} + 1{,}3914993 - 4{,}8640239 \times 10^{-2} T^{*} \\ & + 4{,}1764768 \times 10^{-5} (T^{*})^{2} - 1{,}4452093 \times 10^{-8} (T^{*})^{3} \\ & + 6{,}5459673 \ln T^{*} \end{aligned} \tag{3.3}$$

Verilen bağıl nem ve (3.2) denkleminde bulunan doyma basıncı için havanın kısmi buhar basıncı (3.4) denklemi ile hesaplanır.

$$P_{buhar} = \phi P_{doyma} \tag{3.4}$$

(3.2) denkleminde bulunan doyma basıncı ve (3.4) denkleminde bulunan kısmi buhar basıncı için bağıl nem (3.5) denklemi ile hesaplanır.

$$\phi = \frac{P_{buhar}}{P_{doyma}} \tag{3.5}$$

(3.1) denkleminde bulunan atmosfer basıncı ve (3.4) denkleminde bulunan kısmi buhar basıncı için özgül nem (3.6) denklemi ile hesaplanır.

$$w = 0,621945 \frac{P_{buhar}}{P_{atm} - P_{buhar}} \tag{3.6}$$

(3.1) denkleminde bulunan atmosfer basıncı ve (3.2) denkleminde bulunan doyma basıncı için doymuş havanın özgül nemi (3.7) denklemi ile hesaplanır.

$$w_{doyma} = 0,621945 \frac{P_{doyma}}{P_{atm} - P_{doyma}} \tag{3.7}$$

(3.1) denkleminde bulunan atmosfer basıncı ve (3.3) denkleminde bulunan doyma basıncı için doymuş havanın özgül nemi (3.8) denklemi ile hesaplanır.

$$w^*_{doyma} = 0,621945 \frac{P^*_{doyma}}{P_{atm} - P^*_{doyma}} \tag{3.8}$$

Kuru termometre sıcaklığı, yaş termometre sıcaklığı ve (3.8) denkleminde bulunan doyma durumundaki özgül nem için havanın özgül nemi (3.9) denklemi ile hesaplanır.

$$w^* = \frac{\left(2501 - 2,326t^*\right) w^*_{doyma} - 1,006\left(t - t^*\right)}{2501 + 1,86t - 4,186t^*} \tag{3.9}$$

(3.4) denkleminde bulunan kısmi buhar basıncı için çiy noktası sıcaklığı (3.10) denklemi ile hesaplanır.

$$t_{çiy} = 6,54 + 14,526\ln\left(P_{buhar}/1000\right) + 0,7389\left[\ln\left(P_{buhar}/1000\right)\right]^2 + 0,09486\left[\ln\left(P_{buhar}/1000\right)\right]^3 + 0,4569\left(P_{buhar}/1000\right)^{0,1984} \quad (3.10)$$

Verilen kuru termometre sıcaklığı ve (3.6) denkleminde bulunan özgül nem için özgül entalpi (3.11) denklemi ile hesaplanır.

$$h = 1,006t + w\left(2501 + 1,86t\right) \quad (3.11)$$

Verilen mutlak kuru termometre sıcaklığı, (3.1) denkleminde bulunan atmosfer basıncı ve (3.6) denkleminde bulunan özgül nem için havanın özgül hacmi (3.12) denklemi ile hesaplanır.

$$v_{hava} = \frac{0,287042T(1 + 1,607858w)}{P_{atm}/1000} \quad (3.12)$$

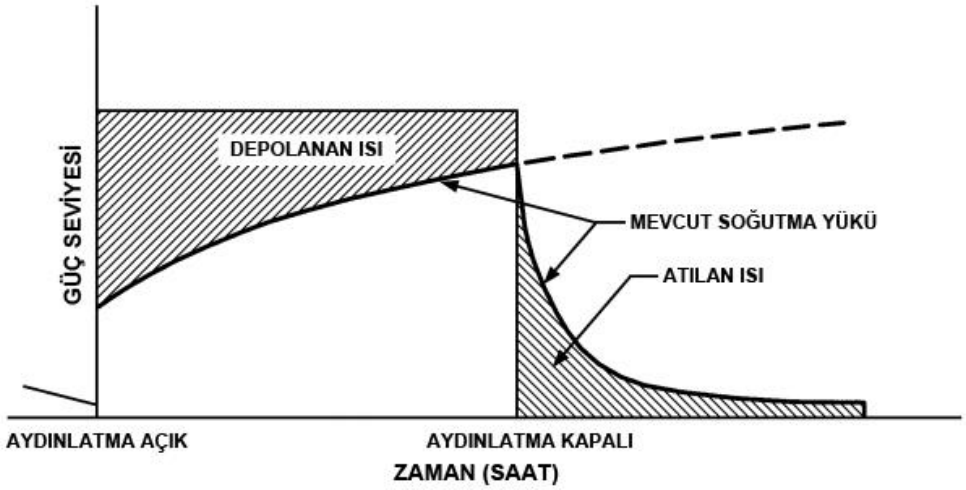
Yaş termometre sıcaklığının belirlenmesi iterasyon yöntemi ile yapılmaktadır. Yaş termometre sıcaklığının alabileceği en küçük değer çiy noktası sıcaklığı, en büyük değer kuru termometre sıcaklığı olduğu için tahmin değerlerinin bu sınırlar içerisinde olması gerekir. İlk tahmin değeri olarak kuru termometre sıcaklığı ile çiy noktası sıcaklığının aritmetik ortalaması alınarak iterasyona başlanır. Tahmin değeri ve kuru termometre sıcaklığı (3.9) denkleminde yerine konularak bulunan w* değeri, (3.6) denkleminde bulunan w değeri ile karşılaştırılır. w*<w ise alt sınır olarak ilk tahmin değeri, w*>w ise üst sınır olarak ilk tahmin değeri alınır. Alt ve üst sınır değerlerinin aritmetik ortalaması alınarak ikinci tahmin değeri belirlenir. Bu tahmin değeri (3.9) denkleminde yerine konularak yeni bir w* değeri bulunur ve w değeri ile karşılaştırılır. w* ile w değerleri arasındaki fark 0,001'den az olana kadar iterasyon sürdürülür. Bu farkın sağlandığı tahmin değeri yaş termometre sıcaklığı olarak belirlenir.

3.2 Soğutma Yükü Hesabı

Bina soğutma yükünün hesaplanmasında ASHRAE tarafından geliştirilen "*Işınım Zaman Serileri*" (*Radiant Time Series - RTS*) yöntemi kullanılmıştır (Owen 2009). 2005 yılında bu yöntem, "*Toplam Eşdeğer Sıcaklık Farkı/Zaman Ortalaması*"

(*Total Equivalent Time Differential/Time Averaging – TETD/TA*) yönteminin yerini almıştır.

Yapı elemanları ve mobilyalar gibi bileşenler tarafından emilen enerji ancak bir zaman gecikmesi ile mahal soğutma yüküne katılır. Bu enerjinin bir kısmı ısı kaynakları ortadan kalksa bile halen mevcuttur ve bir süre daha yayınlanmaya devam eder. Şekil 3.1 zaman gecikmesine örnek olarak aydınlatmadan kaynaklanan ısıl depolama etkisini göstermektedir.



Şekil 3.1: Soğutma yükünde aydınlatma kaynaklı ısıl depolama etkisi (Owen 2009).

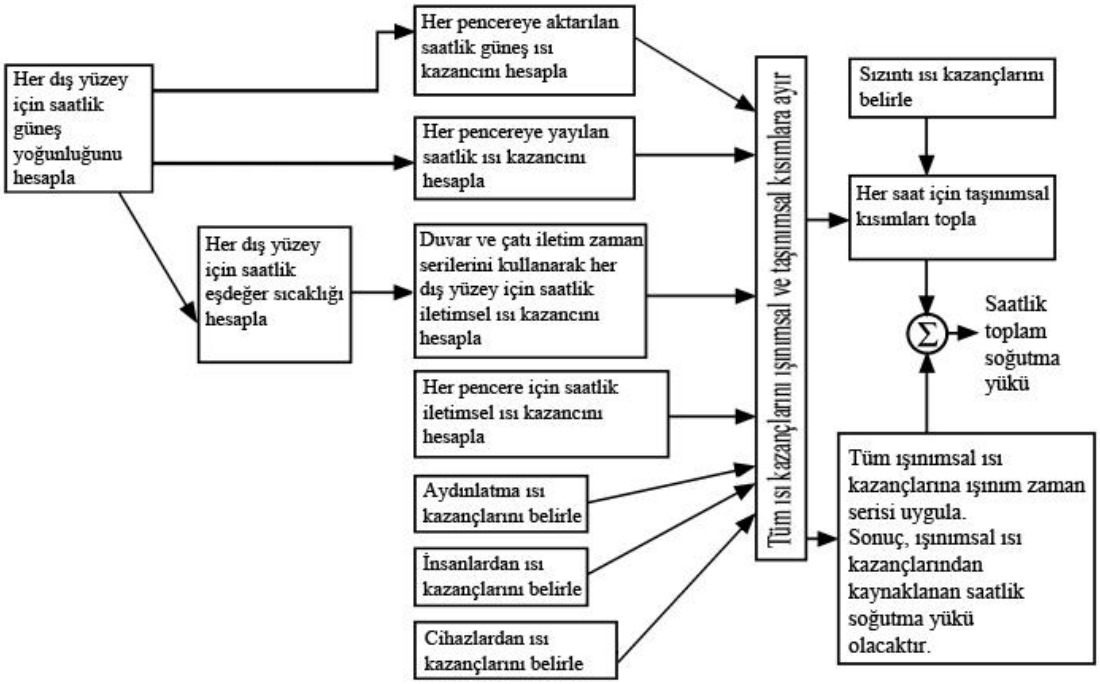
Isı kaynağının aktive edildiği zaman ile yayınlanan enerjinin anlık olarak depolanan enerjiye eşit olduğu nokta arasında daima belirgin bir gecikme vardır. Mahal için gereken soğutma yükü, anlık üretilen ısıdan çok düşük olabileceği ve pik soğutma yükünü önemli ölçüde etkileyebileceği için soğutma yükü hesabında zaman gecikmesi dikkate alınmalıdır.

Soğutma yükleri 24 saatlik döngüyle tekrar eden, periyodik olarak sabit koşullar yaklaşımına dayanır. Belli bir saatte belli bir bileşenin ısı kazancı 24, 48 vs. saat önceki ile aynıdır.

Bu yöntemde opak dış yüzeylerden meydana gelen iletimsel ısı kazançlarının gecikmesi ve ışınımsal ısı kazançlarının soğutma yüküne dönüşümünün gecikmesi olmak üzere iki tür zaman gecikmesi ele alınır. Dış duvarlar ve çatılar, dış ve iç hava arasındaki sıcaklık farkından dolayı ısı iletir. Buna ilave olarak, dış yüzeylerde güneş enerjisi emilir ve sonrasında iletim ile bina içine aktarılır. Duvar ve çatı yapı

malzemelerinin kütle ve ısıl kapasitelerinden dolayı, dış yüzeydeki ısı girişinin iç yüzeye aktarılmasında önemli bir gecikme meydana gelir.

Çoğu ısı kaynağı ortama enerjiyi taşınım ve ışınımın birleşimi şeklinde aktarır. Isı kazancının taşınımsal kısmı anında soğutma yüküne dönüşür. Işınımsal kısım iç yüzeylerdeki kütle tarafından emilip taşınım ile ortam havasına aktarıldıktan sonra soğutma yüküne dönüşür. Dolayısıyla ışınımsal ısı kazançları gecikmeli olarak soğutma yüküne dönüşür. Şekil 3.2 ışınım zaman serileri yönteminin hesap adımlarını göstermektedir.



Şekil 3.2: Işınım zaman serileri yöntemi hesap adımları (Owen 2009).

Işınım zaman serileri yöntemi, saatlik ısı kazançlarını 24 saatlik seriler ile çarparak hem iletimsel, hem de ışınımsal gecikmeleri hesaba katar. Bu çarpma işlemi, ısı kazançlarını zamana yayar. Zaman serilerinde *iletim zaman faktörleri* ve *ışınım zaman faktörleri* olmak üzere iki tür katsayı kullanılır. İletim ve ışınım zaman faktörleri, yine ASHRAE tarafından geliştirilen ve daha karmaşık olan "*Isı Dengesi*" (*Heat Balance - HB*) yöntemi ile türetilmiştir. İletim zaman faktörleri, duvar veya çatının dış yüzeyinde önceden meydana gelen ısı kazancının şimdiki saat boyunca içerideki ısı kazancına dönüşen yüzdesini ifade eder. Işınım zaman faktörleri ise önceden meydana gelen ışınımsal ısı kazancının şimdiki saat boyunca soğutma

yüküne dönüşen yüzdesini ifade eder. Bu seriler yapılardaki gecikme etkisinin birbirleriyle kolayca karşılaştırılabilmesi için kullanılabilir.

Direkt toprak ile temasta olan veya şartlandırılmayan bodrum üzerinde bulunan zeminlerde genellikle ısı kazancı yerine ısı kaybı olduğu için soğutma yükü hesabında dikkate alınmaz. Pozitif basınç altında bulunan ortamlarda dışarıdan içeriye hava sızıntısı olmadığı için sızıntı yoluyla ısı kazancı meydana gelmez. Bu çalışmada soğutma yükü hesaplanan binanın toprak ile temas ettiği ve buharlaşmalı soğutucu ile taze hava üflenerek pozitif basınç altında tutulduğu kabul edilmiştir. Eğer zemin, iç ortam sıcaklığından daha sıcak bir ortam üzerindeyse buradan gerçekleşecek ısı transferi ayrıca dikkate alınmalıdır.

Saatlik iletim zaman faktörleri belli duvar yapıları için Tablo A.1'de, belli çatı yapıları için Tablo A.2'de verilmiştir.

Işınım zaman faktörleri yapı elemanlarının kütle yoğunluğu ile ısıl kapasiteleri, ortamda halı olup olmaması ve cam yüzey alanının toplam ısı transfer alanına olan yüzdesine göre Tablo A.3 ve Tablo A.4'de verilmiştir.

Işınım zaman serileri yöntemi birçok yönden toplam eşdeğer sıcaklık/zaman ortalaması yöntemi ile aynı olmasına rağmen şu yönlerden farklılık gösterir:

- İletimsel ısı kazançlarının hesaplanması
- Tüm ısı kazançlarının ışınımsal ve taşınımsal kısımlara ayrılması
- Işınımsal ısı kazançlarının soğutma yüküne dönüşümü

3.2.1 Saatlik Güneş Işınımı Hesabı

Yüzeylere gelen güneş ışınımı miktarı, soğutma yükü hesaplamalarında önemli bir yer kaplar. Güneş ışınlarına maruz kalan yüzeylerin sıcaklığı dış ortam sıcaklığından daha yüksek olacağı için meydana gelen ısı transferi de farklı olacaktır. Saatlik güneş ışınımı hesaplamaları ASHRAE Temel El Kitabı 2009 basımında bulunan formüller kullanılarak yapılmıştır (Owen 2009). Güneş ışınımı değerleri bulutsuz, açık hava koşulları için geçerlidir.

Dünyanın yörüngesi bir miktar eliptik olduğundan dolayı *dünya dışı ışınım akısı* (E_0) değeri yıl boyunca değişkenlik gösterir. Dünyanın güneşe en yakın olduğu Ocak ayının başında maksimum değer olan 1412 W/m^2, güneşten en uzak olduğu Temmuz ayının başında minimum değer olan 1322 W/m^2 değerine ulaşır.

Dünyanın yörünge hızı yıl boyunca değişkenlik gösterdiği için *belirgin güneş zamanı* (*AST*) belirlenmiştir. Belirgin güneş zamanı, düzgün bir hızla çalışan saatin gösterdiği ortalama zamandan bir miktar farklıdır. Bu fark *zaman eşitliği* (*ET*) olarak adlandırılır.

Dünyanın ekvatoryal düzlemi 23,45° eğiktir. Bu nedenle dünya-güneş çizgisi ile ekvatoryal düzlem arasındaki açı olan *sapma açısı* (δ) yıl boyunca değişkenlik gösterir. Bu değişkenlik, eşit olmayan gündüz ve gece zamanlarıyla birlikte mevsim değişimlerine neden olur.

Tablo 3.1 her ayın 21. günü için hesaplanan yaklaşık astronomik verileri göstermektedir.

Tablo 3.1: Her ayın 21. günü için hesaplanan yaklaşık astronomik veriler (Owen 2009).

Ay	Yılın Günü	E_0 [W/m^2]	Zaman Eşitliği (ET) [dak.]	Sapma Açısı (δ) [°]
Ocak	21	1410	-10,6	-20,1
Şubat	52	1397	-14	-11,2
Mart	80	1378	-7,9	-0,4
Nisan	111	1354	1,2	11,6
Mayıs	141	1334	3,7	20,1
Haziran	172	1323	-1,3	23,4
Temmuz	202	1324	-6,4	20,4
Ağustos	233	1336	-3,6	11,8
Eylül	264	1357	6,9	-0,2
Ekim	294	1380	15,5	-11,8
Kasım	325	1400	13,8	-20,4
Aralık	355	1411	2,2	-23,4

Yerel standart zaman (LST) ve belirgin güneş zamanı (AST) arasındaki dönüşüm iki adımda gerçekleşir. İlk olarak yerel standart zamana, zaman eşitliği (ET) eklenir. Sonra bir boylam düzeltmesi eklenir. Bu boylam düzeltmesi, yerel boylam (LON) ile yerel zaman diliminin bulunduğu standart meridyen boylamı

(LSM) arasındaki her bir derece fark için 4 dakikaya denk gelir. Belirgin güneş zamanı (3.13) denklemi ile hesaplanır.

$$AST = LST + ET\ /\ 60 + (LON - LSM)\ /\ 15 \tag{3.13}$$

Standart meridyenlerin çoğu 0° Greenwich, İngiltere'den her 15 derecede bir bulunur. Yerel standart meridyen (LSM) boylamı, yerel zaman dilimine (TZ) göre (3.14) denklemi ile hesaplanabilir. Türkiye UTC+2 zaman diliminde bulunduğu için yerel standart meridyen boylamı 30°'dir.

$$LSM = 15TZ \tag{3.14}$$

Eğer yaz saati (DST) uygulanıyorsa, ilave bir düzeltmeye ihtiyaç vardır. Çoğu bölgede yaz saati ile yerel saat arasında 1 saatlik fark olduğundan dolayı yerel standart zaman, (3.15) denklemi ile bulunur.

$$LST = DST - 1 \tag{3.15}$$

Güneşin konumu, Şekil 3.3'de gösterildiği üzere yatay düzlem üzerindeki güneş irtifa ve güneyden ölçülen güneş azimut açıları ile ifade edilir. *Güneş irtifa açısı* (β), yatay düzlem ile güneşten çıkan bir çizgi arasındaki açı olarak tanımlanır. Bu açı, güneşin ufukta olduğu 0° ile tam tepede olduğu 90° arasında değişir. Negatif değerler gece zamanını ifade eder. *Güneş azimut açısı* (ϕ), dünya-güneş çizgisinin yatay düzlemdeki izdüşümünün, güneyinden olan açısal yer değiştirme olarak tanımlanır. Bu açı öğleden sonra için pozitif, öğleden önce için negatif değer alır.

Güneş irtifa ve azimut açıları yerel enlem (L), güneş sapma açısı (δ) ve saat açısına (H) bağlıdır. *Saat açısı* (H) dünyanın dönmesinden dolayı güneşin, yerel meridyenin doğusu veya batısından olan açısal yer değiştirmesi olarak tanımlanır ve (3.16) denklemi ile hesaplanır. Saat açısı öğle vakti 0° olur. Öğleden sonra pozitif, öğleden önce negatif değer alır.

$$H = 15(AST - 12) \tag{3.16}$$

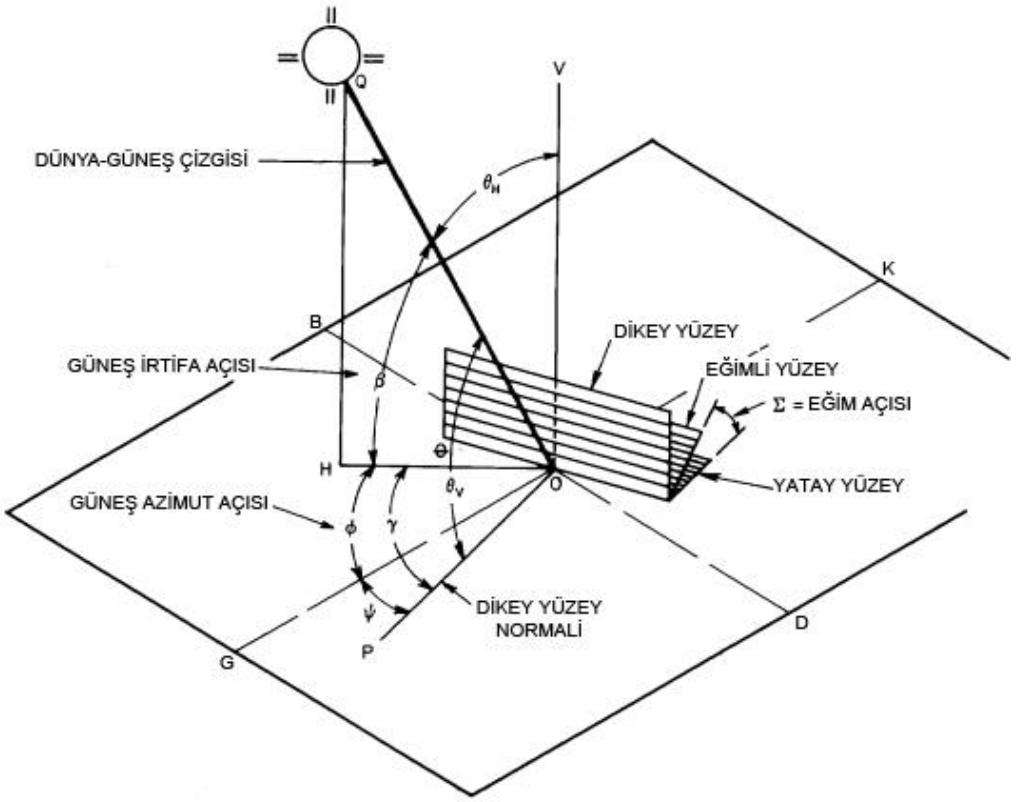
Güneş irtifa açısı (3.17) denklemi ile hesaplanır.

$$\sin\beta = \cos L \cos\delta \cos H + \sin L \sin\delta \tag{3.17}$$

Güneş azimut açısı (3.18) ve (3.19) denklemleri ile hesaplanır.

$$\sin\phi = \sin H \cos\delta / \cos\beta \tag{3.18}$$

$$\cos\phi = (\cos H \cos\delta \sin L - \sin\delta \cos L) / \cos\beta \tag{3.19}$$



Şekil 3.3: Yüzeyler için güneş açıları (Owen 2009).

Bağıl hava kütlesi (*m*), mevcut dünya ve güneş konumundaki atmosfer kütlesinin, güneşin tam tepede olduğu durumdaki kütleye oranıdır ve (3.20) denklemi ile hesaplanır.

$$m = 1 / [\sin\beta + 0{,}50572(6{,}07995 + \beta)^{-1{,}6364}] \tag{3.20}$$

Bulutsuz, açık hava koşullarında birim alan başına düşen güneş ışınımı *direkt* ve *yayılan* olmak üzere iki bileşenden oluşur. Direkt bileşen, direkt olarak güneş dairesinden meydana gelen ışınımı, yayılan bileşen ise gökyüzünün geri kalan kısmından meydana gelen ışınımı ifade eder. Direkt güneş ışınımı (E_b) (3.21) denklemi ile, yayılan güneş ışınımı (E_d) (3.22) denklemi ile hesaplanır.

$$E_b = E_0 \exp(-\tau_b m^{ab}) \tag{3.21}$$

$$E_d = E_0 \exp(-\tau_d m^{ad}) \tag{3.22}$$

τ_b ve τ_d katsayıları sırasıyla direkt ve yayılan optik derinlik değerlerini ifade eder. Bu değerler yerel koşullara bağlıdır ve yıl boyunca değişkenlik gösterir. ASHRAE aylık tasarım koşulları tablolarında her ayın 21. günü için Türkiye'de 56, dünya genelinde 6443 meteoroloji istasyonuna ait optik derinlik değerleri mevcuttur. Tablo 3.2 Denizli iline ait 21 Mayıs ve 21 Eylül tarihleri arasındaki optik derinlik değerlerini göstermektedir.

ab ve *ad* terimleri sırasıyla direkt ve yayılan hava kütlesi üsleridir. Bu terimler τ_b ve τ_d terimlerine (3.23) ve (3.24) denklemlerindeki deneysel bağıntılar aracılığıyla ilişkilendirilmiştir.

$$ab = 1{,}219 - 0{,}043\tau_b - 0{,}151\tau_d - 0{,}204\tau_b\tau_d \tag{3.23}$$

$$ad = 0{,}202 - 0{,}852\tau_b - 0{,}007\tau_d - 0{,}357\tau_b\tau_d \tag{3.24}$$

Tablo 3.2: Denizli iline ait 21 Mayıs - 21 Eylül arası optik derinlik değerleri (Owen 2009).

Tarih	τ_b	τ_d
21 Mayıs	0,427	2,138
21 Haziran	0,433	2,108
21 Temmuz	0,446	2,087
21 Ağustos	0,444	2,108
21 Eylül	0,419	2,225

Cephe yönlerine göre *yüzey azimut açısı* (ψ) değerleri Tablo 3.3'de verilmiştir.

Tablo 3.3: Cephe yönlerine göre yüzey azimut açıları (Owen 2009).

Cephe	**K**	**KD**	**D**	**GD**	**G**	**GB**	**B**	**KB**
Yüzey azimut açısı ψ	180°	-135°	-90°	-45°	0°	45°	90°	135°

Yüzey-güneş azimut açısı (γ), güneş azimut açısı (ϕ) ile yüzey azimut açısı (ψ) arasındaki fark olarak tanımlanır ve (3.25) denklemi ile hesaplanır.

$$\gamma = \phi - \psi \tag{3.25}$$

90°'den büyük veya -90°'den küçük γ değerleri yüzeyin gölgede olduğunu gösterir.

Işınıma maruz kalan yüzeyin normali ile dünya-güneş çizgisi arasındaki açı *geliş açısı* (θ) olarak adlandırılır ve (3.26) denklemi ile hesaplanır. Bu açı, yüzeye gelen direkt güneş ışınımı bileşeninin yoğunluğunu etkilediği için soğutma yükü hesaplamalarında önem arz eder.

$$\cos\theta = \cos\beta\cos\gamma\sin\Sigma + \sin\beta\cos\Sigma \tag{3.26}$$

Yüzey eğim açısının (Σ) 90° olduğu dikey yüzeyler için güneş geliş açısı basit olarak (3.27) denklemi ile ifade edilir.

$$\cos\theta = \cos\beta\cos\gamma \tag{3.27}$$

Yüzey eğim açısının (Σ) 0° olduğu yatay yüzeyler için güneş geliş açısı basit olarak (3.28) denklemi ile ifade edilir.

$$\theta = 90 - \beta \tag{3.28}$$

Bulutsuz, açık hava koşullarında birim cephe yüzeyine gelen güneş ışınımı *direkt*, *yayılan* ve *yerden yansıyan* olmak üzere üç bileşenden oluşur (3.29).

$$E_t = E_{t,b} + E_{t,d} + E_{t,r} \tag{3.29}$$

Direkt bileşen (3.30) denklemi ile hesaplanır. Bu ifade $\cos\theta > 0$ olduğunda geçerlidir, diğer durumlarda $E_{t,b} = 0$ alınır.

$$E_{t,b} = E_t\cos\theta \tag{3.30}$$

Dikey yüzeyler için yayılan bileşen (3.31) ve (3.32) denklemleri ile hesaplanır.

$$E_{t,d} = E_d Y \tag{3.31}$$

$$Y = \max(0,45; 0,55 + 0,437\cos\theta + 0,313\cos^2\theta) \tag{3.32}$$

Eğimli yüzeyler için $\Sigma \leq 90°$ ise (3.33) denklemi, $\Sigma > 90°$ ise (3.34) denklemi kullanılır.

$$E_{t,d} = E_d(Y \sin \Sigma + \cos \Sigma) \tag{3.33}$$

$$E_{t,d} = E_d Y \sin \Sigma \tag{3.34}$$

Yerden yansıyan bileşen tüm cephelerdeki yüzeyler için (3.35) denklemi ile hesaplanır. Zemin yansıtıcılığı (ρ_g) genel kullanımda 0,2 olarak alınır.

$$E_{t,r} = (E_b \sin \beta + E_d)\rho_g (1 - \cos \Sigma) / 2 \tag{3.35}$$

3.2.2 Duvar ve Çatı Soğutma Yükü Hesabı

Duvar ve çatı soğutma yükünün hesaplanması iki adımda gerçekleşir. İlk olarak iletim zaman serileri kullanılarak her saat için iletimsel ısı kazancı hesaplanır. Daha sonra bu ısı kazancı taşınımsal ve ışınımsal kısımlara ayrılır ve ışınımsal kısma ışınım zaman serileri uygulanarak ışınımsal soğutma yükü bulunur. Taşınımsal ısı kazancı direkt olarak soğutma yükü olacağı için bu değer ile ışınımsal soğutma yükü toplanır ve saatlik soğutma yükü hesaplanmış olur. Bu işlemler her saat için tekrarlanır.

İletimsel ısı kazançlarının hesaplanması *eşdeğer sıcaklığa* göre yapılır. *Eşdeğer sıcaklık* (t_e), tüm ışınımsal değişimlerin yokluğunda yüzeye aynı miktarda ısı girişi yapacak dış hava sıcaklığıdır ve (3.36) denklemi ile hesaplanır.

$$t_e = t_{dış} + \frac{\alpha E_t}{h_{dış}} - \frac{\varepsilon \Delta R}{h_{dış}} \tag{3.36}$$

Sadece gökyüzünden uzun dalga ışınım alan yatay yüzeyler için genel kullanımda ΔR değeri 63 W/m^2, ε değeri 1, $h_{dış}$ değeri 17 W/(m^{2}K) alınır. Dikey yüzeyler gökyüzünden olduğu kadar yerden ve çevredeki binalardan da uzun dalga ışınım alır ve genel kullanımda $\varepsilon\Delta R$ değeri 0 alınır. Eşdeğer sıcaklığın hesaplanmasında yüzey renginin de önemi vardır. Genel kullanımda $\alpha/h_{dış}$ değeri açık renkli yüzeyler için 0,026 ve koyu renkli yüzeyler için 0,052 alınır.

Duvar ve çatı için saatlik iletimsel ısı girişi (3.37) denklemiyle hesaplanır. Belli duvar tipleri için ısıl özellikler ve yapı bileşenleri Tablo A.5'de, belli çatı tipleri için ısıl özellikler ve yapı bileşenleri Tablo A.6'da verilmiştir.

$$q_{giren,\theta-n} = UA(t_{e,\theta-n} - t_{iç}) \tag{3.37}$$

Şimdiki saat ve önceki 23 saat için (3.37) denklemiyle hesaplanan iletimsel ısı girişleri ve duvar için Tablo A.1, çatı için Tablo A.2'den seçilen iletim zaman faktörleri kullanılarak duvarlar ve çatıdaki saatlik iletimsel ısı kazancı (3.38) denklemi ile hesaplanır.

$$q_\theta = c_0 q_{giren,\theta} + c_1 q_{giren,\theta-1} + c_2 q_{giren,\theta-2} + \ldots + c_{23} q_{giren,\theta-23} \tag{3.38}$$

Her saat için iletimsel ısı kazancı (3.38) denklemi ile hesaplandıktan sonra Tablo 3.4'de verilen yüzdeler kullanılarak ışınımsal ve taşınımsal kısımlara ayrılır. Taşınımsal ısı kazancı direkt olarak taşınımsal soğutma yüküne dönüşür. Işınımsal ısı kazancı ile bina yapısına göre Tablo A.3'den seçilen güneşe bağlı olmayan ışınım zaman faktörleri kullanılarak (3.39) denklemiyle saatlik ışınımsal soğutma yükü hesaplanır.

Tablo 3.4: Duvar ve çatı için taşınımsal ve ışınımsal ısı kazancı yüzdeleri (Owen 2009).

Isı Kazancı	Taşınımsal Kısım	Işınımsal Kısım
Duvar	%54	%46
Çatı	%40	%60

$$Q_{ışınım,\theta} = r_0 q_{ışınım,\theta} + r_1 q_{ışınım,\theta-1} + r_2 q_{ışınım,\theta-2} + \ldots + r_{23} q_{ışınım,\theta-23} \tag{3.39}$$

Son olarak taşınımsal ve ışınımsal soğutma yükleri toplanarak saatlik soğutma yükü hesaplanır.

3.2.3 Pencere ve Camlı Kapı Soğutma Yükü Hesabı

Pencere ve camlı kapı soğutma yükünün hesaplanmasında iletimsel ısı kazancı, direkt ve yayılan güneş ısı kazancı olmak üzere üç tür ısı kazancı ele alınır.

İletimsel ısı kazancı (3.40) denklemi ile hesaplanır. Belli pencere ve camlı kapı tipleri için ısı transfer katsayıları (U) Tablo 3.5'de verilmiştir.

$$q_{iletim} = UA(t_{dış} - t_{iç}) \tag{3.40}$$

Direkt güneş ısı kazancı (3.41) denklemi ile hesaplanır.

$$q_{direkt} = AE_{t,b}SHGC(\theta) \tag{3.41}$$

Yayılan güneş ısı kazancı (3.42) denklemi ile hesaplanır.

$$q_{yayılan} = A(E_{t,d} + E_{t,r})\left\langle SHGC \right\rangle_D \tag{3.42}$$

Tablo 3.5: Belli pencere ve camlı kapı tipleri için ısı transfer katsayıları (Owen 2009).

	Çerçeve Tipi		
	Alüminyum	Güçlendirilmiş Vinil/Alüminyum Kaplı Ahşap	Vinil/Ahşap
Cam Tipi	**Isı Transfer Katsayısı [W/m^2K]**		
3,2 mm Tek Cam	7,01	5,27	5,2
6,4 mm Boşluklu Çift Cam	4,62	3,24	3,14
12,7 mm Boşluklu Çift Cam	4,3	2,96	2,86
6,4 mm Boşluklu Üçlü Cam	3,78	2,46	2,42
12,7 mm Boşluklu Üçlü Cam	3,46	2,18	2,14

$\langle SHGC \rangle_D$ ve belli güneş geliş açıları için $SHGC(\theta)$ değerleri cam adedi ve kalınlığına göre Tablo 3.6'dan bulunur. Mevcut güneş geliş açısına göre $SHGC(\theta)$ değerini bulmak için interpolasyon yapılır.

Tablo 3.6: Belli cam tipleri için güneş ısı kazanç katsayıları (Owen 2009).

	Güneş Geliş Açısı [°]							
	0	40	50	60	70	80	90	
Cam Tipi	**SHGC**							**$\langle SHGC \rangle_D$**
3 mm Tek Cam	0,86	0,84	0,82	0,78	0,67	0,42	0	0,78
3 mm Çift Cam	0,76	0,74	0,71	0,64	0,5	0,26	0	0,66
3 mm Üçlü Cam	0,68	0,62	0,62	0,54	0,39	0,18	0	0,57

Direkt güneş ışınımından kaynaklanan saatlik soğutma yükü, (3.41) denkleminde bulunan ısı kazancı ve bina yapısına göre Tablo A.4'den seçilen güneşe bağlı ışınım zaman faktörleri kullanılarak (3.39) denklemi ile hesaplanır.

(3.40) denklemi ile bulunan iletimsel ısı kazancı ile (3.41) denklemi ile bulunan yayılan güneş ısı kazancı toplanarak Tablo 3.7'de verilen yüzdelere göre

taşınımsal ve ışınımsal kısımlara ayrılır. Taşınımsal kısım direkt olarak soğutma yüküne dönüşür. Işınımsal ısı kazancı ve bina yapısına göre Tablo A.3'den seçilen güneşe bağlı olmayan ışınım zaman faktörleri kullanılarak (3.39) denklemi ile iletim ve yayılmadan kaynaklanan saatlik ışınımsal soğutma yükü hesaplanır.

Tablo 3.7: Pencere ve camlı kapı için taşınımsal ve ışınımsal ısı kazancı yüzdeleri (Owen 2009).

Isı Kazancı	Taşınımsal Kısım	Işınımsal Kısım
İletim+Yayılan Güneş	%67 (SHGC $>$ 0,5)	%33 (SHGC $>$ 0,5)
	%54 (SHGC $<$ 0,5)	%46 (SHGC $<$ 0,5)
Direkt Güneş	%0	%100

Son olarak direkt güneş ışınımından kaynaklanan soğutma yükü, iletim ve yayılan güneş ısı kazancından hesaplanan taşınımsal soğutma yükü ile ışınım zaman serisi uygulanarak hesaplanan ışınımsal soğutma yükü toplanarak saatlik soğutma yükü hesaplanır. Bu işlemler her saat için tekrarlanır.

3.2.4 Aydınlatmadan Kaynaklanan Soğutma Yükü Hesabı

Aydınlatmadan kaynaklanan soğutma yükünün hesaplanmasında pratik bir yöntem olarak mahal türüne göre Tablo 3.8'de verilen *birim kullanım alanı başına aydınlatma gücü (LPD*) kullanılabilir. Seçilen LPD değeri mahal kullanım alanı ile çarpılarak aydınlatmadan kaynaklanan saatlik ıs kazancı hesaplanır.

Tablo 3.8: Mahal türüne göre birim kullanım alanı başına aydınlatma güçleri (Owen 2009).

Mahal Türü	LPD [W/m^2]
Ofis	12
Konferans/toplantı/çok amaçlı salon	14
Sergi alanı	14
Derslik/eğitim alanı	15
Fabrika (tavan yüksekliği<7,6 m)	13
Fabrika (tavan yüksekliği ≥7,6 m)	18
Fabrika (detaylı üretim)	23
Ambar (ince malzeme)	15
Ambar (orta/iri malzeme)	10
Spor salonu (ring sporları)	29
Spor salonu (kort sporları)	25
Spor salonu (kapalı saha)	15

Hesaplanan saatlik ısı kazancı, lamba türüne göre Tablo 3.9'da verilen yüzdeler kullanılarak taşınımsal ve ışınımsal kısımlara ayrılır.

Tablo 3.9: Lamba türüne göre taşınımsal ve ışınımsal ısı kazancı yüzdeleri (Owen 2009).

Lamba Türü	Taşınımsal Kısım	Işınımsal Kısım
Gömme floresan (merceksiz)	%42	%58
Gömme floresan (mercekli)	%33	%67
Sarkık floresan	%46	%54
Sarkık kompakt floresan	%2	%98
Sarkık akkor	%2	%98

Taşınımsal ısı kazancı direkt olarak soğutma yüküne dönüşür. Işınımsal ısı kazancı ve bina yapısına göre Tablo A.3'den seçilen güneşe bağlı olmayan ışınım zaman faktörleri kullanılarak (3.39) denklemi ile saatlik ışınımsal soğutma yükü hesaplanır. Taşınımsal ve ışınımsal soğutma yükleri toplanarak aydınlatma kaynaklı saatlik toplam soğutma yükü hesaplanır. Bu işlemler her saat için tekrarlanır.

3.2.5 İnsanlardan Kaynaklanan Soğutma Yükü Hesabı

İnsanlardan kaynaklanan soğutma yükünün hesaplanmasında yapılan faaliyet türüne göre Tablo 3.10'da verilen insan başına duyulur ısı kazancı değerleri kullanılabilir. Seçilen değer içeride bulunan insan sayısı ile çarpılarak insanlardan kaynaklanan saatlik ısı kazancı hesaplanır.

Tablo 3.10: Faaliyet türüne göre insan başına düşen duyulur ısı kazançları (Owen 2009).

Faaliyet Türü	İnsan Başına Duyulur Isı Kazancı [W]
Oturma	65
Oturma, çok hafif iş	70
Kısmen aktif ofis işi	75
Yürüme, hafif iş, ayakta durma	75
Hafif tezgah işi	80
4,8 km/h hızla yürüme, hafif makine işi	110
Ağır iş	170
Ağır makine işi	185
Atletizm faaliyeti	210

Hesaplanan saatlik ısı kazancı, %40 taşınımsal ve %60 ışınımsal olmak üzere ayrılır. Taşınımsal ısı kazancı direkt olarak soğutma yüküne dönüşür. Işınımsal ısı kazancı ve bina yapısına göre Tablo A.3'den seçilen güneşe bağlı olmayan ışınım zaman faktörleri kullanılarak (3.39) denklemi ile saatlik ışınımsal soğutma yükü hesaplanır. Taşınımsal ve ışınımsal soğutma yükleri toplanarak insanlardan kaynaklanan saatlik toplam duyulur soğutma yükü hesaplanır. Bu işlemler her saat için tekrarlanır.

3.2.6 Cihazlardan Kaynaklanan Soğutma Yükü Hesabı

Cihazlardan kaynaklanan soğutma yükünün hesabında motor ve tahrik edilen ünitenin konumuna göre üç farklı durum dikkate alınır. Motor ve tahrik edilen ünitenin ikisinin de içeride bulunması durumunda saatlik ısı kazancı (3.43) denklemi ile hesaplanır.

$$q_{em} = P / \eta \tag{3.43}$$

Motorun dışarıda, tahrik edilen ünitenin içeride bulunması durumunda saatlik ısı kazancı (3.44) denklemi ile hesaplanır.

$$q_{em} = P \tag{3.44}$$

Motorun içeride, tahrik edilen ünitenin dışarıda bulunması durumunda saatlik ısı kazancı (3.45) denklemi ile hesaplanır.

$$q_{em} = P\left(\frac{1-\eta}{\eta}\right) \tag{3.45}$$

Hesaplanan ısı kazancı eşit olarak ışınımsal ve taşınımsal kısımlara ayrılır. Taşınımsal ısı kazancı direkt olarak soğutma yüküne dönüşür. Işınımsal ısı kazancı ve bina yapısına göre Tablo A.3'den seçilen güneşe bağlı olmayan ışınım zaman faktörleri kullanılarak (3.39) denklemi ile saatlik ışınımsal soğutma yükü hesaplanır. Taşınımsal ve ışınımsal soğutma yükleri toplanarak makinelerden kaynaklanan saatlik toplam soğutma yükü hesaplanır. Bu işlemler her saat için tekrarlanır.

3.2.7 Maksimum Soğutma Yükü Hesabı

4 adet cephede bulunan duvar, pencere ve kapı ile çatı, aydınlatma, insanlar ve cihazlardan kaynaklanan soğutma yükleri her saat için toplanarak saatlik toplam duyulur soğutma yükü hesaplanır. 24 saatlik periyod içerisindeki maksimum değer pik soğutma yükü olarak, bu yükün oluştuğu saat ise pik saat olarak belirlenir.

3.3 Buharlaşmalı Soğutucu Kapasite Hesabı

Mahal soğutma yükünü karşılamak ve konfor koşullarını sağlayabilmek için soğutucu cihazların içeri beslemesi gereken toplam hava debisinin hesabında *hava değişimi yöntemi* ve *duyulur ısı uzaklaştırma yöntemi* olmak üzere yaygın olarak kullanılan iki yöntem bulunmaktadır (Bhatia 2012).

Her iki yöntemin de kullanımında soğutucu cihazın hava çıkış sıcaklığının belirlenmesine ihtiyaç vardır. Soğutucu hava çıkış sıcaklığı dış ortam sıcaklığı ($t_{dış}$), doyma verimi (ε) ve yaş termometre sıcaklığına (t*) göre (3.46) denklemi ile hesaplanır.

$$t_{çıkış} = t_{dış} - \left[\varepsilon - \left(t_{dış} - t^* \right) \right] \tag{3.46}$$

3.3.1 Hava Değişimi Yöntemi

Bu yöntem, buharlaşmalı soğutucuların hava debisinin hesaplanmasında pratik bir yaklaşımdır ve iki adımdan oluşur. İlk olarak (3.46) denklemi ile cihaz hava çıkış sıcaklığı belirlenir. Daha sonra *mahal şartlandırılmadığında* iç ısı kaynakları ve güneş ışınımından dolayı içeride oluşacak sıcaklık farkı belirlenir. Bu iki sıcaklığa göre saatlik hava değişimi Tablo 3.11'den bulunur. Hava değişimi, mahal soğutma hacmi ile çarpılarak soğutma için gereken toplam hava debisi hesaplanır.

Tablo 3.11: Soğutucu çıkış sıcaklığı ve ortam sıcaklık farkına göre saatlik hava değişimi değerleri (Bhatia 2012).

Soğutucu Çıkış Sıcaklığı [°C]	Şartlandırılmamış Durumda Sıcaklık Farkı [°C]	Saatlik Hava Değişimi
25,5 üstü	11,1	30-60
24,4-25,5	8,3-11,1	20-40
23,3-25,5	5,5-11,1	15-30
22,2-23,3	2,7-5,5	12-20
22,2 altı	5,5 altı	10-15

3.3.2 Duyulur Isı Uzaklaştırma Yöntemi

Bu yöntem, hava değişimi yönteminin aksine yükseltiye bağlı olarak havanın yoğunluk oranını dikkate alır ve bu nedenle daha hassastır. Soğutma için gereken toplam hava debisi iç ortam sıcaklığı ($t_{iç}$), soğutucu çıkış sıcaklığı ($t_{çıkış}$), toplam duyulur soğutma yükü (Q) ve hava yoğunluk oranına göre (3.47) denklemi ile hesaplanır. Bu çalışmada duyulur ısı uzaklaştırma yöntemi kullanılmıştır.

$$\dot{V}_{hava} = 2,9281 \frac{Q_{toplam}}{(t_{iç} - t_{çıkış}) \times \text{yoğunluk oranı}} \tag{3.47}$$

Hava yoğunluk oranı, soğutucu çıkış sıcaklığı ve yükseltiye göre Tablo 3.12'den bulunur.

Tablo 3.12: Soğutucu çıkış sıcaklığı ve yükseltiye göre hava yoğunluk oranları (Bhatia 2012).

Soğutucu Çıkış Sıcaklığı [°C]	Yükselti [m]								
	0	304,8	609,6	914,4	1219,2	1524	1828,8	2133,6	2434,8
20	1	0,97	0,93	0,9	0,87	0,84	0,8	0,77	0,75
21,1	1	0,96	0,93	0,9	0,86	0,83	0,8	0,77	0,74
22,2	1	0,96	0,93	0,89	0,86	0,83	0,8	0,77	0,74
23,3	0,99	0,96	0,92	0,89	0,86	0,83	0,8	0,77	0,74
24,4	0,99	0,95	0,92	0,89	0,85	0,82	0,79	0,76	0,73
25,5	0,99	0,95	0,92	0,88	0,85	0,82	0,79	0,76	0,73
26,6	0,98	0,95	0,91	0,88	0,85	0,82	0,79	0,76	0,73

3.3.3 Soğutucu Cihaz Adedi ve Su Debisi Hesabı

Soğutma için gereken toplam hava debisi hesaplandıktan sonra bu değer, üretici kataloglarında bulunan cihaz hava debisine bölünerek gereken toplam soğutucu adedi bulunur. Cihaz başına beslenmesi gereken su debisi (3.48) denklemi ile hesaplanarak cihaz adedi ile çarpılır ve sisteme beslenmesi gereken toplam su debisi bulunur.

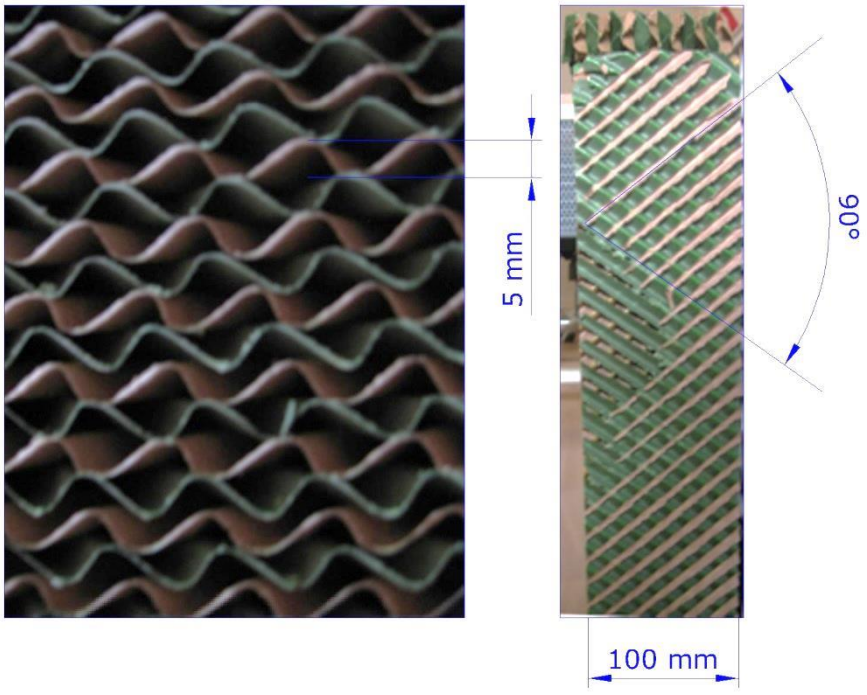
$$\dot{V}_{su} = \frac{\dot{V}_{hava}(w_{çıkış} - w_{giriş})v_{su}}{v_{hava}} \tag{3.48}$$

4. DENEYSEL YÖNTEM VE DENEYSEL ÇALIŞMALAR

Buharlaşmalı soğutucuların performansı doyma verimi ile belirlenir. Doyma verimi soğutucu ped malzemesine, boyutlarına, kalınlığına, diş aralığına, açısına ve üzerinden geçen havanın hızına bağlı olarak değişir. Buharlaşmalı soğutucuların doyma verimi giriş ve çıkıştaki kuru termometre sıcaklıkları ile yaş termometre sıcaklığına göre (4.1) denklemi ile hesaplanır.

$$\varepsilon = \frac{t_{giriş} - t_{çıkış}}{t_{giriş} - t^{*}} \tag{4.1}$$

İki ayrı çalışma yapılarak piyasada yaygın olarak kullanılan Greenpad markasına ait 5090/100 kağıt soğutma pedinin doyma verimi belirlenmiştir. Pedin kalınlığı 100 mm, diş genişliği 5 mm, açısı 90º'dir (Şekil 4.1). İlk olarak Yaşam Mühendislik Ltd. Şti. Denizli fabrikasında kurulu bulunan deney düzeneği üzerinde ped giriş ve çıkışına ait sıcaklık, bağıl nem ve hava hızı değerleri ölçülmüştür. Daha sonra aynı fabrikada bir direkt buharlaşmalı soğutucu çalıştırılarak saha ölçümü yapılmıştır.



Şekil 4.1: Greenpad 5090/100 soğutma pedi ölçüleri.

4.1 Deney Düzeneğinde Yapılan Çalışmalar

Yaşam Mühendislik Ltd. Şti. Denizli fabrikasında kurulu bulunan deney düzeneği iklimlendirme santrali, test odası ve ölçüm istasyonu olmak üzere üç bölümden oluşmaktadır.

İklimlendirme ünitesi İMEKSAN İKS-25 model olup ısıtma, soğutma ve nemlendirme kısımlarından oluşmaktadır ve 15-45 °C sıcaklık ile %10-60 arasında bağıl nem değerlerini üniform biçimde sağlayabilmektedir (Şekil 4.2). Cihaz hava çıkış debisi 5000 m^3/h, basıncı 936 Pa, fan hızı 2999 min^{-1}'dir. Sisteme nem beslemesi 30 kg/h debide buhar veren 22,57 kW'lık Devatec ElectroVap MC2 buharlı nemlendirici ile sağlanmaktadır (Şekil 4.3).

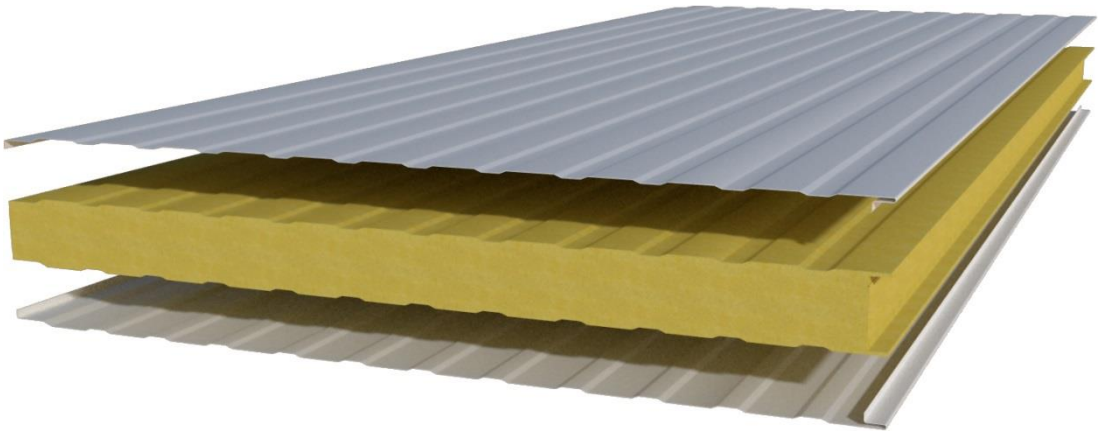
Şekil 4.2: İklimlendirme ünitesi.

Test odası 5000 mm x 3000 mm x 6000 mm ölçülerinde, 0,5 mm kalınlığında galvaniz saclar arası 80 mm poliüretan izolasyon köpüğünden oluşan sandviç paneller ile kurulmuştur (Şekil 4.4). Poliüretan köpüğün ısı iletim katsayısı 0,024 W/mK'dir. Bu sayede oluşacak ısı kaybının önüne geçilerek, direkt buharlaşmalı

soğutucuları test emek için farklı dış ortam koşullarını temsil eden üniform sıcaklık ve bağıl nem değerleri garanti altına alınabilmektedir. Şekil 4.5 sandviç panelin yapısını göstermektedir.

Şekil 4.3: Buharlı nemlendirici.

Şekil 4.4: Test odası.

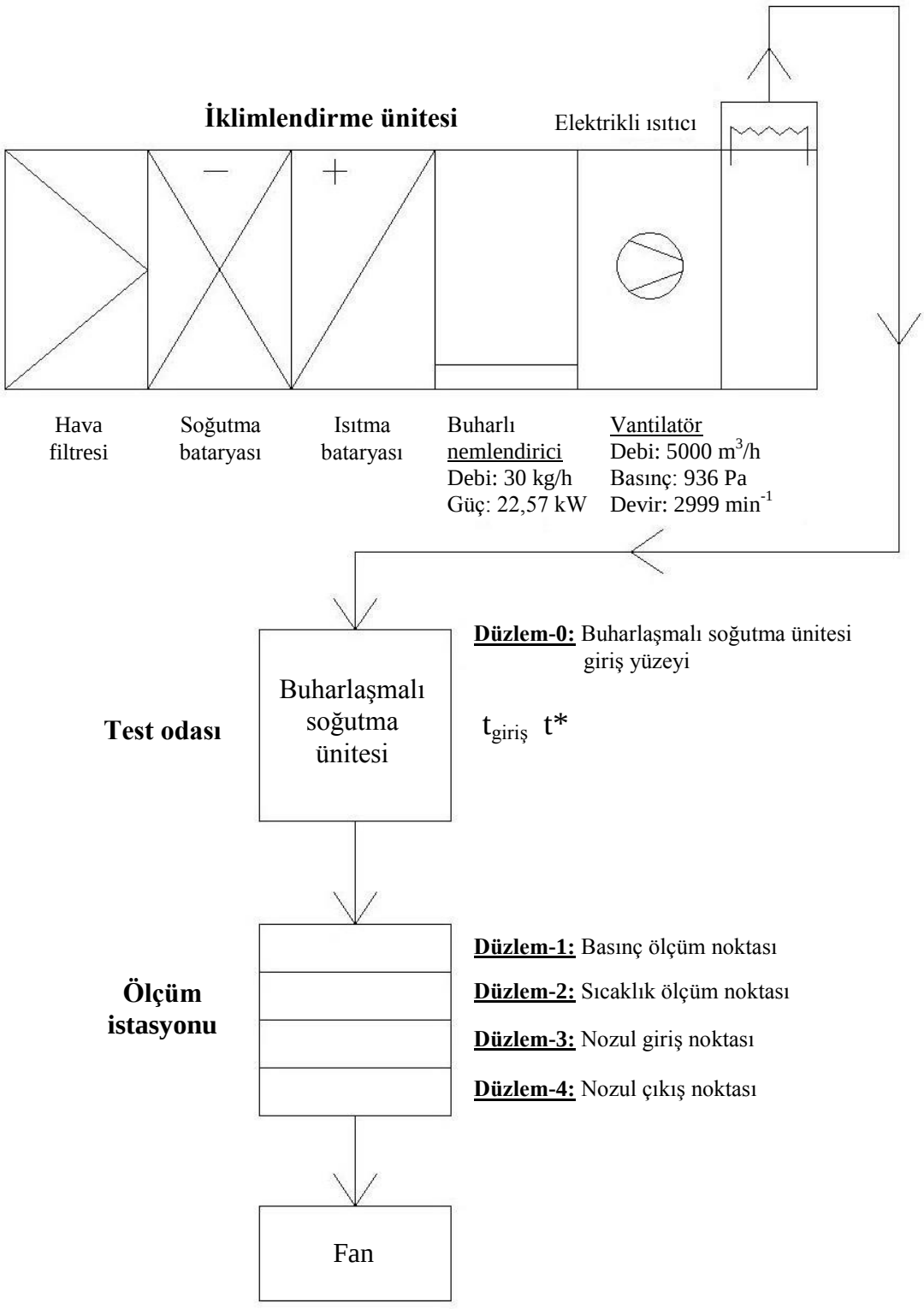


Şekil 4.5: Poliüretan köpük dolgulu sandviç panel (url-3).

Girişteki sıcaklık ve bağıl nem değerlerini biriktirip transfer etmek için NTC dış sensörlü iki adet Testo 6224 radyo prob kullanılmıştır. Probların sıcaklık ölçüm aralığı -20 ila +70 °C, hassasiyeti ±0,2 °C'dir. Aynı tür problar çıkıştaki sıcaklık ve bağıl nem değerlerinin ölçümü için de kullanılmıştır. Ayrıca kontrol amaçlı olarak giriş havasının kuru ve yaş termometre sıcaklığını ölçen cıvalı termometreler kullanılmıştır. Şekil 4.6 ölçüm istasyonunu göstermektedir.

Şekil 4.6: Ölçüm istasyonu.

Şekil 4.7 deney düzeneğini şematik olarak göstermektedir.



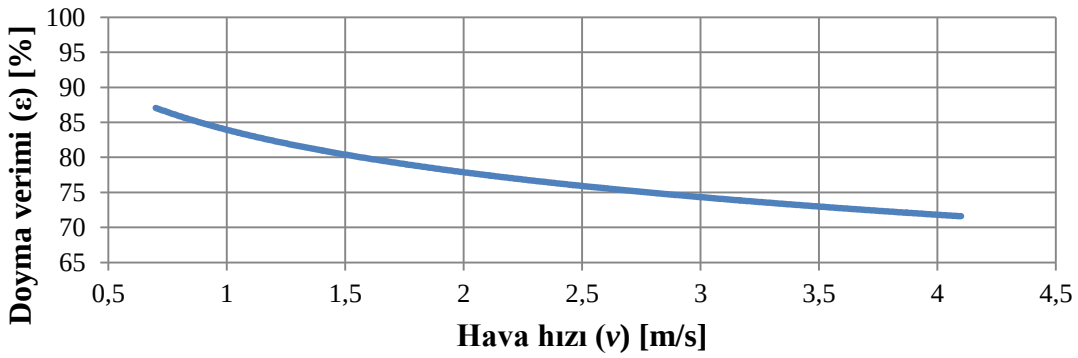
Şekil 4.7: Deney düzeneği şematik gösterimi.

Deney düzeneğinde farklı hava hızlarının doyma verimine etkisini incelemek için ped girişinde kuru termometre sıcaklığı 38 ºC, bağıl nem %20 civarında tutularak hava hızı değiştirilmiştir. Deney sonuçları Tablo 4.1'de verilmiştir.

Tablo 4.1: Greenpad 5090/100 soğutma pedi için deney sonuçları.

$t_{giriş}$ [ºC]	$\phi_{giriş}$ [%]	$t_{çıkış}$ [ºC]	$\phi_{çıkış}$ [%]	t^* [ºC]	v [m/s]	ε [%]
38,1	18,0	22,7	80,7	20,0	0,7	84,9
37,4	18,8	22,6	80,5	19,8	0,8	84,0
37,5	18,8	22,4	80,5	19,9	0,9	85,6
37,6	18,8	22,2	79,5	19,9	1,0	87,1
37,8	20,1	22,1	80,0	19,2	1,4	82,5
37,8	18,6	23,3	73,9	20,0	1,6	81,2
37,2	19,4	23,6	73,9	19,9	1,7	78,8
37,9	18,7	24,1	72,5	20,1	1,8	77,5
37,9	18,7	24,2	70,6	20,1	1,9	77,3
32,2	19,1	23,7	71,0	19,8	2,0	77,6
32,7	18,7	23,9	70,7	20,0	2,1	78,1
36,4	19,9	23,4	71,4	19,5	2,2	76,9
38,7	18,5	25,0	67,5	20,5	2,3	75,3
36,9	19,2	23,6	69,6	19,6	2,4	76,7
38,1	18,1	24,1	68,5	20,0	2,5	77,6
37,7	18,7	24,3	67,3	20,0	2,6	75,6
37,4	19,0	24,3	67,5	19,9	2,7	75,1
36,2	19,8	23,8	67,9	19,3	2,8	73,4
39,7	16,7	24,6	66,3	20,5	2,9	78,6
39,1	17,3	26	61	20,4	3,6	70,1
36,1	20,2	24,6	63,8	19,4	4,0	69,2
35,9	20,1	23,6	65,2	19,2	4,1	73,7

Şekil 4.8'de doyma veriminin hava hızı ile ters orantılı olarak değiştiği görülmektedir.



Şekil 4.8: Doyma veriminin hıza bağlı değişimi.

4.2 Saha Ölçüm Çalışması

Soğutma pedinin gerçek durumda doyma verimini belirlemek için 4 Haziran 2015 tarihinde, Yaşam Mühendislik Ltd. Şti. Denizli fabrikasında kurulu bulunan Alindair 20S direkt buharlaşmalı soğutucu çalıştırılarak saha ölçümü yapılmıştır. Soğutucunun hava debisi 20000 m^3/h olup içerisinde 4 adet 885 mm x 980 mm x 100 mm ölçülerinde Greenpad 5090/100 kağıt soğutma pedi bulunmaktadır (Şekil 4.9). Toplam ped alanı 3,47 m^2, hava hızı 1,6 m/s'dir. Soğutucuya ait katalog bilgileri Tablo B.1'de mevcuttur. Soğutucu rejime girdikten sonra Testo 410-2 anemometre ile hava giriş ve çıkışındaki kuru termometre sıcaklığı ile bağıl nem değerleri ölçülmüştür. Ölçüm cihazının sıcaklık ölçüm aralığı -10 ila +50 °C, hassasiyeti ±0,5 °C'dir. Bağıl nem ölçüm aralığı %0 ila 100, hassasiyeti ±%2,5'dir.

Şekil 4.9: Alindair 20S direkt buharlaşmalı soğutucu (url-4).

Soğutucunun hava girişinde kuru termometre sıcaklığı 26 °C, bağıl nem %35, hava çıkışında kuru termometre sıcaklığı 17,5 °C, bağıl nem %87,5 olarak ölçülmüştür. İterasyon yöntemi kullanılarak (bkz. Bölüm 3.1) yaş termometre sıcaklığı 15,9 °C, (4.1) denklemi kullanılarak doyma verimi %84,1 olarak belirlenmiştir. Belirlenen %84,1 doyma verimi ile 1,6 m/s hava hızı için deney düzeneğinden elde edilen %81,2 doyma verimi arasında %3,5 fark tespit edilmiştir. Bu fark, ölçüm cihazlarının hassasiyetinden kaynaklanmış olmakla birlikte değerler tutarlılık göstermiştir.

5. PROGRAMLAMA VE YAZILIM ÇALIŞMASI

Visual C# programlama dilinde bina soğutma yükünü hesaplayan ve bu soğutma yükünü karşılayacak direkt buharlaşmalı soğutucunun hava debisi, su debisi ve adedini belirleyen bir program yazılmıştır. Program, "*BuharlasmaliSogutucuHesapPrograms.exe*", "*fonksiyon.dll*" ve "*data.mdb*" olmak üzere üç dosyadan oluşmaktadır. "BuharlasmaliSogutucuHesapPrograms.exe" dosyası ana program dosyasıdır. Psikrometri, güneş ışınımı ve soğutma yükü hesap formüllerini içeren fonksiyonlar "fonksiyon.dll" dosyasında saklanmıştır ve buradan çağrılmaktadır. Programda veri alışverişi "data.mdb" veritabanı dosyası üzerinden yapılmaktadır. Programa ait kaynak kodları EK C'de verilmiştir.

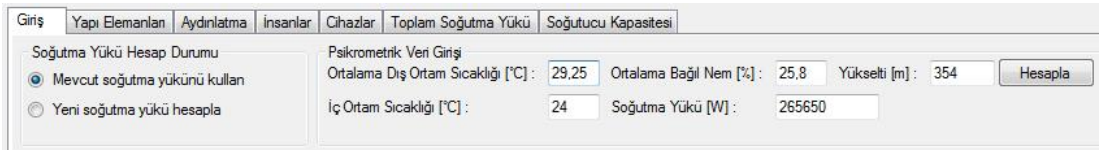
Hazırlanan programda kullanıcı dostu sekmeli arayüz oluşturulmuştur. "*Giriş*", "*Yapı Elemanları*", "*Aydınlatma*", "*İnsanlar*", "*Cihazlar*", "*Toplam Soğutma Yükü*" ve "*Soğutucu Kapasitesi*" olmak üzere 7 adet sekme mevcuttur. Adım adım ilerlemeyi garanti altına almak için her adımda "*Hesapla*" butonu yerleştirilmiş olup bu butonlara basılıp basılmadığını kontrol eden koşul fonksiyonları oluşturulmuştur. "Hesapla" butonuna basılmadan sonraki adımlara geçiş engellenmiştir.

Programın kullanımını göstermek için Denizli ilinde model bir sanayi binası tasarlanmıştır. Model binanın bulunduğu yükselti 354 m, boylamı 29,1º ve enlemi 37,76º'dir.

5.1 Giriş Sekmesi

Giriş sekmesi, program başlatıldığında kullanıcıyı karşılayan ekrandır. Bu ekranda iki adet seçenek mevcuttur. Eğer kullanıcının elinde mevcut bir soğutma yükü var ise "*Mevcut soğutma yükünü kullan*" seçeneği işaretlenerek yeniden soğutma yükü hesabına gerek kalmadan ortalama dış sıcaklık, bağıl nem, yükselti, istenen iç ortam sıcaklığı ve mevcut soğutma yükü değerleri girilip doğrudan kapasite hesap sekmesine geçiş yapılabilir (Şekil 5.1). Soğutma yükü hesabı

yapılmak isteniyorsa "*Yeni soğutma yükü hesapla*" seçeneği işaretlenerek ışınım zaman serileri yöntemiyle bina soğutma yükünü hesaplamak için iklim verisi girişi yapılabilir (Şekil 5.2).



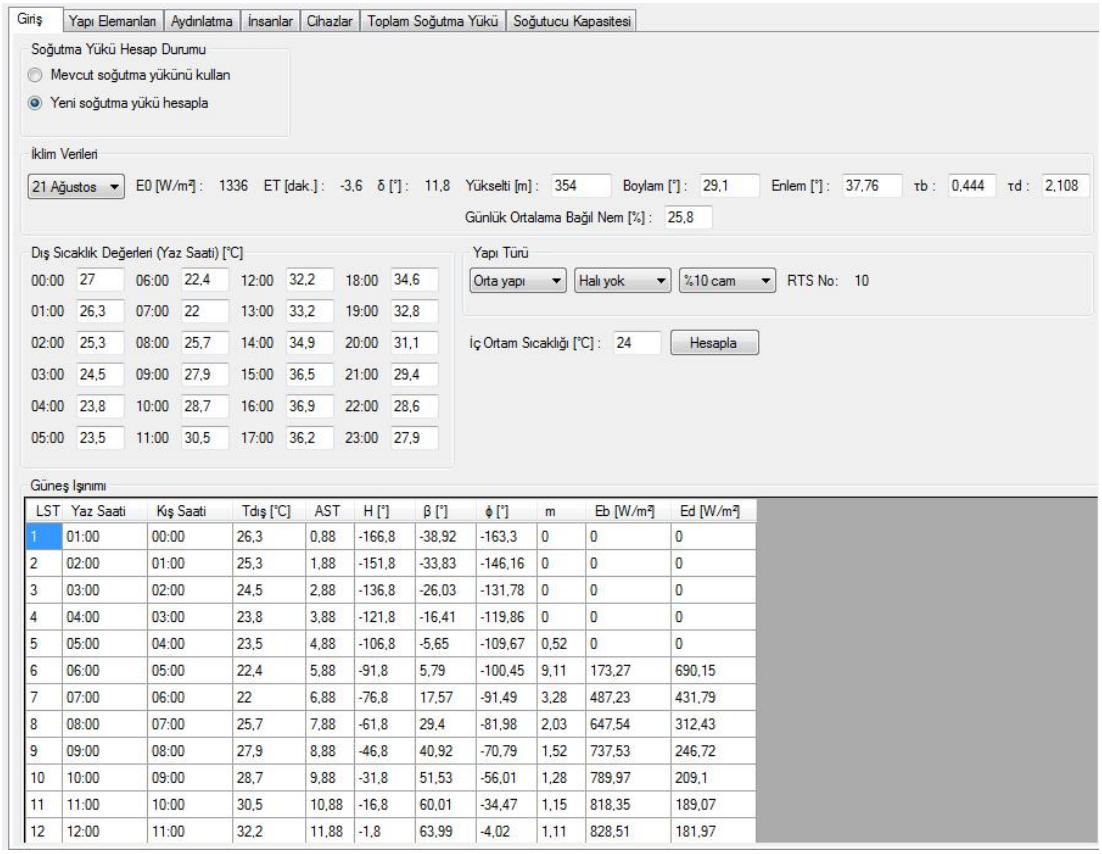
Şekil 5.1: Mevcut soğutma yükü giriş ekranı.

Soğutma yükü hesabı için öncelikle listeden tarih seçilmelidir. Seçilen tarihe göre dünya dışı ışınım akısı (E_0), zaman eşitliği (ET) ve sapma açısı (δ) değerleri ekranda gösterilir. Yükselti, boylam, enlem, optik derinlik ve bağıl nem değerleri girildikten sonra saatlik sıcaklık değerleri girilir. Daha sonra ışınım zaman faktörlerini belirlemek için yapı türü, ortamda halı olup olmaması ve cam alanının yüzdesi ilgili listelerden seçilir. Son olarak istenen iç ortam sıcaklığı girilir.

Model binanın soğutma yükünün hesaplanmasında Meteoroloji Genel Müdürlüğü TÜMAS veritabanından elde edilen Denizli iline ait 2014 yılı 21 Ağustos günü için saatlik sıcaklık ve günlük ortalama bağıl nem değerleri kullanılmıştır (Tablo D.1). Optik derinlik değerleri 21 Ağustos günü için Tablo 3.2'den alınmıştır. Bina 24 ºC sıcaklıkta tutulmak istenmektedir.

"Hesapla" butonuna basıldığında 24 saat için hesaplanan belirgin güneş zamanı (AST), saat açısı (H), güneş irtifa açısı (β), güneş azimut açısı (ϕ), bağıl hava kütlesi (m), direkt güneş ışınımı (E_b) ve yayılan güneş ışınımı (E_d) değerleri tablo halinde ekranda gösterilir.

Model binaya ait hesap verileri incelendiğinde gece saatlerinde bağıl hava kütlesi ve güneş ışınımı değerlerinin sıfır olduğu görülmektedir.



Şekil 5.2: İklim verisi giriş ekranı.

5.2 Yapı Elemanları Sekmesi

Yapı elemanları sekmesinde 4 adet cepheye ve çatıya ait bilgiler girilerek bu elemanlardan kaynaklanan soğutma yükleri hesaplanır. Bu çalışmada zeminin toprak ile temasta olduğu kabul edildiğinden hesaplamalara dahil edilmemiştir. Cephe seçimi ağaç yapısı üzerinden yapılmaktadır. Her ekranda "Hesapla" butonuna basılmadan diğer cephe ekranlarına geçiş engellenmiştir.

Cephe ekranında ilk olarak pusula üzerinden yön seçimi yapılır. Daha sonra duvara ait en ve boy ölçüleri girilerek ilgili listelerden tipi seçilir ve rengi işaretlenir. Seçilen duvar tipinin yapı elemanları ve ısıl özellikleri ekranda gösterilir. Cephede pencere ve kapı varsa bunlara ait adet, en ve boy ölçüleri girilerek ilgili listelerden tipi seçilir, yoksa adet kutusu boş bırakılır.

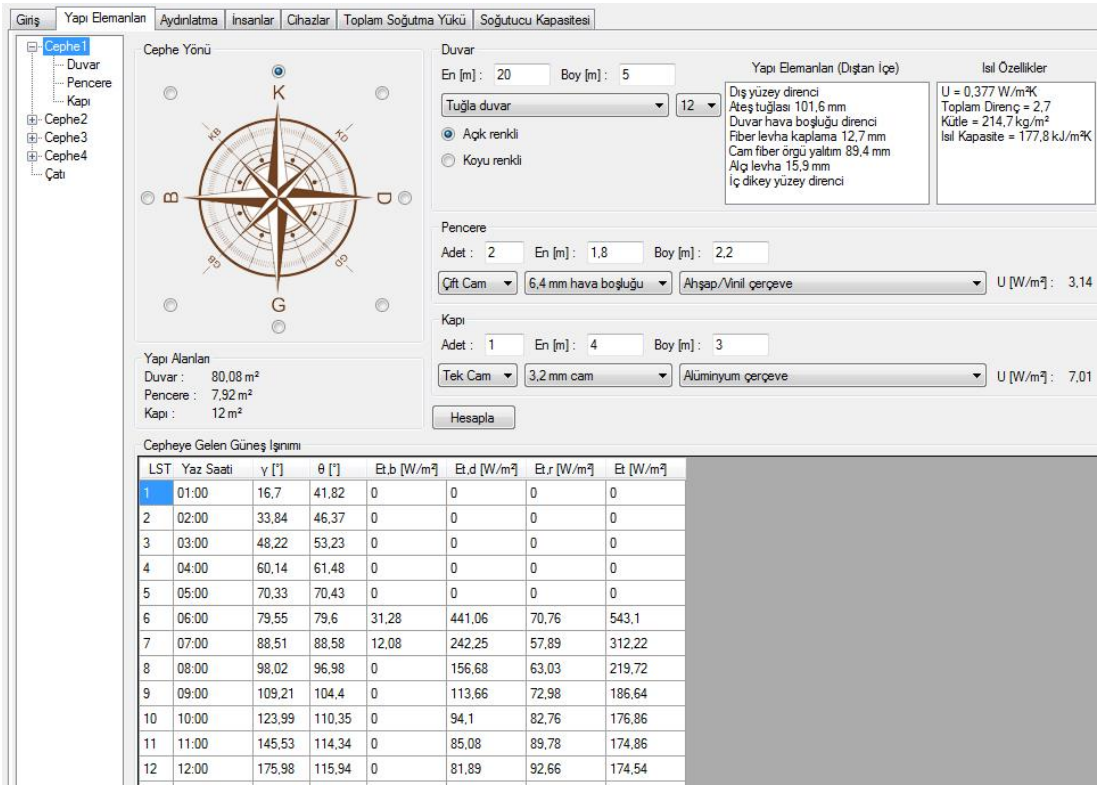
Model binada açık renkli tip-12 tuğla duvar (bkz. Tablo A.5), açık renkli tip-3 çerçeve çatı (bkz. Tablo A.6), çift camlı, 6,4 mm hava boşluklu, vinil çerçeveli

pencere (bkz. Tablo 3.5) ve 3,2 mm tek camlı, alüminyum çerçeveli kapı (bkz. Tablo 3.5) seçilmiştir. Binaya ait cephe yönleri ve yapı ölçüleri Tablo 5.1'de verilmiştir.

Tablo 5.1: Model binaya ait cephe yönleri ve yapı ölçüleri.

		Duvar		Pencere			Kapı		
Cephe	**Yön**	**En [m]**	**Boy [m]**	**Adet**	**En [m]**	**Boy [m]**	**Adet**	**En [m]**	**Boy [m]**
1	K	20	5	2	1,8	2,2	1	4	3
2	G	20	5	1	3,6	0,75	2	1,8	2,2
3	B	30	5	3	3,6	1,3	-	-	-
4	D	30	5	3	3,6	1,3	-	-	-
		Çatı							
Çatı	-	20	30						

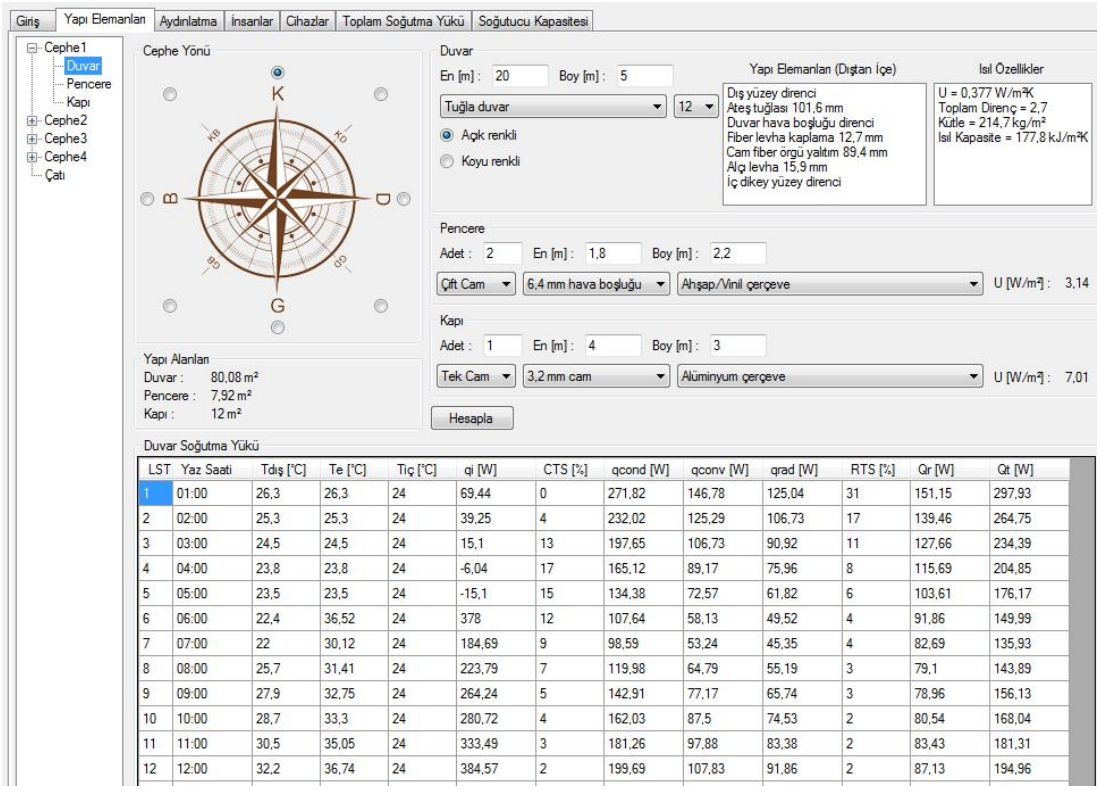
"Hesapla" butonuna basıldığında net duvar, pencere ve kapı alanları hesaplanarak ekranda gösterilir ve 24 saat için hesaplanan yüzey-güneş azimut açısı (γ), geliş açısı (θ), yüzeye gelen direkt güneş ışınımı ($E_{t,b}$), yüzeye yayılan güneş ışınımı ($E_{t,d}$) ve yerden yüzeye yansıyan güneş ışınımı ($E_{t,r}$) değerleri tablo halinde ekranda gösterilir (Şekil 5.3).



Şekil 5.3: Cephe ekranı.

Model binaya ait cephelere gelen güneş ışınımı hesap verileri incelendiğinde gece saatlerinde her üç ışınım bileşeninin de sıfır olduğu, cephenin gölgede kaldığı saatlerde ise sadece direkt bileşenin sıfır olduğu görülmektedir.

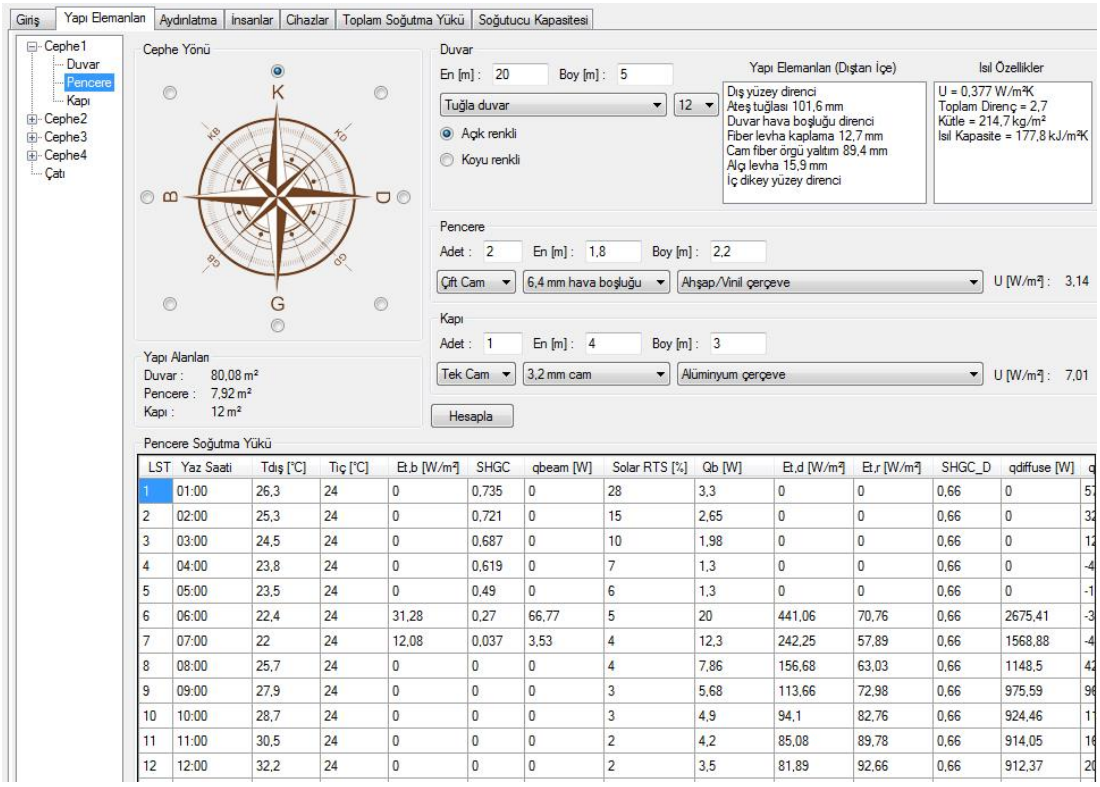
Ağaç yapısından ilgili cephenin altındaki "Duvar" kısmına tıklanarak 24 saat için hesaplanan eşdeğer sıcaklık, ısı girişi, iletim zaman faktörü, iletimsel ısı kazancı, taşınımsal ısı kazancı, ışınımsal ısı kazancı, ışınım zaman faktörü, ışınımsal soğutma yükü ve toplam soğutma yükü tablo halinde ekranda gösterilir (Şekil 5.4).



LST	Yaz Saati	Tdış [°C]	Te [°C]	Tiç [°C]	qi [W]	CTS [%]	qcond [W]	qconv [W]	qrad [W]	RTS [%]	Qr [W]	Qt [W]
1	01:00	26,3	26,3	24	69,44	0	271,82	146,78	125,04	31	151,15	297,93
2	02:00	25,3	25,3	24	39,25	4	232,02	125,29	106,73	17	139,46	264,75
3	03:00	24,5	24,5	24	15,1	13	197,65	106,73	90,92	11	127,66	234,39
4	04:00	23,8	23,8	24	-6,04	17	165,12	89,17	75,96	8	115,69	204,85
5	05:00	23,5	23,5	24	-15,1	15	134,38	72,57	61,82	6	103,61	176,17
6	06:00	22,4	36,52	24	378	12	107,64	58,13	49,52	4	91,86	149,99
7	07:00	22	30,12	24	184,69	9	98,59	53,24	45,35	4	82,69	135,93
8	08:00	25,7	31,41	24	223,79	7	119,98	64,79	55,19	3	79,1	143,89
9	09:00	27,9	32,75	24	264,24	5	142,91	77,17	65,74	3	78,96	156,13
10	10:00	28,7	33,3	24	280,72	4	162,03	87,5	74,53	2	80,54	168,04
11	11:00	30,5	35,05	24	333,49	3	181,26	97,88	83,38	2	83,43	181,31
12	12:00	32,2	36,74	24	384,57	2	199,69	107,83	91,86	2	87,13	194,96

Şekil 5.4: Duvar ekranı.

İlgili cepheye ait "Pencere" kısmına tıklandığında pencerelere ait 24 saat için hesaplanan güneş ısı kazanç katsayısı, direkt güneş ısı kazancı, güneşe bağlı ışınım zaman faktörü, direkt güneş soğutma yükü, yayılan güneş ısı kazanç katsayısı, yayılan güneş ısı kazancı, iletim ısı kazancı, iletim ve yayılan güneş ısı kazançlarının taşınımsal ve ışınımsal kısımları, güneşe bağlı olmayan ışınım zaman faktörü, ışınımsal soğutma yükü ve toplam soğutma yükü tablo halinde ekranda gösterilir (Şekil 5.5).



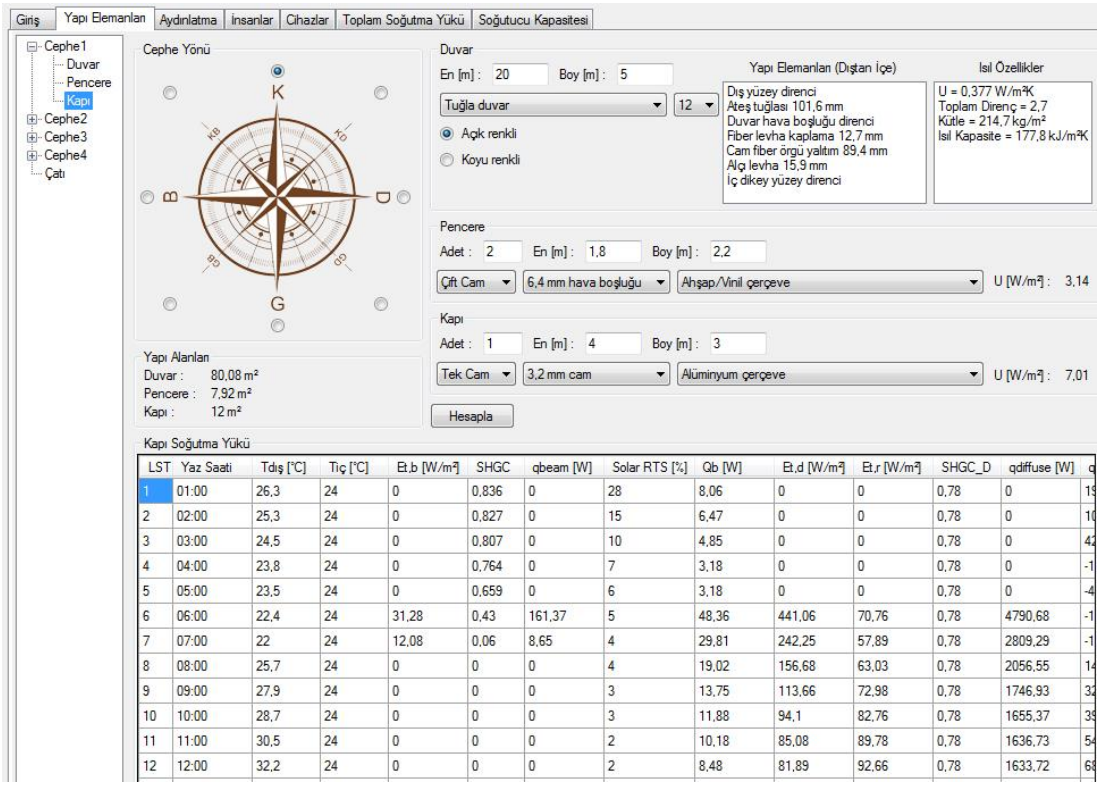
LST	Yaz Saati	Tdış [°C]	Tiç [°C]	Et,b [W/m²]	SHGC	qbeam [W]	Solar RTS [%]	Qb [W]	Et,d [W/m²]	Et,r [W/m²]	SHGC_D	qdiffuse [W]	q
1	01:00	26,3	24	0	0,735	0	28	3,3	0	0	0,66	0	57
2	02:00	25,3	24	0	0,721	0	15	2,65	0	0	0,66	0	32
3	03:00	24,5	24	0	0,687	0	10	1,98	0	0	0,66	0	12
4	04:00	23,8	24	0	0,619	0	7	1,3	0	0	0,66	0	-4
5	05:00	23,5	24	0	0,49	0	6	1,3	0	0	0,66	0	-1
6	06:00	22,4	24	31,28	0,27	66,77	5	20	441,06	70,76	0,66	2675,41	-3
7	07:00	22	24	12,08	0,037	3,53	4	12,3	242,25	57,89	0,66	1568,88	-4
8	08:00	25,7	24	0	0	0	4	7,86	156,68	63,03	0,66	1148,5	42
9	09:00	27,9	24	0	0	0	3	5,68	113,66	72,98	0,66	975,59	96
10	10:00	28,7	24	0	0	0	3	4,9	94,1	82,76	0,66	924,46	11
11	11:00	30,5	24	0	0	0	2	4,2	85,08	89,78	0,66	914,05	16
12	12:00	32,2	24	0	0	0	2	3,5	81,89	92,66	0,66	912,37	20

Şekil 5.5: Pencere ekranı.

İlgili cepheye ait "Kapı" kısmına tıklandığında kapılara ait 24 saat için hesaplanan güneş ısı kazanç katsayısı, direkt güneş ısı kazancı, güneşe bağlı ışınım zaman faktörü, direkt güneş soğutma yükü, yayılan güneş ısı kazanç katsayısı, yayılan güneş ısı kazancı, iletim ısı kazancı, iletim ve yayılan güneş ısı kazançlarının taşınımsal ve ışınımsal kısımları, güneşe bağlı olmayan ışınım zaman faktörü, ışınımsal soğutma yükü ve toplam soğutma yükü tablo halinde ekranda gösterilir (Şekil 5.6).

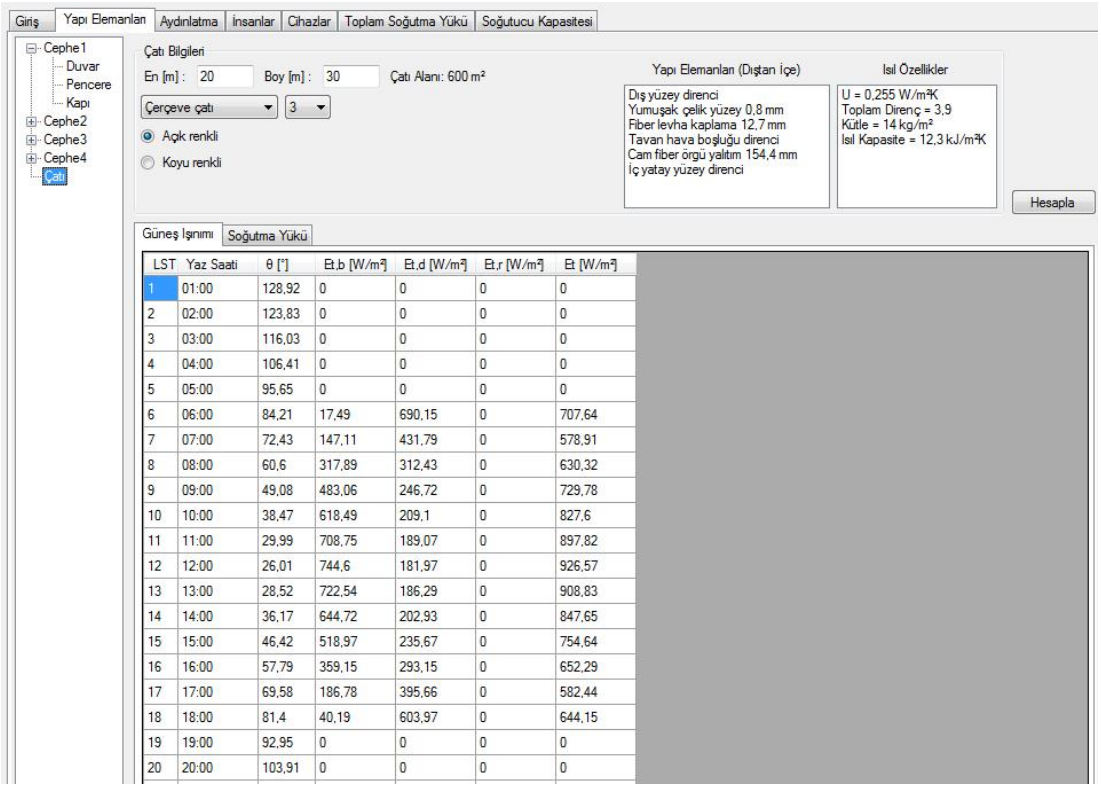
"Çatı" ekranında çatıya ait en ve boy ölçüleri girilerek ilgili listelerden tipi seçilir. Seçilen çatı türüne göre yapı bileşenleri ve ısıl özellikler ekranda gösterilir. Renk seçimi yapıldıktan sonra "Hesapla" butonuna basıldığında 24 saat için hesaplanan geliş açısı (θ), yüzeye gelen direkt güneş ışınımı ($E_{t,b}$), yüzeye yayılan güneş ışınımı ($E_{t,d}$) ve yerden yüzeye yansıyan güneş ışınımı ($E_{t,r}$) değerleri tablo halinde ekranda gösterilir (Şekil 5.7).

Model binaya ait çatıya gelen güneş ışınımı verileri incelendiğinde gece saatlerinde her üç ışınım bileşeninin de sıfır olduğu, yerden yansıyan ışınım bileşeninin ise her saatte sıfır olduğu görülmektedir.



LST	Yaz Saati	Tdış [°C]	Tiç [°C]	Et,b [W/m²]	SHGC	qbeam [W]	Solar RTS [%]	Qb [W]	Et,d [W/m²]	Et,r [W/m²]	SHGC_D	qdiffuse [W]
1	01:00	26,3	24	0	0,836	0	28	8,06	0	0	0,78	0
2	02:00	25,3	24	0	0,827	0	15	6,47	0	0	0,78	0
3	03:00	24,5	24	0	0,807	0	10	4,85	0	0	0,78	0
4	04:00	23,8	24	0	0,764	0	7	3,18	0	0	0,78	0
5	05:00	23,5	24	0	0,659	0	6	3,18	0	0	0,78	0
6	06:00	22,4	24	31,28	0,43	161,37	5	48,36	441,06	70,76	0,78	4790,68
7	07:00	22	24	12,08	0,06	8,65	4	29,81	242,25	57,89	0,78	2809,29
8	08:00	25,7	24	0	0	0	4	19,02	156,68	63,03	0,78	2056,55
9	09:00	27,9	24	0	0	0	3	13,75	113,66	72,98	0,78	1746,93
10	10:00	28,7	24	0	0	0	3	11,88	94,1	82,76	0,78	1655,37
11	11:00	30,5	24	0	0	0	2	10,18	85,08	89,78	0,78	1636,73
12	12:00	32,2	24	0	0	0	2	8,48	81,89	92,66	0,78	1633,72

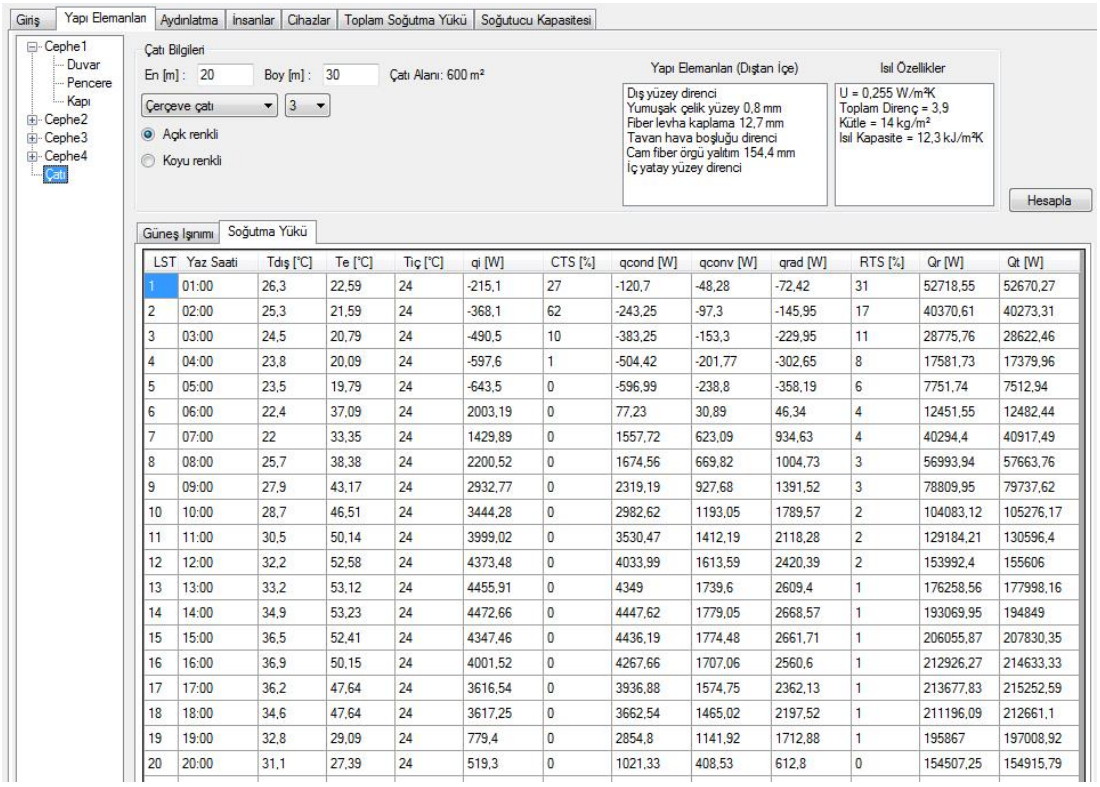
Şekil 5.6: Kapı ekranı.



LST	Yaz Saati	θ [°]	Et,b [W/m²]	Et,d [W/m²]	Et,r [W/m²]	Et [W/m²]
1	01:00	128,92	0	0	0	0
2	02:00	123,83	0	0	0	0
3	03:00	116,03	0	0	0	0
4	04:00	106,41	0	0	0	0
5	05:00	95,65	0	0	0	0
6	06:00	84,21	17,49	690,15	0	707,64
7	07:00	72,43	147,11	431,79	0	578,91
8	08:00	60,6	317,89	312,43	0	630,32
9	09:00	49,08	483,06	246,72	0	729,78
10	10:00	38,47	618,49	209,1	0	827,6
11	11:00	29,99	708,75	189,07	0	897,82
12	12:00	26,01	744,6	181,97	0	926,57
13	13:00	28,52	722,54	186,29	0	908,83
14	14:00	36,17	644,72	202,93	0	847,65
15	15:00	46,42	518,97	235,67	0	754,64
16	16:00	57,79	359,15	293,15	0	652,29
17	17:00	69,58	186,78	395,66	0	582,44
18	18:00	81,4	40,19	603,97	0	644,15
19	19:00	92,95	0	0	0	0
20	20:00	103,91	0	0	0	0

Şekil 5.7: Çatı ışınım ekranı.

"Soğutma Yükü" sekmesine tıklandığında 24 saat için hesaplanan eşdeğer sıcaklık, ısı girişi, iletim zaman faktörü, iletimsel ısı kazancı, taşınımsal ısı kazancı, ışınımsal ısı kazancı, ışınım zaman faktörü, ışınımsal soğutma yükü ve toplam soğutma yükü tablo halinde ekranda gösterilir (Şekil 5.8).



LST	Yaz Saati	Tdış [°C]	Te [°C]	Tiç [°C]	qi [W]	CTS [%]	qcond [W]	qconv [W]	qrad [W]	RTS [%]	Qr [W]	Qt [W]
1	01:00	26,3	22,59	24	-215,1	27	-120,7	-48,28	-72,42	31	52718,55	52670,27
2	02:00	25,3	21,59	24	-368,1	62	-243,25	-97,3	-145,95	17	40370,61	40273,31
3	03:00	24,5	20,79	24	-490,5	10	-383,25	-153,3	-229,95	11	28775,76	28622,46
4	04:00	23,8	20,09	24	-597,6	1	-504,42	-201,77	-302,65	8	17581,73	17379,96
5	05:00	23,5	19,79	24	-643,5	0	-596,99	-238,8	-358,19	6	7751,74	7512,94
6	06:00	22,4	37,09	24	2003,19	0	77,23	30,89	46,34	4	12451,55	12482,44
7	07:00	22	33,35	24	1429,89	0	1557,72	623,09	934,63	4	40294,4	40917,49
8	08:00	25,7	38,38	24	2200,52	0	1674,56	669,82	1004,73	3	56993,94	57663,76
9	09:00	27,9	43,17	24	2932,77	0	2319,19	927,68	1391,52	3	78809,95	79737,62
10	10:00	28,7	46,51	24	3444,28	0	2982,62	1193,05	1789,57	2	104083,12	105276,17
11	11:00	30,5	50,14	24	3999,02	0	3530,47	1412,19	2118,28	2	129184,21	130596,4
12	12:00	32,2	52,58	24	4373,48	0	4033,99	1613,59	2420,39	2	153992,4	155606
13	13:00	33,2	53,12	24	4455,91	0	4349	1739,6	2609,4	1	176258,56	177998,16
14	14:00	34,9	53,23	24	4472,66	0	4447,62	1779,05	2668,57	1	193069,95	194849
15	15:00	36,5	52,41	24	4347,46	0	4436,19	1774,48	2661,71	1	206055,87	207830,35
16	16:00	36,9	50,15	24	4001,52	0	4267,66	1707,06	2560,6	1	212926,27	214633,33
17	17:00	36,2	47,64	24	3616,54	0	3936,88	1574,75	2362,13	1	213677,83	215252,59
18	18:00	34,6	47,64	24	3617,25	0	3662,54	1465,02	2197,52	1	211196,09	212661,1
19	19:00	32,8	29,09	24	779,4	0	2854,8	1141,92	1712,88	1	195867	197008,92
20	20:00	31,1	27,39	24	519,3	0	1021,33	408,53	612,8	0	154507,25	154915,79

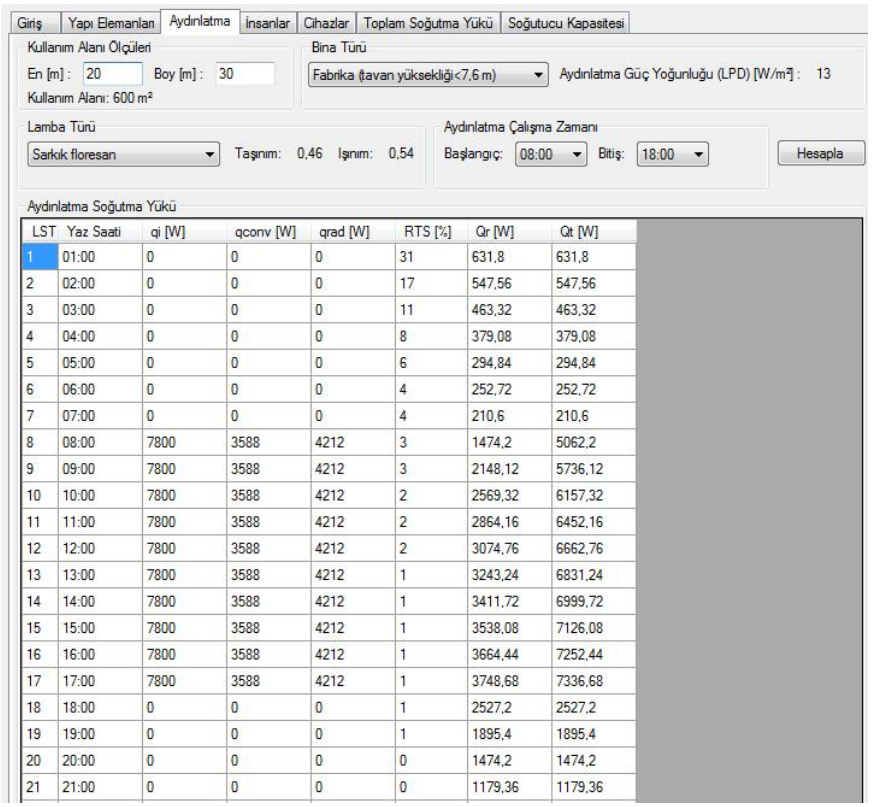
Şekil 5.8: Çatı soğutma yükü ekranı.

5.3 Aydınlatma Sekmesi

"Aydınlatma" sekmesinde binanın kullanım alanı ölçüleri girilerek aydınlatma güç yoğunluğunun belirlenmesi için listeden bina türü seçilir. Daha sonra kullanılan lambaların türü ilgili listeden seçilir. Son olarak aydınlatmanın başlangıç ve bitiş saatleri ilgili listelerden seçilir.

Model binanın kullanım alanı ölçüleri 20 m x 30 m olup, 7,6 m'den küçük tavan yüksekliğinde fabrika olarak kullanılmaktadır. Mesai saatleri olan 08:00 ile 18:00 saatleri arasında sarkık floresan lambalar ile aydınlatma sağlanmaktadır.

"Hesapla" butonuna basıldığında 24 saat için hesaplanan ısı kazancı, ısı kazancının taşınımsal ve ışınımsal kısımları, ışınım zaman faktörü, ışınımsal soğutma yükü ve toplam soğutma yükü tablo halinde ekranda gösterilir (Şekil 5.9). Başlangıç ve bitiş saatleri aynı girilirse sistem kullanıcıyı uyararak hesaplamayı durduracaktır (Şekil 5.10).



LST	Yaz Saati	qi [W]	qconv [W]	qrad [W]	RTS [%]	Qr [W]	Qt [W]
1	01:00	0	0	0	31	631,8	631,8
2	02:00	0	0	0	17	547,56	547,56
3	03:00	0	0	0	11	463,32	463,32
4	04:00	0	0	0	8	379,08	379,08
5	05:00	0	0	0	6	294,84	294,84
6	06:00	0	0	0	4	252,72	252,72
7	07:00	0	0	0	4	210,6	210,6
8	08:00	7800	3588	4212	3	1474,2	5062,2
9	09:00	7800	3588	4212	3	2148,12	5736,12
10	10:00	7800	3588	4212	2	2569,32	6157,32
11	11:00	7800	3588	4212	2	2864,16	6452,16
12	12:00	7800	3588	4212	2	3074,76	6662,76
13	13:00	7800	3588	4212	1	3243,24	6831,24
14	14:00	7800	3588	4212	1	3411,72	6999,72
15	15:00	7800	3588	4212	1	3538,08	7126,08
16	16:00	7800	3588	4212	1	3664,44	7252,44
17	17:00	7800	3588	4212	1	3748,68	7336,68
18	18:00	0	0	0	1	2527,2	2527,2
19	19:00	0	0	0	1	1895,4	1895,4
20	20:00	0	0	0	0	1474,2	1474,2
21	21:00	0	0	0	0	1179,36	1179,36

Şekil 5.9: Aydınlatma sekmesi.

Model binaya ait hesap verileri incelendiğinde çalışma saatlerinde 7800 W'lık bir ısı kazancı olduğu ve aydınlatmadan kaynaklanan maksimum soğutma yükünün saat 17:00'da meydana geldiği görülmektedir.

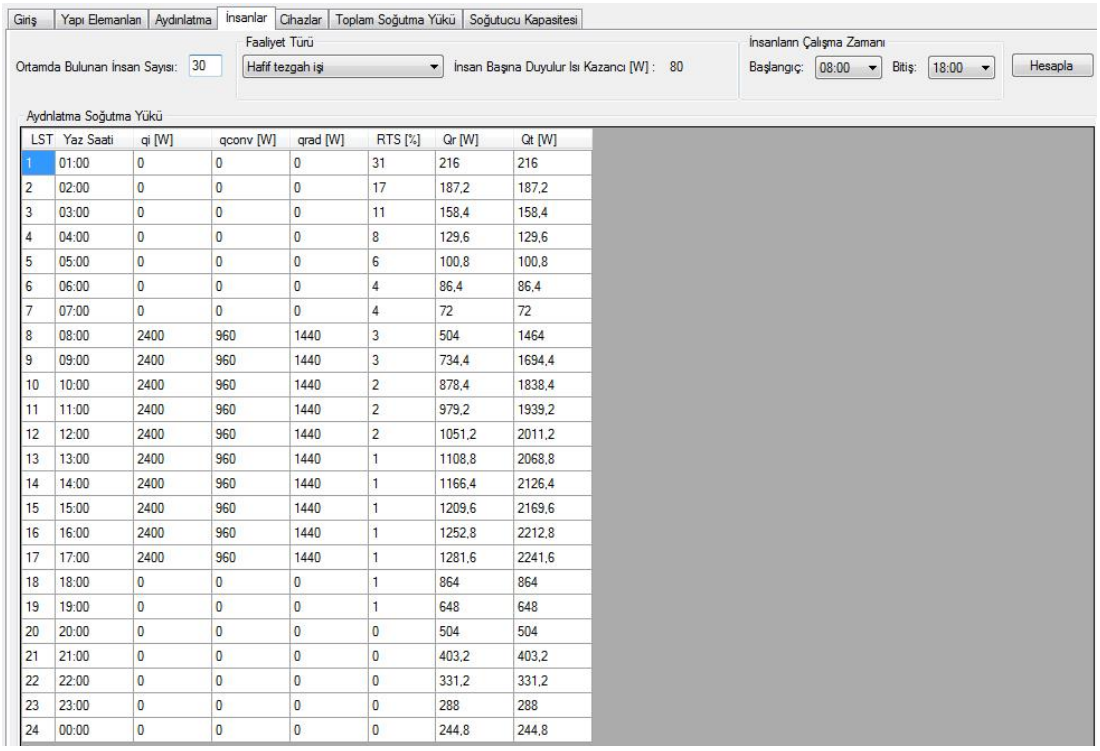
Şekil 5.10: Saat aralığı uyarı mesajı.

5.4 İnsanlar Sekmesi

"İnsanlar" sekmesinde içeride bulunan insan sayısı girilerek listeden faaliyet türü seçilir. Son olarak faaliyetin başlangıç ve bitiş saatleri ilgili listelerden seçilir.

Model binada 08:00 ile 18:00 saatleri arasında 30 kişi tarafından hafif tezgah işi yapılmaktadır.

"Hesapla" butonuna basıldığında 24 saat için hesaplanan ısı kazancı, ısı kazancının taşınımsal ve ışınımsal kısımları, ışınım zaman faktörü, ışınımsal soğutma yükü ve toplam soğutma yükü tablo halinde ekranda gösterilir (Şekil 5.11). Başlangıç ve bitiş saatleri aynı girilirse sistem kullanıcıyı uyararak hesaplamayı durduracaktır (bkz. Şekil 5.10).



LST	Yaz Saati	qi [W]	qconv [W]	qrad [W]	RTS [%]	Qr [W]	Qt [W]
1	01:00	0	0	0	31	216	216
2	02:00	0	0	0	17	187,2	187,2
3	03:00	0	0	0	11	158,4	158,4
4	04:00	0	0	0	8	129,6	129,6
5	05:00	0	0	0	6	100,8	100,8
6	06:00	0	0	0	4	86,4	86,4
7	07:00	0	0	0	4	72	72
8	08:00	2400	960	1440	3	504	1464
9	09:00	2400	960	1440	3	734,4	1694,4
10	10:00	2400	960	1440	2	878,4	1838,4
11	11:00	2400	960	1440	2	979,2	1939,2
12	12:00	2400	960	1440	2	1051,2	2011,2
13	13:00	2400	960	1440	1	1108,8	2068,8
14	14:00	2400	960	1440	1	1166,4	2126,4
15	15:00	2400	960	1440	1	1209,6	2169,6
16	16:00	2400	960	1440	1	1252,8	2212,8
17	17:00	2400	960	1440	1	1281,6	2241,6
18	18:00	0	0	0	1	864	864
19	19:00	0	0	0	1	648	648
20	20:00	0	0	0	0	504	504
21	21:00	0	0	0	0	403,2	403,2
22	22:00	0	0	0	0	331,2	331,2
23	23:00	0	0	0	0	288	288
24	00:00	0	0	0	0	244,8	244,8

Şekil 5.11: İnsanlar sekmesi.

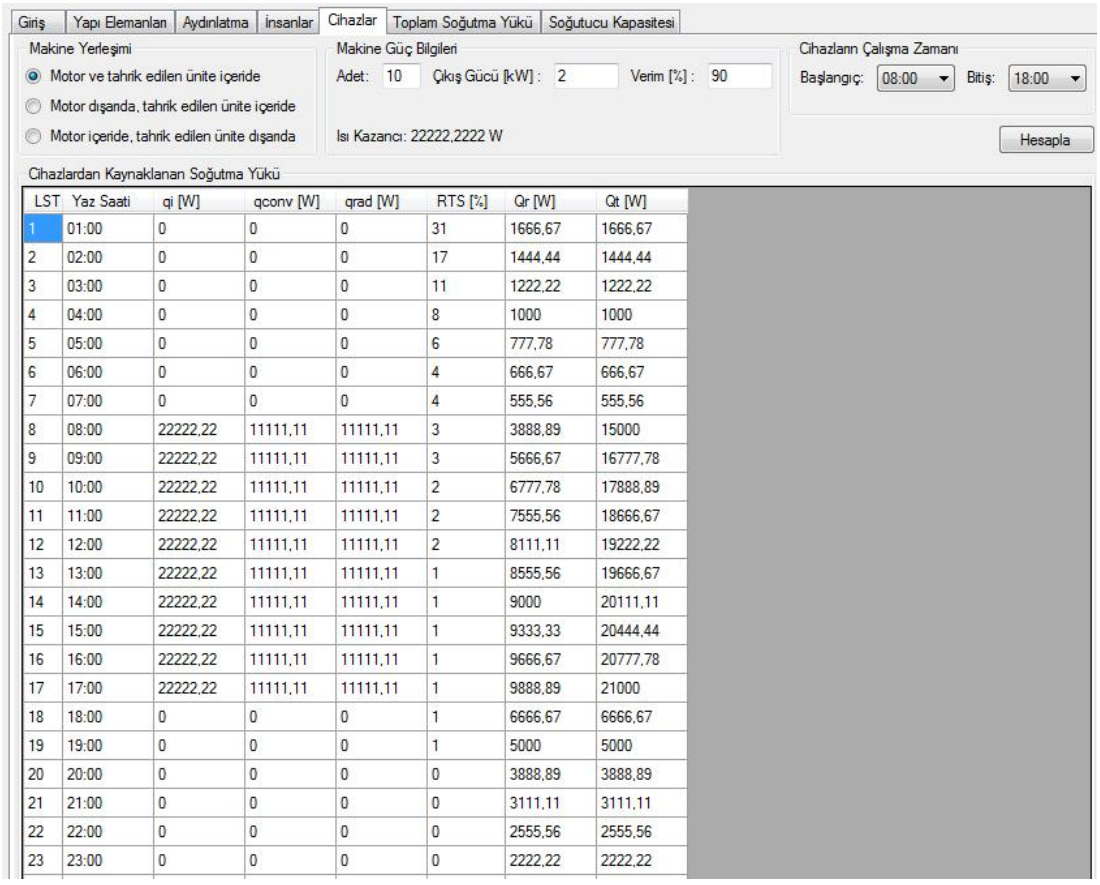
Model binaya ait hesap verileri incelendiğinde çalışma saatlerinde 2400 W'lık bir ısı kazancı olduğu ve insanlardan kaynaklanan maksimum soğutma yükünün saat 17:00'da gerçekleştiği görülmektedir.

5.5 Cihazlar Sekmesi

"Cihazlar" sekmesinde ilk olarak makinelerin yerleşim türü seçilir. Daha sonra adet, çıkış gücü ve verim değerleri girilerek makinelerin çalışma başlangıç ve bitiş saatleri ilgili listelerden seçilir. Adet kutusuna veri girilmediğinde cihazlardan kaynaklanan soğutma yükü sıfır olarak kaydedilir.

Model binada motoru ve tahrik ünitesi içeride bulunan, %90 verimli, 2 kW çıkış gücüne sahip 10 adet makine 08:00 ile 18:00 saatleri arasında çalıştırılmaktadır.

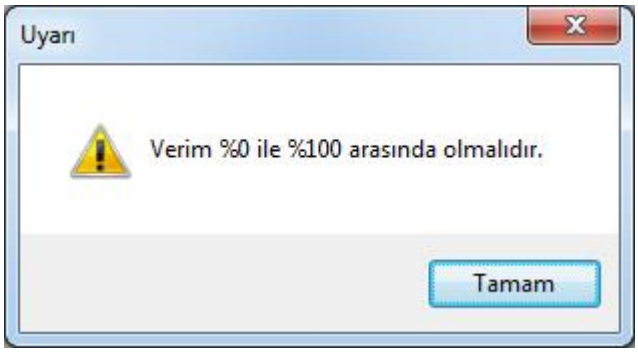
"Hesapla" butonuna basıldığında 24 saat için hesaplanan ısı kazancı, ısı kazancının taşınımsal ve ışınımsal kısımları, ışınım zaman faktörü, ışınımsal soğutma yükü ve toplam soğutma yükü tablo halinde ekranda gösterilir (Şekil 5.12). Verim değeri %0-100 aralığından farklı girilirse (Şekil 5.13) veya başlangıç ve bitiş saatleri aynı girilirse (bkz. Şekil 5.10) sistem uyarı vererek hesaplamayı durduracaktır.



LST	Yaz Saati	qi [W]	qconv [W]	qrad [W]	RTS [%]	Qr [W]	Qt [W]
1	01:00	0	0	0	31	1666,67	1666,67
2	02:00	0	0	0	17	1444,44	1444,44
3	03:00	0	0	0	11	1222,22	1222,22
4	04:00	0	0	0	8	1000	1000
5	05:00	0	0	0	6	777,78	777,78
6	06:00	0	0	0	4	666,67	666,67
7	07:00	0	0	0	4	555,56	555,56
8	08:00	22222,22	11111,11	11111,11	3	3888,89	15000
9	09:00	22222,22	11111,11	11111,11	3	5666,67	16777,78
10	10:00	22222,22	11111,11	11111,11	2	6777,78	17888,89
11	11:00	22222,22	11111,11	11111,11	2	7555,56	18666,67
12	12:00	22222,22	11111,11	11111,11	2	8111,11	19222,22
13	13:00	22222,22	11111,11	11111,11	1	8555,56	19666,67
14	14:00	22222,22	11111,11	11111,11	1	9000	20111,11
15	15:00	22222,22	11111,11	11111,11	1	9333,33	20444,44
16	16:00	22222,22	11111,11	11111,11	1	9666,67	20777,78
17	17:00	22222,22	11111,11	11111,11	1	9888,89	21000
18	18:00	0	0	0	1	6666,67	6666,67
19	19:00	0	0	0	1	5000	5000
20	20:00	0	0	0	0	3888,89	3888,89
21	21:00	0	0	0	0	3111,11	3111,11
22	22:00	0	0	0	0	2555,56	2555,56
23	23:00	0	0	0	0	2222,22	2222,22

Şekil 5.12: Cihazlar sekmesi.

Model binaya ait hesap verileri incelendiğinde çalışma saatlerinde 22222,22 W'lık bir ısı kazancı olduğu ve cihazlardan kaynaklanan maksimum soğutma yükünün saat 17:00'da gerçekleştiği görülmektedir.



Şekil 5.13: Cihaz verimi uyarı mesajı.

5.6 Toplam Soğutma Yükü Sekmesi

Tüm adımlarda hesaplamalar yapıldıktan sonra "Toplam Soğutma Yükü" sekmesine gelinir. Bu sekmede tüm bileşenlerin saatlik soğutma yükü toplanarak pik soğutma yükü ve bu yükün oluştuğu pik saat ekranda gösterilir. Ayrıca 24 saat boyunca cepheler, çatı, aydınlatma, insanlar ve cihazlardan kaynaklanan soğutma yükleri ile toplam soğutma yükü tablo halinde ekranda gösterilir (Şekil 5.14).

Model binaya ait verilerin girişi sonucunda hesaplanan soğutma yükleri incelendiğinde saat 17:00'da 265650,6845 W olarak maksimum yükün gerçekleştiği görülmektedir.

"Grafik" sekmesine tıklandığında soğutma yüklerinin dağılımı grafiksel olarak gösterilir (Şekil 5.15).

Model binaya ait hesaplanan soğutma yüklerinin grafiksel dağılımı incelendiğinde çatıdan kaynaklanan yükün, diğer bileşenlerden kaynaklanan yüklere göre çok daha yüksek olduğu görülmektedir.

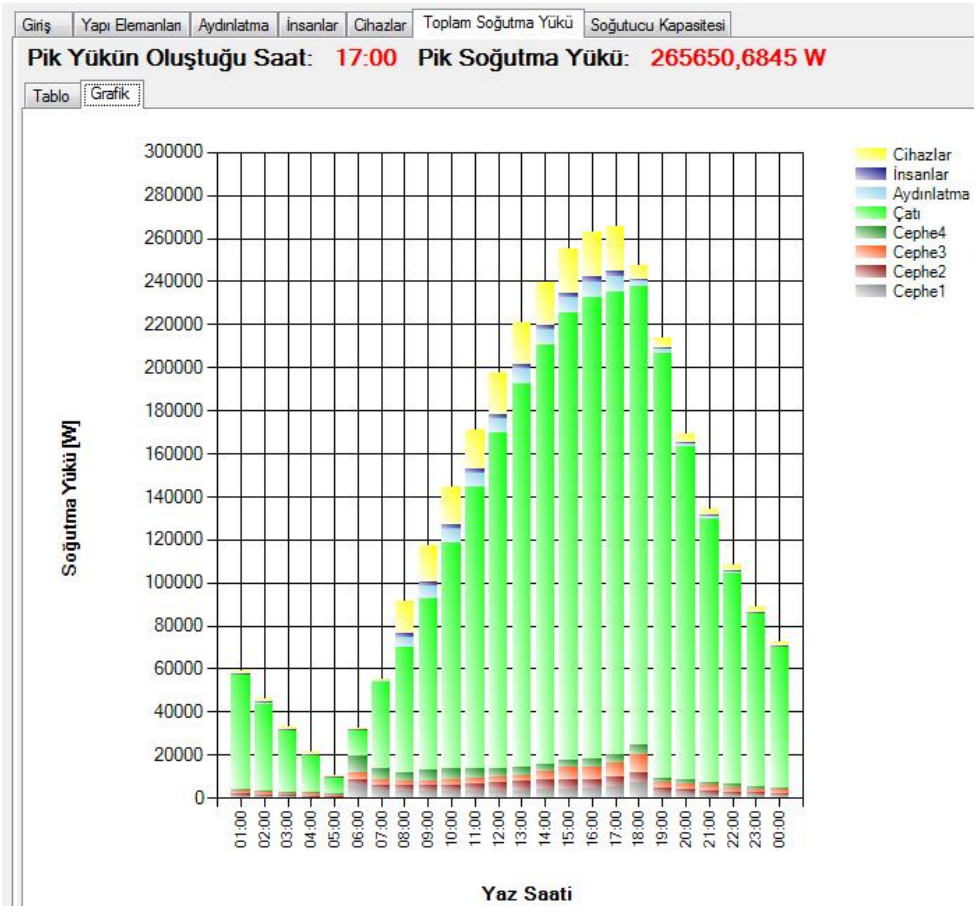
Giriş | Yapı Elemanları | Aydınlatma | İnsanlar | Cihazlar | Toplam Soğutma Yükü | Soğutucu Kapasitesi

Pik Yükün Oluştuğu Saat: 17:00 Pik Soğutma Yükü: 265650,6845 W

Tablo | Grafik

LST	Yaz Saati	Kış Saati	Cephe1	Cephe2	Cephe3	Cephe4	Çatı	Aydınlatma	İnsanlar	Cihazlar	Qtoplam
1	01:00	00:00	871,91	956,15	1283,6	951,74	52670,27	631,8	216	1666,67	59248,14
2	02:00	01:00	701,64	801,81	1114,82	813,28	40273,31	547,56	187,2	1444,44	45884,07
3	03:00	02:00	543,38	663,03	950,08	689,25	28622,46	463,32	158,4	1222,22	33312,14
4	04:00	03:00	415,18	545,33	796,25	584,51	17379,96	379,08	129,6	1000	21229,91
5	05:00	04:00	321,75	454,44	639,81	497,47	7512,94	294,84	100,8	777,78	10599,83
6	06:00	05:00	5988,15	2803,46	3237,1	7148,38	12482,44	252,72	86,4	666,67	32665,32
7	07:00	06:00	3921,2	2213,8	2439,78	5037,34	40917,49	210,6	72	555,56	55367,76
8	08:00	07:00	3392,19	2248,02	2246,92	4126,81	57663,76	5062,2	1464	15000	91203,9
9	09:00	08:00	3229,76	2531,83	2192,78	5147,81	79737,62	5736,12	1694,4	16777,78	117048,11
10	10:00	09:00	3200,72	2958,82	2174,57	5186,24	105276,17	6157,32	1838,4	17888,89	144681,13
11	11:00	10:00	3326,17	3470,41	2200,28	4645,33	130596,4	6452,16	1939,2	18666,67	171296,61
12	12:00	11:00	3498,86	3936,94	2339,79	4044,65	155606	6662,76	2011,2	19222,22	197322,42
13	13:00	12:00	3610	4232,49	2818,92	3715,04	177998,16	6831,24	2068,8	19666,67	220941,31
14	14:00	13:00	3829,83	4406,11	3996,36	3611,54	194849	6999,72	2126,4	20111,11	239930,08
15	15:00	14:00	4085,88	4388,5	5549,27	3562,1	207830,35	7126,08	2169,6	20444,44	255156,22
16	16:00	15:00	4492,09	4256,28	5552,19	3627,71	214633,33	7252,44	2212,8	20777,78	262804,62
17	17:00	16:00	5260,92	4225,17	6501,56	3832,17	215252,59	7336,68	2241,6	21000	265650,68
18	18:00	17:00	7150,62	4577,24	8636,35	4475,82	212661,1	2527,2	864	6666,67	247559
19	19:00	18:00	2257,77	2226,92	2918,76	2023,42	197008,92	1895,4	648	5000	213979,19
20	20:00	19:00	1892,86	1932,72	2619,42	1808,11	154915,79	1474,2	504	3888,89	169035,98
21	21:00	20:00	1592,42	1670,31	2322,81	1623,38	122493,45	1179,36	403,2	3111,11	134396,04
22	22:00	21:00	1386,13	1466,62	2046,29	1477,53	98164,55	968,76	331,2	2555,56	108396,63
23	23:00	22:00	1206,25	1284,98	1754,81	1287,67	79728,94	842,4	288	2222,22	88615,27
24	00:00	23:00	1044,78	1116,11	1529,18	1117,83	65168,37	716,04	244,8	1888,89	72826

Şekil 5.14: Toplam soğutma yükü tablo görünümü.

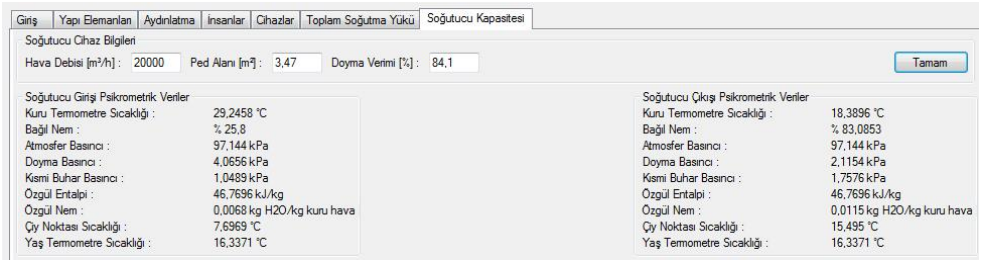


Şekil 5.15: Toplam soğutma yükü grafik görünümü.

5.7 Soğutucu Kapasitesi Sekmesi

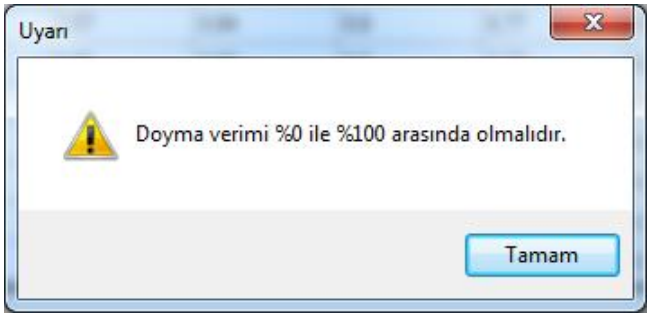
"Soğutucu Kapasitesi" sekmesi iki aşamalıdır. İlk aşamada soğutucu cihaz bilgileri girilir. Üretici kataloglarından temin edilen cihaz hava debisi, ped alanı ve deneysel çalışmayla elde edilen doyma verimi değerleri girildikten sonra "Tamam" butonuna basılır. Soğutucu giriş ve çıkışındaki pskirometrik hesaplamalar ekranda gösterilir (Şekil 5.16). Doyma verimi değeri %0-100 aralığından farklı girilirse (Şekil 5.17), soğutucu çıkış sıcaklığı iç ortam sıcaklığından yüksek veya çiy noktası sıcaklığından düşük çıkarsa (Şekil 5.18) sistem uyarı vererek hesaplamayı durduracaktır.

Model bina için soğutucu kapasitesinin belirlenmesinde Alindair 20S direkt buharlaşmalı soğutucuya ait hava debisi ve ped alanı değerleri (bkz. Tablo B.1) ile bu cihaz ile yapılan saha ölçümünden elde edilen doyma verimi değeri (bkz. Bölüm 4.2) kullanılmıştır.

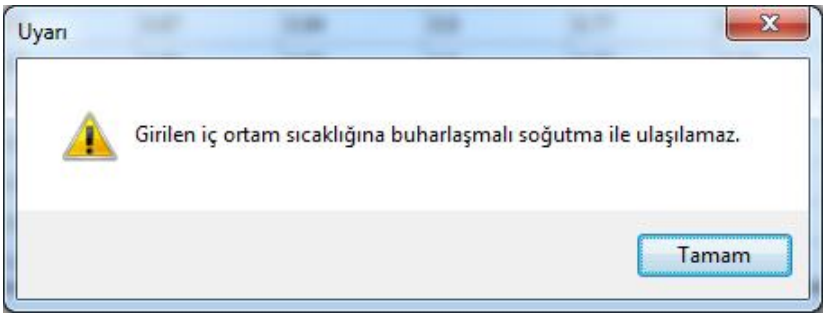


Şekil 5.16: Cihaz bilgileri giriş ekranı.

Model binada kullanılacak olan soğutucuya ait psikrometrik hesap verileri incelendiğinde soğutucudan 18,3896 ºC kuru termometre sıcaklığında ve % 83,0853 bağıl nemde hava çıktığı görülmektedir.



Şekil 5.17: Doyma verimi uyarı mesajı.

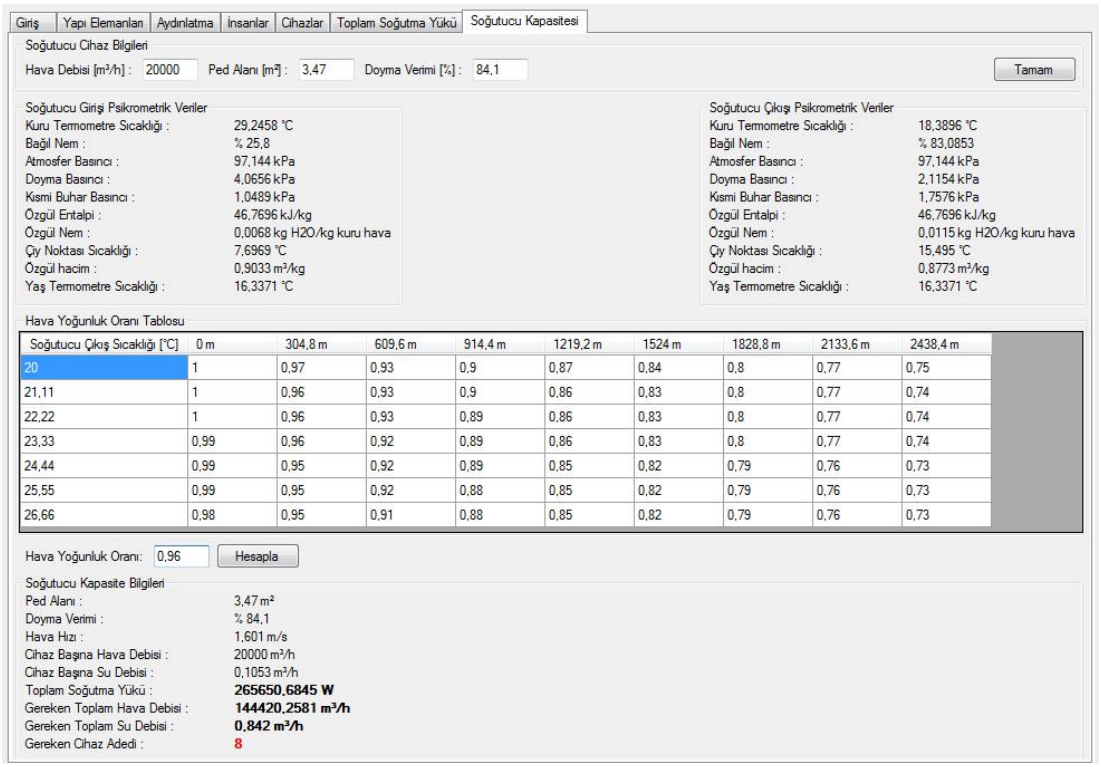


Şekil 5.18: Buharlaşmalı soğutma işlemi uyarı mesajı.

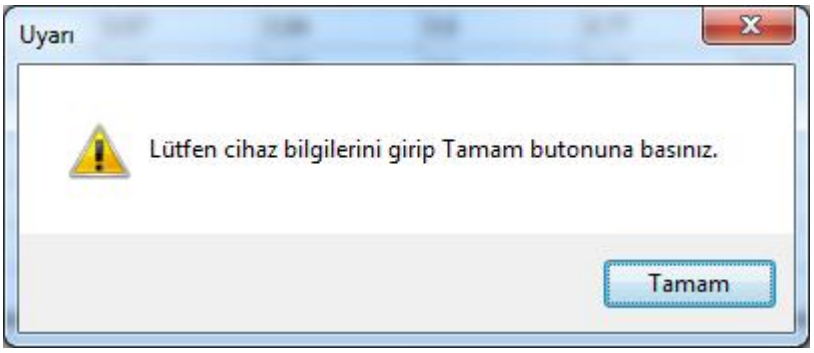
Son olarak hava yoğunluk oranı girilir. Soğutucu çıkış sıcaklığı ve yükseltiye göre hava yoğunluk oranı tablosu bilgi olarak ekranda mevcuttur.

Model binanın yükseltisi 354 m, soğutucu çıkış sıcaklığı 18,3896 ºC olduğu için hava yoğunluk oranı olarak 0,96 değeri seçilmiştir.

"Hesapla" butonuna basıldığında hesaplanan soğutucu hava hızı, cihaz başına beslenmesi gereken su debisi, soğutma için gereken toplam hava ve su debileri ile cihaz adedi ekranda gösterilir (Şekil 5.19). İlk aşamada "Tamam" butonuna basılmazsa sistem uyarı vererek diğer aşamaya geçmeyecektir (Şekil 5.20).



Şekil 5.19: Soğutucu kapasitesi sonuç ekranı.



Şekil 5.20: Tamam butonu uyarı mesajı.

Yapılan hesaplar sonucunda model binanın soğutma yükünü karşılamak için 20000 m^3/h hava debisi veren Alindair 20S direkt buharlaşmalı soğutucudan 8 adet kullanılması gerektiği görülmektedir. Bu soğutuculara toplamda 0,842 m^3/h debide su beslenmelidir.

6. SONUÇ VE ÖNERİLER

Deney düzeneğinden elde edilen veriler sonucunda, yaklaşık olarak sabit hava giriş koşulları altında Greenpad 5090/100 buharlaşmalı soğutma pedine ait doyma veriminin hava hızıyla ters orantılı olarak değiştiği görülmüştür. Ayrıca direkt buharlaşmalı soğutucu üzerinde pedin gerçek durumdaki doyma verimi belirlenmiş ve deney verileriyle tutarlılık gözlemlenmiştir.

Hazırlanan program, karmaşık ve zaman alıcı soğutma yükü hesaplarını anında yapıp sonuçlarını gösterebilmektedir. Metin kutularında sadece sayı girişine izin verilmesi, zaman aralıkları veya verim değerleri yanlış girildiğinde sistemin uyarı vererek hesaplamayı durdurması kullanıcılar tarafından yapılacak olası hatalı girişleri önlemektedir. Ayrıca bu program sayesinde binalarda meydana gelebilecek soğutma yükleri tasarım aşamasında belirlenebilmekte, farklı türdeki yapı elemanlarının soğutma yüküne etkileri kolayca karşılaştırılabilmekte ve buharlaşmalı soğutma sistemlerinin binanın iklim koşulları için uygun olup olmadığı tespit edilebilmektedir.

Bu programda genel uygulama olarak bina zemininin toprakla temas ettiği ve buharlaşmalı soğutucu kullanılıp ortam pozitif basınç altında tutulduğu için dışarıdan içeriye sızıntı olmadığı kabul edilerek hesaplamalar yapılmaktadır. Eğer zemin, iç ortam sıcaklığından daha sıcak bir ortam üzerindeyse veya dışarıdan içeriye sızıntı varsa buralardan gerçekleşecek ısı transferi ayrıca dikkate alınmalıdır.

Soğutma yükü hesaplamalarında ASHRAE tarafından belli yapı elemanları için türetilen iletim ve ışınım zaman faktörleri kullanılmıştır. Türkiye'de daha sık kullanılan yapı elemanları için iletim ve ışınım zaman faktörlerinin türetilmesi, bina soğutma yükünün hesaplanmasında daha hassas ve doğru sonuçlar verecektir.

7. KAYNAKLAR

Bhatia, A., "Principles of evaporative cooling system [online]", (2 April 2015), http://www.pdhonline.org/courses/m231/m231content.pdf, (2012).

Bulut, H., Durmaz, A. F. ve Aktacir, M. A., "İklimlendirme sistemleri için soğutma yükü hesap yöntemlerinin karşılaştırılması", *VII. Uluslararası Yapıda Tesisat Teknolojisi Sempozyumu*, İstanbul, (2006).

Camargo, J. R., Ebinuma, C. D. and Silveira, J. L., "Experimental performance of a direct evaporative cooler operating during summer in a Brazilian city", *Int. Journal of Refrigeration*, 28 (7), 1124-1132, Elsevier, (2005).

Dai, Y. J. and Sumathy, K., "Theoretical study on a cross-flow direct evaporative cooler using honeycomb paper as packing material", *Applied Thermal Engineering*, 22 (13), 1417-1430, Elsevier, (2002).

Elmetenani, S., Yousfi, M. L., Merabeti, L., Belgroun, Z. and Chikouche, A., "Investigation of an evaporative air cooler using solar energy under Algerian climate", *Energy Procedia*, 6, 573-582, Elsevier, (2011).

Fouda, A. and Melikyan, Z., "A simplified model for analysis of heat and mass transfer in a direct evaporative cooler", *Applied Thermal Engineering*, 31 (5), 932-936, Elsevier, (2011).

Guo, X. C. and Zhao, T. S., "A parametric study of an indirect evaporative air cooler", *Int. Comm. Heat and Mass Transfer*, 25 (2), 217-226, Elseiver, (1998).

Hajidavalloo, E., "Application of evaporative cooling on the condenser of window-air-conditioner", *Applied Thermal Engineering*, 27 (11-12), 1937-1943, Elseiver, (2007).

Halasz, B., "A general mathematical model of evaporative cooling devices", *Revue Générale de Thermique*, 37 (4), 245-255, Elsevier, (1998).

Joudi, K. A. and Mehdi, S. M., "Application of indirect evaporative cooling to variable domestic cooling load", *Energy Conversion & Management*, 41 (17), 1931-1951, Elsevier, (2000).

Lazzarin, R. M., "Introduction of a simple diagram-based method for analyzing evaporative cooling", *Applied Thermal Engineering*, 27 (11-12), 2011-2025, Elseiver, (2007).

Martin, R. H., "Characterization of a semi-indirect evaporative cooler", *Applied Thermal Engineering*, 29 (10), 2113-2117, Elsevier, (2009).

Oranlıer, B. ve Eyriboyun, M., "İklimlendirme amaçlı ısı kazancı hesabı için bir yazılım", *IX. Ulusal Tesisat Mühendisliği Kongresi*, İzmir, 225-242, (2009).

Owen, M. S. (Ed.), *2009 ASHRAE Handbook Fundamentals SI Edition*, Atlanta: ASHRAE, (2009).

Özgören, M., Erdoğan, K., Kahraman, A. ve Solmaz, Ö., "Model bir konutun soğutma yükünün dinamik hesaplanması: İzmir örneği", *X. Ulusal Tesisat Mühendisliği Kongresi*, İzmir, 1231-1247, (2011).

Pires, L., Silva, P. D. and Gomez, J. P. C., "Performance of textile and building materials for a particular evaporative cooling purpose", *Experimental Thermal and Fluid Science*, 35 (4), 670-675, Elsevier, (2011).

Riffat, S. B. and Zhu, J., "Mathematical model of indirect evaporative cooler using porous ceramic and heat pipe", *Applied Thermal Engineering*, 24 (4), 457-470, Elsevier, (2004).

Steeman, M., Janssens, A. and Paepe, M. D., "Performance evaluation of indirect evaporative cooling using whole-building hygrothermal simulations", *Applied Thermal Engineering*, 29 (14-15), 2870-2875, Elsevier, (2009).

Tulsidasani, T. R., Sawhney, R. L., Singh, S. P. and Sodha, M. S., "Recent researches in indirect evaporative cooler III: Optimization of the cooling potential of a room-coupled indirect evaporative cooler", *Int. Journal of Energy Research*, 22, 741-750, John Wiley & Sons, (1998).

url-1, "Nonrefrigerative cooling systems [online]", (25 December 2013), http://fpl.bizenergyadvisor.com/members/CEMC-OMCK-5x-BEA/OM-Checklist/Nonrefrigerative-Cooling-Systems, (2013).

url-2, "Evaporative cooling systems: Saving energy in more ways than ever [online]", (25 December 2013), http://energydesignresources.com/resources/e-news/e-news-71-evaporative-cooling-saving-energy-in-more-ways-than-ever.aspx, (2010).

url-3, http://orencati.com/?page_id=12, (4 Haziran 2015).

url-4,
http://www.alindair.com.tr/?sayfa=icerikSayfa&mode=3&id=11&urun=1, (4 Haziran 2015).

Wang, T., Sheng, C. and Nnanna, A. G. A., "Experimental investigation of air conditioning system using evaporative cooling condenser", *Energy and Buildings*, 81, 435-443, Elsevier, (2014).

Zhao, X., Li, J. M. and Riffat, S. B., "Numerical study of a novel counter-flow heat and mass exchanger for dew point evaporative cooling", *Applied Thermal Engineering*, 28 (14-15), 1942-1951, Elsevier, (2008).

EKLER

8. EKLER

EK A Soğutma Yükü Hesap Tabloları

Tablo A.1: Belli duvar tipleri için iletim zaman faktörleri (Owen 2009).

Tip	**Perde Duvar**			**Saplama Duvar**				**Kompozit Yalıtımlı Duvar**		
No	1	2	3	4	5	6	7	8	9	10
Saat	**İletim Zaman Faktörü [%]**									
0	18	25	8	19	6	7	5	11	2	1
1	58	57	45	59	42	44	41	50	25	2
2	20	15	32	18	33	32	34	26	31	6
3	4	3	11	3	13	12	13	9	20	9
4	0	0	3	1	4	4	4	3	11	9
5	0	0	1	0	1	1	2	1	5	9
6	0	0	0	0	1	0	1	0	3	8
7	0	0	0	0	0	0	0	0	2	7
8	0	0	0	0	0	0	0	0	1	6
9	0	0	0	0	0	0	0	0	0	6
10	0	0	0	0	0	0	0	0	0	5
11	0	0	0	0	0	0	0	0	0	5
12	0	0	0	0	0	0	0	0	0	4
13	0	0	0	0	0	0	0	0	0	4
14	0	0	0	0	0	0	0	0	0	3
15	0	0	0	0	0	0	0	0	0	3
16	0	0	0	0	0	0	0	0	0	3
17	0	0	0	0	0	0	0	0	0	2
18	0	0	0	0	0	0	0	0	0	2
19	0	0	0	0	0	0	0	0	0	2
20	0	0	0	0	0	0	0	0	0	2
21	0	0	0	0	0	0	0	0	0	1
22	0	0	0	0	0	0	0	0	0	1
23	0	0	0	0	0	0	0	0	0	0

Tablo A.1 (devam): Belli duvar tipleri için iletim zaman faktörleri (Owen 2009).

Tip	**Tuğla Duvar**									
No	11	12	13	14	15	16	17	18	19	20
Saat	**İletim Zaman Faktörü [%]**									
0	0	0	0	1	2	2	1	3	4	3
1	5	4	1	1	2	2	1	3	4	3
2	14	13	7	2	2	2	3	3	4	3
3	17	17	12	5	3	4	6	3	4	4
4	15	15	13	8	5	5	7	3	4	4
5	12	12	13	9	6	6	8	4	4	4
6	9	9	11	9	7	6	8	4	4	5
7	7	7	9	9	7	7	8	5	4	5
8	5	5	7	8	7	7	8	5	4	5
9	4	4	6	7	7	6	7	5	4	5
10	3	3	5	7	6	6	6	5	4	5
11	2	2	4	6	6	6	6	5	5	5
12	2	2	3	5	5	5	5	5	5	5
13	1	2	2	4	5	5	4	5	5	5
14	1	2	2	4	5	5	4	5	5	5
15	1	1	1	3	4	4	3	5	4	4
16	1	1	1	3	4	4	3	5	4	4
17	1	1	1	2	3	4	3	4	4	4
18	0	0	1	2	3	3	2	4	4	4
19	0	0	1	2	3	3	2	4	4	4
20	0	0	0	1	3	3	2	4	4	4
21	0	0	0	1	2	2	1	4	4	4
22	0	0	0	1	2	2	1	4	4	3
23	0	0	0	0	1	1	1	3	4	3

Tablo A.1 (devam): Belli duvar tipleri için iletim zaman faktörleri (Owen 2009).

Tip	**Beton Blok Duvar**						**Öndökümlü ve Yerinde Dökme Duvar**			
No	21	22	23	24	25	26	27	28	29	30
Saat	**İletim Zaman Faktörü [%]**									
0	0	1	0	1	0	1	1	0	1	2
1	4	1	2	11	3	1	10	8	1	2
2	13	5	8	21	12	2	20	18	3	3
3	16	9	12	20	16	5	18	18	6	5
4	14	11	12	15	15	7	14	14	8	6
5	11	10	11	10	12	9	10	11	9	6
6	9	9	9	7	10	9	7	8	9	6
7	7	8	8	5	8	8	5	6	9	6
8	6	7	7	3	6	8	4	4	8	6
9	4	6	6	2	4	7	3	3	7	6
10	3	5	5	2	3	6	2	2	7	5
11	3	4	4	1	3	6	2	2	6	5
12	2	4	3	1	2	5	1	2	5	5
13	2	3	2	1	2	4	1	1	4	5
14	2	3	2	0	1	4	1	1	4	4
15	1	3	2	0	1	3	1	1	3	4
16	1	2	1	0	1	3	0	1	2	4
17	1	2	1	0	1	2	0	0	2	3
18	1	2	1	0	0	2	0	0	1	3
19	0	1	1	0	0	2	0	0	1	3
20	0	1	1	0	0	2	0	0	1	3
21	0	1	1	0	0	2	0	0	1	3
22	0	1	1	0	0	1	0	0	1	3
23	0	1	0	0	0	1	0	0	1	2

Tablo A.1 (devam): Belli duvar tipleri için iletim zaman faktörleri (Owen 2009).

Tip	**Öndökümlü ve Yerinde Dökme Duvar**				
No	31	32	33	34	35
Saat	**İletim Zaman Faktörü [%]**				
0	1	3	1	2	1
1	2	3	2	2	2
2	3	4	5	3	4
3	6	5	8	3	7
4	7	6	9	5	8
5	8	6	9	5	8
6	8	6	8	6	8
7	7	5	7	6	8
8	7	5	6	6	7
9	6	5	6	6	6
10	6	5	5	6	6
11	5	5	5	5	5
12	5	4	4	5	4
13	4	4	4	5	4
14	4	4	3	4	4
15	3	4	3	4	3
16	3	4	3	4	3
17	3	4	2	4	3
18	2	4	2	4	2
19	2	3	2	3	2
20	2	3	2	3	2
21	2	3	2	3	1
22	2	3	1	3	1
23	2	2	1	3	1

Tablo A.2: Belli çatı tipleri için iletim zaman faktörleri (Owen 2009).

Tip	**Çerçeve Çatı**						**Ahşap Döşeme**		**Metal Döşeme**	
No	1	2	3	4	5	6	7	8	9	10
Saat	**İletim Zaman Faktörü [%]**									
0	6	10	27	1	1	1	0	1	18	4
1	45	57	62	17	17	12	7	3	61	41
2	33	27	10	31	34	25	18	8	18	35
3	11	5	1	24	25	22	18	10	3	14
4	3	1	0	14	13	15	15	10	0	4
5	1	0	0	7	6	10	11	9	0	1
6	1	0	0	4	3	6	8	8	0	1
7	0	0	0	2	1	4	6	7	0	0
8	0	0	0	0	0	2	5	6	0	0
9	0	0	0	0	0	1	3	5	0	0
10	0	0	0	0	0	1	3	5	0	0
11	0	0	0	0	0	1	2	4	0	0
12	0	0	0	0	0	0	1	4	0	0
13	0	0	0	0	0	0	1	3	0	0
14	0	0	0	0	0	0	1	3	0	0
15	0	0	0	0	0	0	1	3	0	0
16	0	0	0	0	0	0	0	2	0	0
17	0	0	0	0	0	0	0	2	0	0
18	0	0	0	0	0	0	0	2	0	0
19	0	0	0	0	0	0	0	2	0	0
20	0	0	0	0	0	0	0	1	0	0
21	0	0	0	0	0	0	0	1	0	0
22	0	0	0	0	0	0	0	1	0	0
23	0	0	0	0	0	0	0	0	0	0

Tablo A.2 (devam): Belli çatı tipleri için iletim zaman faktörleri (Owen 2009).

Tip	**Metal Döşeme**			**Beton Çatı**					
No	11	12	13	14	15	16	17	18	19
Saat	**İletim Zaman Faktörü [%]**								
0	8	1	0	1	2	2	2	3	1
1	53	23	10	2	2	2	2	3	2
2	30	38	22	8	3	3	5	3	6
3	7	22	20	11	6	4	6	5	8
4	2	10	14	11	7	5	7	6	8
5	0	4	10	10	8	6	7	6	8
6	0	2	7	9	8	6	6	6	7
7	0	0	5	7	7	6	6	6	7
8	0	0	4	6	7	6	6	6	6
9	0	0	3	5	6	6	5	5	5
10	0	0	2	5	5	6	5	5	5
11	0	0	1	4	5	5	5	5	5
12	0	0	1	3	5	5	4	5	4
13	0	0	1	3	4	5	4	4	4
14	0	0	0	3	4	4	4	4	3
15	0	0	0	2	3	4	4	4	3
16	0	0	0	2	3	4	3	4	3
17	0	0	0	2	3	4	3	4	3
18	0	0	0	1	3	3	3	3	2
19	0	0	0	1	2	3	3	3	2
20	0	0	0	1	2	3	3	3	2
21	0	0	0	1	2	3	3	3	2
22	0	0	0	1	2	3	2	2	2
23	0	0	0	1	1	2	2	2	2

Tablo A.3: Güneşe bağlı olmayan ışınım zaman faktörleri (Owen 2009).

Yapı Türü	**Hafif**					
	Halı var			Halı yok		
Cam %	%10	%50	%90	%10	%50	%90
Saat	**Işınım Zaman Faktörü [%]**					
0	47	50	53	41	43	46
1	19	18	17	20	19	19
2	11	10	9	12	11	11
3	6	6	5	8	7	7
4	4	4	3	5	5	5
5	3	3	2	4	3	3
6	2	2	2	3	3	2
7	2	1	1	2	2	2
8	1	1	1	1	1	1
9	1	1	1	1	1	1
10	1	1	1	1	1	1
11	1	1	1	1	1	1
12	1	1	1	1	1	1
13	1	1	1	0	1	0
14	0	0	1	0	1	0
15	0	0	1	0	0	0
16	0	0	0	0	0	0
17	0	0	0	0	0	0
18	0	0	0	0	0	0
19	0	0	0	0	0	0
20	0	0	0	0	0	0
21	0	0	0	0	0	0
22	0	0	0	0	0	0
23	0	0	0	0	0	0

Tablo A.3 (devam): Güneşe bağlı olmayan ışınım zaman faktörleri (Owen 2009).

Yapı Türü	**Orta**					
	Halı var			Halı yok		
Cam %	%10	%50	%90	%10	%50	%90
Saat	**Işınım Zaman Faktörü [%]**					
0	46	49	52	31	33	35
1	18	17	16	17	16	15
2	10	9	8	11	10	10
3	6	5	5	8	7	7
4	4	3	3	6	5	5
5	2	2	2	4	4	4
6	2	2	2	4	3	3
7	1	1	1	3	3	3
8	1	1	1	3	2	2
9	1	1	1	2	2	2
10	1	1	1	2	2	2
11	1	1	1	2	2	2
12	1	1	1	1	1	1
13	1	1	1	1	1	1
14	1	1	1	1	1	1
15	1	1	1	1	1	1
16	1	1	1	1	1	1
17	1	1	1	1	1	1
18	1	1	1	1	1	1
19	0	1	0	0	1	1
20	0	0	0	0	1	1
21	0	0	0	0	1	1
22	0	0	0	0	1	0
23	0	0	0	0	0	0

Tablo A.3 (devam): Güneşe bağlı olmayan ışınım zaman faktörleri (Owen 2009).

Yapı Türü	**Ağır**					
	Halı var			Halı yok		
Cam %	%10	%50	%90	%10	%50	%90
Saat	**Işınım Zaman Faktörü [%]**					
0	34	38	42	22	25	28
1	9	9	9	10	9	9
2	6	6	5	6	6	6
3	4	4	4	5	5	5
4	4	4	4	5	5	4
5	4	3	3	4	4	4
6	3	3	3	4	4	4
7	3	3	3	4	4	4
8	3	3	3	4	3	3
9	3	3	2	3	3	3
10	3	2	2	3	3	3
11	2	2	2	3	3	3
12	2	2	2	3	3	3
13	2	2	2	3	3	2
14	2	2	2	3	2	2
15	2	2	2	2	2	2
16	2	2	2	2	2	2
17	2	2	2	2	2	2
18	2	2	1	2	2	2
19	2	2	1	2	2	2
20	2	1	1	2	2	2
21	2	1	1	2	2	2
22	1	1	1	2	2	2
23	1	1	1	2	2	1

Tablo A.4: Güneşe bağlı ışınım zaman faktörleri (Owen 2009).

Yapı Türü	**Hafif**					
	Halı var			Halı yok		
Cam %	%10	%50	%90	%10	%50	%90
Saat	**Işınım Zaman Faktörü [%]**					
0	53	55	56	44	45	46
1	17	17	17	19	20	20
2	9	9	9	11	11	11
3	5	5	5	7	7	7
4	3	3	3	5	5	5
5	2	2	2	3	3	3
6	2	2	2	3	2	2
7	1	1	1	2	2	2
8	1	1	1	1	1	1
9	1	1	1	1	1	1
10	1	1	1	1	1	1
11	1	1	1	1	1	1
12	1	1	1	1	1	0
13	1	1	0	1	0	0
14	1	0	0	0	0	0
15	1	0	0	0	0	0
16	0	0	0	0	0	0
17	0	0	0	0	0	0
18	0	0	0	0	0	0
19	0	0	0	0	0	0
20	0	0	0	0	0	0
21	0	0	0	0	0	0
22	0	0	0	0	0	0
23	0	0	0	0	0	0

Tablo A.4 (devam): Güneşe bağlı ışınım zaman faktörleri (Owen 2009).

Yapı Türü	**Orta**					
	Halı var			Halı yok		
Cam %	%10	%50	%90	%10	%50	%90
Saat	**Işınım Zaman Faktörü [%]**					
0	52	54	55	28	29	29
1	16	16	15	15	15	15
2	8	8	8	10	10	10
3	5	4	4	7	7	7
4	3	3	3	6	6	6
5	2	2	2	5	5	5
6	2	1	1	4	4	4
7	1	1	1	4	3	3
8	1	1	1	3	3	3
9	1	1	1	3	3	3
10	1	1	1	2	2	2
11	1	1	1	2	2	2
12	1	1	1	2	2	2
13	1	1	1	2	2	2
14	1	1	1	1	1	1
15	1	1	1	1	1	1
16	1	1	1	1	1	1
17	1	1	1	1	1	1
18	1	1	1	1	1	1
19	0	0	0	1	1	1
20	0	0	0	1	1	1
21	0	0	0	0	0	0
22	0	0	0	0	0	0
23	0	0	0	0	0	0

Tablo A.4 (devam): Güneşe bağlı ışınım zaman faktörleri (Owen 2009).

Yapı Türü	**Ağır**					
	Halı var			Halı yok		
Cam %	%10	%50	%90	%10	%50	%90
Saat	**Işınım Zaman Faktörü [%]**					
0	47	49	51	26	27	28
1	11	12	12	12	13	13
2	6	6	6	7	7	7
3	4	4	3	5	5	5
4	3	3	3	4	4	4
5	2	2	2	4	4	4
6	2	2	2	3	3	3
7	2	2	2	3	3	3
8	2	2	2	3	3	3
9	2	2	2	3	3	3
10	2	2	2	3	3	3
11	2	2	1	3	3	2
12	2	1	1	2	2	2
13	2	1	1	2	2	2
14	2	1	1	2	2	2
15	1	1	1	2	2	2
16	1	1	1	2	2	2
17	1	1	1	2	2	2
18	1	1	1	2	2	2
19	1	1	1	2	2	2
20	1	1	1	2	2	2
21	1	1	1	2	2	2
22	1	1	1	2	1	1
23	1	1	1	2	1	1

Tablo A.5: Belli duvar tipleri için ısıl özellikler ve yapı bileşenleri (Owen 2009).

Tip	**Perde Duvar**		
No	1	2	3
Isı Transfer Katsayısı	0,428 W/m^2K	0,429 W/m^2K	0,428 W/m^2K
Isıl Direnç	2,3 m^2K/W	2,3 m^2K/W	2,3 m^2K/W
Kütle Yoğunluğu	31 kg/m^2	20,9 kg/m^2	80 kg/m^2
Isıl Kapasite	30,7 kJ/m^2K	20,4 kJ/m^2K	67,5 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Opak spandrel cam 6,4 mm	Yumuşak çelik yüzey 0,8 mm	Granit taş 25,4 mm
	Duvar hava boşluğu direnci	Duvar hava boşluğu direnci	Duvar hava boşluğu direnci
	EPS yalıtım levhası 50,8 mm	EPS yalıtım levhası 50,8 mm	EPS yalıtım levhası 50,8 mm
	Duvar hava boşluğu direnci	Duvar hava boşluğu direnci	Duvar hava boşluğu direnci
	Alçı levha 15,9 mm	Alçı levha 15,9 mm	Alçı levha 15,9 mm
	İç dikey yüzey direnci	İç dikey yüzey direnci	İç dikey yüzey direnci
Tip	**Saplama Duvar**		
No	4	5	6
Isı Transfer Katsayısı	0,419 W/m^2K	0,417 W/m^2K	0,406 W/m^2K
Isıl Direnç	2,4 m^2K/W	2,4 m^2K/W	2,5 m^2K/W
Kütle Yoğunluğu	25,5 kg/m^2	84,6 kg/m^2	25,6 kg/m^2
Isıl Kapasite	24,5 kJ/m^2K	73,6 kJ/m^2K	32,7 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Yumuşak çelik yüzey 0,8 mm	Granit taş 25,4 mm	Ahşap kaplama 12,7 mm
	Fiber levha kaplama 12,7 mm	Fiber levha kaplama 12,7 mm	Kontrplak 15,9 mm
	Cam fiber örgü yalıtım 89,4 mm	Cam fiber örgü yalıtım 89,4 mm	Cam fiber örgü yalıtım 89,4 mm
	Alçı levha 15,9 mm	Alçı levha 15,9 mm	Ahşap 12,7 mm
	İç dikey yüzey direnci	İç dikey yüzey direnci	İç dikey yüzey direnci

Tablo A.5 (devam): Belli duvar tipleri için ısıl özellikler ve yapı bileşenleri (Owen 2009).

Tip	**Saplama Duvar**	**Kompozit Yalıtımlı Duvar**	
No	7	8	9
Isı Transfer Katsayısı	0,413 W/m^2K	0,668 W/m^2K	0,305 W/m^2K
Isıl Direnç	2,4 m^2K/W	1,5 m^2K/W	3,3 m^2K/W
Kütle Yoğunluğu	66,7 kg/m^2	36,6 kg/m^2	38,3 kg/m^2
Isıl Kapasite	61,3 kJ/m^2K	36,7 kJ/m^2K	38,8 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Sıva 25,4 mm	Kompozit yalıtımlı panel 9,5 mm	Kompozit yalıtımlı panel 9,5 mm
	Fiber levha kaplama 12,7 mm	EPS yalıtım levhası 25,4 mm	EPS yalıtım levhası 25,4 mm
	Cam fiber örgü yalıtım 89,4 mm	Fiber levha kaplama 12,7 mm	Fiber levha kaplama 12,7 mm
	Alçı levha 15,9 mm	Duvar hava boşluğu direnci	Cam fiber örgü yalıtım 89,4 mm
	İç dikey yüzey direnci	Alçı levha 15,9 mm	Alçı levha 15,9 mm
		İç dikey yüzey direnci	İç dikey yüzey direnci
Tip	**Kompozit Yalıtımlı Duvar**	**Tuğla Duvar**	
No	10	11	12
Isı Transfer Katsayısı	0,524 W/m^2K	0,571 W/m^2K	0,377 W/m^2K
Isıl Direnç	1,9 m^2K/W	1,7 m^2K/W	2,7 m^2K/W
Kütle Yoğunluğu	130,9 kg/m^2	214,1 kg/m^2	214,7 kg/m^2
Isıl Kapasite	120,6 kJ/m^2K	177,8 kJ/m^2K	177,8 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Kompozit yalıtımlı panel 9,5 mm	Ateş tuğlası 101,6 mm	Ateş tuğlası 101,6 mm
	EPS yalıtım levhası 25,4 mm	Duvar hava boşluğu direnci	Duvar hava boşluğu direnci
	Fiber levha kaplama 12,7 mm	EPS yalıtım levhası 25,4 mm	Fiber levha kaplama 12,7 mm
	Hafif beton blok 203,2 mm	Fiber levha kaplama 12,7 mm	Duvar hava boşluğu direnci
	Duvar hava boşluğu direnci	Duvar hava boşluğu direnci	Alçı levha 15,9 mm
	Alçı levha 15,9 mm	Alçı levha 15,9 mm	İç dikey yüzey direnci
	İç dikey yüzey direnci	İç dikey yüzey direnci	

Tablo A.5 (devam): Belli duvar tipleri için ısıl özellikler ve yapı bileşenleri (Owen 2009).

Tip	**Tuğla Duvar**		
No	13	14	15
Isı Transfer Katsayısı	0,283 W/m^2K	0,581 W/m^2K	0,348 W/m^2K
Isıl Direnç	3,5 m^2K/W	1,7 m^2K/W	2,9 m^2K/W
Kütle Yoğunluğu	215,8 kg/m^2	290,6 kg/m^2	304 kg/m^2
Isıl Kapasite	177,8 kJ/m^2K	239,1 kJ/m^2K	253,5 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Ateş tuğlası 101,6 mm	Ateş tuğlası 101,6 mm	Ateş tuğlası 101,6 mm
	Duvar hava boşluğu direnci	Duvar hava boşluğu direnci	Duvar hava boşluğu direnci
	EPS yalıtım levhası 25,4 mm	EPS yalıtım levhası 25,4 mm	Hafif beton blok 203,2 mm
	Fiber levha kaplama 12,7 mm	Hafif beton blok 203,2 mm	Cam fiber örgü yalıtım 89,4 mm
	Cam fiber örgü yalıtım 89,4 mm	İç dikey yüzey direnci	Alçı levha 15,9 mm
	Alçı levha 15,9 mm		İç dikey yüzey direnci
	İç dikey yüzey direnci		
Tip	**Tuğla Duvar**		
No	16	17	18
Isı Transfer Katsayısı	0,628 W/m^2K	0,702 W/m^2K	0,514 W/m^2K
Isıl Direnç	1,6 m^2K/W	1,4 m^2K/W	1,9 m^2K/W
Kütle Yoğunluğu	371,7 kg/m^2	391,5 kg/m^2	469,3 kg/m^2
Isıl Kapasite	320,9 kJ/m^2K	312,7 kJ/m^2K	388,4 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Ateş tuğlası 101,6 mm	Ateş tuğlası 101,6 mm	Ateş tuğlası 101,6 mm
	Duvar hava boşluğu direnci	Duvar hava boşluğu direnci	Duvar hava boşluğu direnci
	EPS yalıtım levhası 25,4 mm	EPS yalıtım levhası 25,4 mm	EPS yalıtım levhası 25,4 mm
	Beton blok 203,2 mm	Ateş tuğlası 101,6 mm	Hafif beton 203,2 mm
	Alçı levha 15,9 mm	İç dikey yüzey direnci	Duvar hava boşluğu direnci
	İç dikey yüzey direnci		Alçı levha 15,9 mm
			İç dikey yüzey direnci

Tablo A.5 (devam): Belli duvar tipleri için ısıl özellikler ve yapı bileşenleri (Owen 2009).

Tip	**Tuğla Duvar**		**Beton Blok Duvar**
No	19	20	21
Isı Transfer Katsayısı	0,581 W/m^2K	0,389 W/m^2K	0,383 W/m^2K
Isıl Direnç	1,7 m^2K/W	2,6 m^2K/W	2,6 m^2K/W
Kütle Yoğunluğu	892,2 kg/m^2	665,1 kg/m^2	108,8 kg/m^2
Isıl Kapasite	784,9 kJ/m^2K	580,5 kJ/m^2K	98,1 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Ateş tuğlası 101,6 mm	Ateş tuğlası 101,6 mm	Hafif beton blok 203,2 mm
	Duvar hava boşluğu direnci	Duvar hava boşluğu direnci	Cam fiber örgü yalıtım 89,4 mm
	EPS yalıtım levhası 25,4 mm	Ağır beton 304,8 mm	Alçı levha 15,9 mm
	Ağır beton 304,8 mm	Cam fiber örgü yalıtım 89,4 mm	İç dikey yüzey direnci
	Duvar hava boşluğu direnci	Alçı levha 15,9 mm	
	Alçı levha 15,9 mm	İç dikey yüzey direnci	
	İç dikey yüzey direnci		
Tip	**Beton Blok Duvar**		
No	22	23	24
Isı Transfer Katsayısı	0,335 W/m^2K	0,414 W/m^2K	1,056 W/m^2K
Isıl Direnç	3 m^2K/W	2,4 m^2K/W	0,9 m^2K/W
Kütle Yoğunluğu	108,8 kg/m^2	224,3 kg/m^2	94,3 kg/m^2
Isıl Kapasite	98,1 kJ/m^2K	204,4 kJ/m^2K	83,8 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Hafif beton blok (dolgulu) 203,2 mm	Sıva 25,4 mm	Hafif beton blok (dolgulu) 203,2 mm
	Cam fiber örgü yalıtım 89,4 mm	Beton blok 203,2 mm	İç dikey yüzey direnci
	Alçı levha 15,9 mm	Cam fiber örgü yalıtım 89,4 mm	
	İç dikey yüzey direnci	Alçı levha 15,9 mm	
		İç dikey yüzey direnci	

Tablo A.5 (devam): Belli duvar tipleri için ısıl özellikler ve yapı bileşenleri (Owen 2009).

Tip	**Beton Blok Duvar**		**Öndökümlü ve Yerinde Dökme Duvar**
No	25	26	27
Isı Transfer Katsayısı	0,834 W/m^2K	0,689 W/m^2K	0,673 W/m^2K
Isıl Direnç	1,2 m^2K/W	1,5 m^2K/W	1,5 m^2K/W
Kütle Yoğunluğu	107,1 kg/m^2	168,9 kg/m^2	143,9 kg/m^2
Isıl Kapasite	96,1 kJ/m^2K	151,3 kJ/m^2K	124,27 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Hafif beton blok (dolgulu) 203,2 mm	Hafif beton blok (dolgulu) 304,8 mm	Hafif beton 101,6 mm
	Duvar hava boşluğu direnci	Duvar hava boşluğu direnci	EPS yalıtım levhası 25,4 mm
	Alçı levha 15,9 mm	Alçı levha 15,9 mm	Duvar hava boşluğu direnci
	İç dikey yüzey direnci	İç dikey yüzey direnci	Alçı levha 15,9 mm
			İç dikey yüzey direnci
Tip	**Öndökümlü ve Yerinde Dökme Duvar**		
No	28	29	30
Isı Transfer Katsayısı	0,418 W/m^2K	0,434 W/m^2K	0,65 W/m^2K
Isıl Direnç	2,4 m^2K/W	2,3 m^2K/W	1,5 m^2K/W
Kütle Yoğunluğu	144,6 kg/m^2	262,5 kg/m^2	291,8 kg/m^2
Isıl Kapasite	124,7 kJ/m^2K	220,8 kJ/m^2K	247,3 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Hafif beton 101,6 mm	Hafif beton 101,6 mm	Kompozit yalıtımlı panel 9,5 mm
	Cam fiber örgü yalıtım 89,4 mm	EPS yalıtım levhası 50,8 mm	EPS yalıtım levhası 25,4 mm
	Alçı levha 15,9 mm	Hafif beton 101,6 mm	Hafif beton 203,2 mm
	İç dikey yüzey direnci	İç dikey yüzey direnci	Alçı levha 15,9 mm
			İç dikey yüzey direnci

Tablo A.5 (devam): Belli duvar tipleri için ısıl özellikler ve yapı bileşenleri (Owen 2009).

Tip	**Öndökümlü ve Yerinde Dökme Duvar**		
No	31	32	33
Isı Transfer Katsayısı	0,387 W/m^2K	0,467 W/m^2K	0,434 W/m^2K
Isıl Direnç	2,6 m^2K/W	2,1 m^2K/W	2,3 m^2K/W
Kütle Yoğunluğu	274,7 kg/m^2	488,1 kg/m^2	469,9 kg/m^2
Isıl Kapasite	233 kJ/m^2K	441,5 kJ/m^2K	425,2 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Hafif beton 203,2 mm	Kompozit yalıtımlı panel 9,5 mm	Ağır beton 203,2 mm
	Cam fiber örgü yalıtım 89,4 mm	EPS yalıtım levhası 50,8 mm	Cam fiber örgü yalıtım 89,4 mm
	Alçı levha 15,9 mm	Ağır beton 203,2 mm	Alçı levha 15,9 mm
	İç dikey yüzey direnci	Alçı levha 15,9 mm	İç dikey yüzey direnci
		İç dikey yüzey direnci	
Tip	**Öndökümlü ve Yerinde Dökme Duvar**		
No	34	35	
Isı Transfer Katsayısı	0,266 W/m^2K	3,122 W/m^2K	
Isıl Direnç	3,8 m^2K/W	0,3 m^2K/W	
Kütle Yoğunluğu	698,9 kg/m^2	683,2 kg/m^2	
Isıl Kapasite	631,6 kJ/m^2K	615,2 kJ/m^2K	
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	
	Ağır beton 304,2 mm	Ağır beton 304,2 mm	
	Cam fiber örgü yalıtım 154,4 mm	İç dikey yüzey direnci	
	Alçı levha 15,9 mm		
	İç dikey yüzey direnci		

Tablo A.6: Belli çatı tipleri için ısıl özellikler ve yapı bileşenleri (Owen 2009).

Tip	**Çerçeve Çatı**		
No	1	2	3
Isı Transfer Katsayısı	0,249 W/m^2K	0,227 W/m^2K	0,255 W/m^2K
Isıl Direnç	4 m^2K/W	4,4 m^2K/W	3,9 m^2K/W
Kütle Yoğunluğu	26,7 kg/m^2	21 kg/m^2	14 kg/m^2
Isıl Kapasite	26,6 kJ/m^2K	16,4 kJ/m^2K	12,3 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Yumuşak çelik yüzey 0,8 mm	Yumuşak çelik yüzey 0,8 mm	Yumuşak çelik yüzey 0,8 mm
	Fiber levha kaplama 12,7 mm	Fiber levha kaplama 12,7 mm	Fiber levha kaplama 12,7 mm
	Tavan hava boşluğu direnci	Tavan hava boşluğu direnci	Tavan hava boşluğu direnci
	Cam fiber örgü yalıtım 154,4 mm	Cam fiber örgü yalıtım 154,4 mm	Cam fiber örgü yalıtım 154,4 mm
	Alçı levha 15,9 mm	Tavan hava boşluğu direnci	İç yatay yüzey direnci
	İç yatay yüzey direnci	Mineral fiber akustik karo 19,1 mm	
		İç yatay yüzey direnci	
Tip	**Çerçeve Çatı**		
No	4	5	6
Isı Transfer Katsayısı	0,235 W/m^2K	0,239 W/m^2K	0,231 W/m^2K
Isıl Direnç	4,2 m^2K/W	4,2 m^2K/W	4,3 m^2K/W
Kütle Yoğunluğu	34,7 kg/m^2	55,5 kg/m^2	34,9 kg/m^2
Isıl Kapasite	47 kJ/m^2K	73,5 kJ/m^2K	47 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Asfalt kiremit 3,2 mm	Kayrak karo 12,7 mm	Ahşap kiremit 6,4 mm
	Ahşap 25,4 mm	Ahşap 25,4 mm	Ahşap 25,4 mm
	Tavan hava boşluğu direnci	Tavan hava boşluğu direnci	Tavan hava boşluğu direnci
	Cam fiber örgü yalıtım 154,4 mm	Cam fiber örgü yalıtım 154,4 mm	Cam fiber örgü yalıtım 154,4 mm
	Tavan hava boşluğu direnci	Tavan hava boşluğu direnci	Tavan hava boşluğu direnci
	Alçı levha 15,9 mm	Alçı levha 15,9 mm	Alçı levha 15,9 mm
	İç yatay yüzey direnci	İç yatay yüzey direnci	İç yatay yüzey direnci

Tablo A.6 (devam): Belli çatı tipleri için ısıl özellikler ve yapı bileşenleri (Owen 2009).

Tip	**Ahşap Döşeme**		**Metal Döşeme**
No	7	8	9
Isı Transfer Katsayısı	0,393 W/m^2K	0,329 W/m^2K	0,452 W/m^2K
Isıl Direnç	2,5 m^2K/W	3 m^2K/W	2,2 m^2K/W
Kütle Yoğunluğu	48,9 kg/m^2	55,9 kg/m^2	23,9 kg/m^2
Isıl Kapasite	75,6 kJ/m^2K	79,7 kJ/m^2K	28,6 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Birleşik çatılama 9,5 mm	Birleşik çatılama 9,5 mm	
	Fiber levha kaplama 12,7 mm	Fiber levha kaplama 12,7 mm	
	EPS yalıtım levhası 50,8 mm	EPS yalıtım levhası 50,8 mm	
	Ahşap 50,8 mm	Ahşap 50,8 mm	
	İç yatay yüzey direnci	Tavan hava boşluğu direnci	
		Mineral fiber akustik karo 19,1 mm	
		İç yatay yüzey direnci	
Tip	**Metal Döşeme**		
No	10	11	12
Isı Transfer Katsayısı	0,37 W/m^2K	0,323 W/m^2K	0,206 W/m^2K
Isıl Direnç	2,7 m^2K/W	3,1 m^2K/W	4,9 m^2K/W
Kütle Yoğunluğu	30,9 kg/m^2	25 kg/m^2	27,2 kg/m^2
Isıl Kapasite	32,7 kJ/m^2K	28,6 kJ/m^2K	32,7 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Birleşik çatılama 9,5 mm	Birleşik çatılama 9,5 mm	Birleşik çatılama 9,5 mm
	Fiber levha kaplama 12,7 mm	Fiber levha kaplama 12,7 mm	Fiber levha kaplama 12,7 mm
	EPS yalıtım levhası 50,8 mm	EPS yalıtım levhası 50,8 mm	EPS yalıtım levhası 50,8 mm
	Yumuşak çelik yüzey 0,8 mm	Yumuşak çelik yüzey 0,8 mm	EPS yalıtım levhası 76,2 mm
	İç yatay yüzey direnci	Tavan hava boşluğu direnci	Yumuşak çelik yüzey 0,8 mm
		Mineral fiber akustik karo 19,1 mm	
		İç yatay yüzey direnci	

Tablo A.6 (devam): Belli çatı tipleri için ısıl özellikler ve yapı bileşenleri (Owen 2009).

Tip	**Metal Döşeme**	**Beton Çatı**	
No	13	14	15
Isı Transfer Katsayısı	0,297 W/m^2K	0,304 W/m^2K	0,296 W/m^2K
Isıl Direnç	3,4 m^2K/W	3,3 m^2K/W	3,4 m^2K/W
Kütle Yoğunluğu	57,6 kg/m^2	149,2 kg/m^2	214,3 kg/m^2
Isıl Kapasite	57,2 kJ/m^2K	134,9 kJ/m^2K	190,1 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Hafif beton çatı balastı 50,8 mm	Birleşik çatılama 9,5 mm	Birleşik çatılama 9,5 mm
	Birleşik çatılama 9,5 mm	Fiber levha kaplama 12,7 mm	Fiber levha kaplama 12,7 mm
	Fiber levha kaplama 12,7 mm	EPS yalıtım levhası 76,2 mm	EPS yalıtım levhası 76,2 mm
	EPS yalıtım levhası 76,2 mm	Hafif beton 101,6 mm	Hafif beton 152,4 mm
	Yumuşak çelik yüzey 0,8 mm	İç yatay yüzey direnci	İç yatay yüzey direnci
	İç yatay yüzey direnci		
Tip	**Metal Döşeme**		
No	16	17	18
Isı Transfer Katsayısı	0,288 W/m^2K	0,315 W/m^2K	0,313 W/m^2K
Isıl Direnç	3,5 m^2K/W	3,2 m^2K/W	3,2 m^2K/W
Kütle Yoğunluğu	279,3 kg/m^2	360,7 kg/m^2	474,5 kg/m^2
Isıl Kapasite	245,2 kJ/m^2K	333,2 kJ/m^2K	437,4 kJ/m^2K
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci	Dış yüzey direnci	Dış yüzey direnci
	Birleşik çatılama 9,5 mm	Birleşik çatılama 9,5 mm	Birleşik çatılama 9,5 mm
	Fiber levha kaplama 12,7 mm	Fiber levha kaplama 12,7 mm	Fiber levha kaplama 12,7 mm
	EPS yalıtım levhası 76,2 mm	EPS yalıtım levhası 76,2 mm	EPS yalıtım levhası 76,2 mm
	Hafif beton 203,2 mm	Ağır beton 152,4 mm	Ağır beton 203,2 mm
	İç yatay yüzey direnci	İç yatay yüzey direnci	İç yatay yüzey direnci
Tip	**Beton Çatı**		
No	19		
Isı Transfer Katsayısı	0,239 W/m^2K		
Isıl Direnç	4,2 m^2K/W		
Kütle Yoğunluğu	362,3 kg/m^2		
Isıl Kapasite	331,1 kJ/m^2K		
Yapı Bileşenleri (Dıştan İçe)	Dış yüzey direnci		
	Birleşik çatılama 9,5 mm		
	Ağır beton 152,4 mm		
	Tavan hava boşluğu direnci		
	EPS yalıtım levhası 154,4 mm		
	Mineral fiber akustik karo 19,1 mm		
	İç yatay yüzey direnci		

EK B Alindair 20S Direkt Buharlaşmalı Soğutucu Katalog Bilgileri

Tablo B.1: Alindair 20S direkt buharlaşmalı soğutucu katalog bilgileri (url-4).

Hava Debisi	20000 m^3/h
Motor Gücü	1,5 kW
Motor Akımı	8 A
Gerilim	220 V / 50 Hz
Fan Tipi	Aksiyal
Fan Çapı	ϕ7200
Fan Devri	1400 min^{-1} (Kademeli olarak değiştirilebilir)
Fan Basıncı	230 Pa
Soğutucu Ped Adedi	4
Soğutucu Ped Ölçüleri	885 mm x 980 mm x 100 mm
Toplam Soğutucu Ped Alanı	3,47 m^2
Tank Kapasitesi	40 *l*
Drenaj	Otomatik
Makine Hacim Ölçüleri	1250 mm x 1250 mm x 1300 mm
Ağırlık	125 kg
Çalıştırma Ağırlığı	165 kg
Kanal Ölçüleri	770 mm x 770 mm

EK C Program Kaynak Kodları

EK C.1 "fonksiyon.dll" Dosyası Psikrometrik Hesap Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Psikrometrik Hesap Formülleri
// 2009 ASHRAE Handbook-Fundamentals (SI) Chapter 1
//
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace fonksiyon
{
    public class Bolum1
    {
        public double t_zero = 273.15; // °C-->K dönüşümü
        public double getPatm(double z)
        { return 101325 * Math.Pow((1 - 2.25577 * z * Math.Pow(10, -
5)), 5.2559); } // Eqn. (3) Atmosfer basıncı
        public double getPsat(double tdry)
        {
            double Tabs = tdry + t_zero; // Mutlak sıcaklık (K)
            double c8, c9, c10, c11, c12, c13;
            c8 = -5.8002206E+03;
            c9 = 1.3914993;
            c10 = -4.8640239E-02;
            c11 = 4.1764768E-05;
            c12 = -1.4452093E-08;
            c13 = 6.5459673;
            return Math.Exp(c8 / Tabs + c9 + c10 * Tabs + c11 * Math.Pow(T
abs, 2) + c12 * Math.Pow(Tabs, 3) + c13 * Math.Log(Tabs)); //eqn. (6) Doym
a basıncı
        }
        public double getPvap(double Psat, double RelHum)
        { return RelHum * Psat; } // Eqn. (24) Kısmi buhar basıncı
        public double get_h(double tdry, double w)
        { return 1.006 * tdry + w * (2501 + 1.86 * tdry); } // Eqn. (32) Ö
zgül entalpi
        public double get_tdew(double Pvap)
        {
            double Pk = Pvap / 1000;
            double alpha = Math.Log(Pk); // Pa-->kPa dönüşümü
            double c14, c15, c16, c17, c18;
            c14 = 6.54;
            c15 = 14.526;
            c16 = 0.7389;
            c17 = 0.09486;
            c18 = 0.4569;
```

```
            return c14 + c15 * alpha + c16 * Math.Pow(alpha, 2) + c17 * Ma
th.Pow(alpha, 3) + c18 * Math.Pow(Pk, 0.1984); // Eqn. (39) Çiy noktası sı
caklığı
        }
        public double get_w(double Patm, double Pvap)
        { return 0.621945 * Pvap / (Patm - Pvap); } // Eqn. (22) Özgül nem
        public double get_w_from_h(double tdry, double h)
        { return (h - 1.006 * tdry) / (2501 -
 1.86 * tdry); } // Eqn. (32) Entalpi için özgül nem
        public double getPvap_from_w(double Patm, double w)
        { return w * Patm / (w + 0.621945); } // Eqn. (22) Özgül nem için
kısmi buhar basıncı
        public double getRelHum(double Pvap, double Psat)
        { return Pvap / Psat; } // Eqn. (24) Bağıl nem
        public double get_w_star(double tdry, double twet, double Patm)
        {
            double Psatstar = getPsat(twet);
            double w_s_star = get_w(Patm, Psatstar);
            return ((2501 - 2.326 * twet) * w_s_star - 1.006 * (tdry -
 twet)) / (2501 + 1.86 * tdry -
 4.186 * twet); // Eqn. (35) Yaş termometre sıcaklığı için özgül nem

        }
        public double get_v(double t, double Patm, double w)
        {
            double Tabs = t + t_zero; // Mutlak sıcaklık (K)
            double Pk = Patm / 1000; // Pa-->kPa dönüşümü
            return 0.287042 * Tabs * (1 + 1.607858 * w) / Pk; // Eqn. (28)
 Nemli havanın özgül hacmi
        }
        public double get_twet(double tdry, double tdew, double Patm, doub
le w)
        {
            double Wstar;
            double twetSup = tdry; // Üst tahmin değeri= Kuru termometre s
ıcaklığı
            double twetInf = tdew; // Alt tahmin değeri= Çiy noktası sıcak
lığı
            double twet = (twetSup + twetInf) / 2; // İlk tahmin değeri
            while (twetSup -
 twetInf > 0.001) // Yaş termometre sıcaklığı için iterasyon döngüsü
            {
                Wstar = get_w_star(tdry, twet, Patm);
                if (Wstar > w) twetSup = twet;
                else twetInf = twet;
                twet = (twetSup + twetInf) / 2;
            }
            return twet;
        }
    }
}
```

EK C.2 “fonksiyon.dll” Dosyası Güneş Işınımı Hesap Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Güneş Işınımı Hesap Formülleri
// 2009 ASHRAE Handbook-Fundamentals (SI) Chapter 14
//
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace fonksiyon
{
    public class Bolum14
    {
        // Cephe yönlerinin açı değerleri
        public double kuzey = 180;
        public double kuzeydogu = -135;
        public double dogu = -90;
        public double guneydogu = -45;
        public double guney = 0;
        public double guneybati = 45;
        public double bati = 90;
        public double kuzeybati = 135;
        //
        // Açık ve koyu renk için alpha/h0 değerleri
        public double acik_renk = 0.026;
        public double koyu_renk = 0.052;
        //
        public double DegToRad(double angle)
        { return angle * Math.PI / 180; } // Açıyı dereceden radyana çevir
ir.
        public double RadToDeg(double angle)
        { return angle * 180 / Math.PI; } // Açıyı radyandan dereceye çevi
rir.
        public double getAST(int LST, double ET, double LON, double LSM)
        { return LST + ET / 60 + (LON -
 LSM) / 15; } // eqn. (7) Belirgin güneş zamanı
        public double getH(double AST)
        { return 15 * (AST - 12); } // eqn. (11) Saat açısı
        public double getBeta(double L, double Delta, double H)
        {
            double Lrad = DegToRad(L);
            double Deltarad = DegToRad(Delta);
            double Hrad = DegToRad(H);
            double Betarad = Math.Asin(Math.Cos(Lrad) * Math.Cos(Deltarad)
 * Math.Cos(Hrad) + Math.Sin(Lrad) * Math.Sin(Deltarad)); // eqn. (12) Gün
eş irtifa açısı
            return RadToDeg(Betarad);
        }
        public double getPhi(double L, double Delta, double H, double Beta
)
```

```
        {
            double Phirad;
            double Lrad = DegToRad(L);
            double Deltarad = DegToRad(Delta);
            double Hrad = DegToRad(H);
            double Betarad = DegToRad(Beta);
            double sinPhi = Math.Sin(Hrad) * Math.Cos(Deltarad) / Math.Cos
(Betarad); // Eqn. (14)
            double cosPhi = (Math.Cos(Hrad) * Math.Cos(Deltarad) * Math.Si
n(Lrad) -
 Math.Sin(Deltarad) * Math.Cos(Lrad)) / Math.Cos(Betarad); // Eqn. (15)
            // Açı değerini -
180° ile +180° arasında olacak şekilde dönüştürür.
            if (sinPhi > 0 && cosPhi < 0)
            { Phirad = Math.Acos(cosPhi); }
            else if (sinPhi < 0 && cosPhi < 0)
            { Phirad = -Math.Acos(cosPhi); }
            else if (sinPhi < 0 && cosPhi > 0)
            { Phirad = Math.Asin(sinPhi); }
            else { Phirad = Math.Acos(cosPhi); }
            //
            return RadToDeg(Phirad); // Güneş azimut açısı
        }
        public double get_m(double Beta)
        {
            double m;
            double Betarad = DegToRad(Beta);
            m = 1 / (Math.Sin(Betarad) + 0.50572 * Math.Pow(6.07995 + Beta
, -1.6364)); // eqn. (16) Bağıl hava kütlesi
            if (double.IsNaN(m)) return 0; // Bağıl hava kütlesinin sonsuz
 değer almasını engeller.
            else return m;
        }
        public double get_ab(double taub, double taud)
        { return 1.219 - 0.043 * taub - 0.151 * taud -
 0.204 * taub * taud; } // eqn. (19) ab üssü
        public double get_ad(double taub, double taud)
        { return 0.202 - 0.852 * taub - 0.007 * taud -
 0.357 * taub * taud; } // eqn. (20) ad üssü
        public double getEb(double E0, double m, double taub, double ab)
        { return E0 * Math.Exp(-
taub * Math.Pow(m, ab)); } // eqn. (17) Direkt güneş ışınımı
        public double getEd(double E0, double m, double taud, double ad)
        { return E0 * Math.Exp(-
taud * Math.Pow(m, ad)); } // eqn. (18) Yayılan güneş ışınımı
        public double getGamma(double Phi, double Psi)
        {
            double Gamma = Phi - Psi; // Eqn. (21) Yüzey-
güneş azimut açısı
            if (Gamma > -360 && Gamma < -180)
            { return 360 + Gamma; }
            else { return Gamma; }
        }
        public double getThetaVer(double Beta, double Gamma)
        {
            double Betarad = DegToRad(Beta);
            double Gammarad = DegToRad(Gamma);
            double Thetarad = Math.Acos(Math.Cos(Betarad) * Math.Cos(Gamma
rad)); // eqn. (23) Dikey yüzeyler için güneş geliş açısı
```

```
            return RadToDeg(Thetarad);
        }
        public double getThetaHor(double Beta)
        { return 90 -
 Beta; } // eqn. (24) Yatay yüzeyler için güneş geliş açısı
        public double getEtb(double Eb, double Theta)
        {
            double Thetarad = DegToRad(Theta);
            if (Math.Cos(Thetarad) > 0)
            { return Eb * Math.Cos(Thetarad); } // eqn. (26) Yüzeye gelen
direkt güneş ışınımı
            else { return 0; } // Negatif değer almasını engeller.
        }
        public double getEtd(double Ed, double Theta, double Sigma) // Yüz
eye gelen yayılan güneş ışınımı
        {
            double Thetarad = DegToRad(Theta);
            double Sigmarad = DegToRad(Sigma);
            double Y = Math.Max(0.45, 0.55 + 0.437 * Math.Cos(Thetarad) +
0.313 * Math.Pow(Math.Cos(Thetarad), 2)); // eqn. (28)
            if (Sigma == 90) return Ed * Y; // eqn. (27)
            else if (Sigma < 90) return Ed * (Y * Math.Sin(Sigmarad) + Mat
h.Cos(Sigmarad)); // eqn. (29)
            else return Ed * Y * Math.Sin(Sigmarad); // eqn. (30)
        }
        public double getEtrHor = 0; // eqn. (31) Yatay yüzeye yerden yans
ıyan güneş ışınımı (Yatay yüzeyler için cosSigma=1)
        public double getEtrVer(double Eb, double Beta, double Ed) // Dike
y yüzeye yerden yansıyan güneş ışınımı
        {
            double Betarad = DegToRad(Beta);
            double Rhog = 0.2; // Table 5 Genel Kullanım 0.2
            return (Eb * Math.Sin(Betarad) + Ed) * Rhog / 2; // eqn. (31)
Dikey yüzeyler için cosSigma=0
        }
        public double GetEt(double Etb, double Etd, double Etr)
        { return Etb + Etd + Etr; } // eqn. (25)
    }
}
```

EK C.3 “fonksiyon.dll” Dosyası Soğutma Yükü Hesap Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Soğutma Yükü Hesap Formülleri
// 2009 ASHRAE Handbook-Fundamentals (SI) Chapter 18
//
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace fonksiyon
{
    public class Bolum18
    {
        // Ortak değişkenler
        public double acik_renk = 0.026;
        public double koyu_renk = 0.052;
        //
        public double get_teVer(double t_dis, double Et, double alphah0) /
/ Dikey yüzeyler için eşdeğer sıcaklık
        {
            return t_dis + alphah0 * Et; // Eqn. (30) Dikey yüzeyler için
(EpsilonDeltaR=0)
        }
        public double get_teHor(double t_dis, double Et, double alphah0) /
/ Yatay yüzeyler için eşdeğer sıcaklık
        {
            // ASHRAE tarafından yatay yüzeyler için önerilen katsayılar
            double Epsilon = 1;
            double h0 = 17;
            double DeltaR = 63;
            //
            return t_dis + alphah0 * Et -
 Epsilon * DeltaR / h0; // Eqn. (30) Yatay yüzeyler için eşdeğer sıcaklık
        }
        public double get_qi(double U, double A, double te, double t_ic)
        {
            return U * A * (te -
 t_ic); // Eqn. (31) Duvar ve çatıdan ısı girişi
        }
        public double get_qbeam(double A, double Etb, double shgc)
        {
            return A * Etb * shgc; // Eqn. (13) Pencere ve kapı için direk
t güneş ısı kazancı
        }
        public double get_qdiffuse(double A, double Etd, double Etr, doubl
e shgc_d)
        {
            return A * (Etd + Etr) * shgc_d; // Eqn. (14) Pencere va kapı
için yayılan gübeş ısı kazancı
        }
```

```
        public double get_qem1(double p, double verim) // Cihaz ısı eşdeğe
ri (Motor ve tahrik edilen ünite içeride)
        {
            return (p / verim); // Eqn. (2) Genel kullanımda Fum=1 Flm=1
        }
        public double get_qem2(double p) // Cihaz ısı eşdeğeri (Motor dışa
rıda, tahrik edilen ünite içeride)
        {
            return p; // Eqn. (3) Genel kullanımda Fum=1 Flm=1
        }
        public double get_qem3(double p, double verim) // Cihaz ısı eşdeğe
ri (Motor içeride, tahrik edilen ünite dışarıda)
        {
            return p * (1 -
 verim) / verim; // Eqn. (4) Genel kullanımda Fum=1 Flm=1
        }
    }
}
```

EK C.4 Ana Program Kaynak Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Ana Program Kaynak Kodları
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BuharlasmaliSogutucuHesapProgrami
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        bool Qt_kontrol = false; // Toplam soğutma yükü hesabı kontrol değ
işkeni
        private void Form1_Load(object sender, EventArgs e)
        {
        }
        // Sekmelerde adım adım ilerlemeyi sağlayan navigasyon kontrolleri
        private void tabControl1_Selecting(object sender, TabControlCancel
EventArgs e)
        {
            if (tabControl1.SelectedTab == Giris_tabPage)
            {
                giris1.Visible = true; panel1.Visible = false; aydinlatma1
.Visible = false;
                insanlar1.Visible = false; cihazlar1.Visible = false; qtop
lam1.Visible = false;
            }
            else if (tabControl1.SelectedTab == Yapi_tabPage)
            {
                if (giris1.q_var == true) tabControl1.SelectedTab = Kapasi
te_tabPage;
                else if (giris1.kontrol == false) tabControl1.SelectedTab
= Giris_tabPage;
                else
                {
                    giris1.Visible = false; panel1.Visible = true; aydinla
tma1.Visible = false;
                    insanlar1.Visible = false; cihazlar1.Visible = false;
qtoplam1.Visible = false; kapasite1.Visible = false;
                }
            }
            else if (tabControl1.SelectedTab == Aydinlatma_tabPage)
```

```
            {
                if (giris1.q_var == true) tabControl1.SelectedTab = Kapasi
te_tabPage;
                else if (giris1.kontrol == false) tabControl1.SelectedTab
= Giris_tabPage;
                else if (cati1.kontrol == false) tabControl1.SelectedTab =
 Yapi_tabPage;
                else
                {
                    giris1.Visible = false; panel1.Visible = false; aydinl
atma1.Visible = true;
                    insanlar1.Visible = false; cihazlar1.Visible = false;
qtoplam1.Visible = false; kapasite1.Visible = false;
                }
            }
            else if (tabControl1.SelectedTab == Insan_tabPage)
            {
                if (giris1.q_var == true) tabControl1.SelectedTab = Kapasi
te_tabPage;
                else if (giris1.kontrol == false) tabControl1.SelectedTab
= Giris_tabPage;
                else if (cati1.kontrol == false) tabControl1.SelectedTab =
 Yapi_tabPage;
                else if (aydinlatma1.kontrol == false) tabControl1.Selecte
dTab = Aydinlatma_tabPage;
                else
                {
                    giris1.Visible = false; panel1.Visible = false; aydinl
atma1.Visible = false;
                    insanlar1.Visible = true; cihazlar1.Visible = false; q
toplam1.Visible = false; kapasite1.Visible = false;
                }
            }
            else if (tabControl1.SelectedTab == Cihazlar_tabPage)
            {
                if (giris1.q_var == true) tabControl1.SelectedTab = Kapasi
te_tabPage;
                else if (giris1.kontrol == false) tabControl1.SelectedTab
= Giris_tabPage;
                else if (cati1.kontrol == false) tabControl1.SelectedTab =
 Yapi_tabPage;
                else if (aydinlatma1.kontrol == false) tabControl1.Selecte
dTab = Aydinlatma_tabPage;
                else if (insanlar1.kontrol == false) tabControl1.SelectedT
ab = Insan_tabPage;
                else
                {
                    giris1.Visible = false; panel1.Visible = false; aydinl
atma1.Visible = false;
                    insanlar1.Visible = false; cihazlar1.Visible = true; q
toplam1.Visible = false; kapasite1.Visible = false;
                }
            }
            else if (tabControl1.SelectedTab == Qt_tabPage)
            {
                if (giris1.q_var == true) tabControl1.SelectedTab = Kapasi
te_tabPage;
                else if (giris1.kontrol == false) tabControl1.SelectedTab
= Giris_tabPage;
```

```
                else if (cati1.kontrol == false) tabControl1.SelectedTab =
 Yapi_tabPage;
                else if (aydinlatma1.kontrol == false) tabControl1.Selecte
dTab = Aydinlatma_tabPage;
                else if (insanlar1.kontrol == false) tabControl1.SelectedT
ab = Insan_tabPage;
                else if (cihazlar1.kontrol == false) tabControl1.SelectedT
ab = Cihazlar_tabPage;
                else
                {
                    giris1.Visible = false; panel1.Visible = false; aydinl
atma1.Visible = false;
                    insanlar1.Visible = false; cihazlar1.Visible = false;
qtoplam1.Visible = true; kapasite1.Visible = false;
                    Qt_kontrol = true;
                }
            }
            else if (tabControl1.SelectedTab == Kapasite_tabPage)
            {
                if (giris1.kontrol == false && giris1.q_var == false) tabC
ontrol1.SelectedTab = Giris_tabPage;
                else if (giris1.q_var == false && giris1.kontrol == true)
                {
                    if (cati1.kontrol == false) tabControl1.SelectedTab =
Yapi_tabPage;
                    else if (aydinlatma1.kontrol == false) tabControl1.Sel
ectedTab = Aydinlatma_tabPage;
                    else if (insanlar1.kontrol == false) tabControl1.Selec
tedTab = Insan_tabPage;
                    else if (cihazlar1.kontrol == false) tabControl1.Selec
tedTab = Cihazlar_tabPage;
                    else if (Qt_kontrol == false) tabControl1.SelectedTab
= Qt_tabPage;
                    else
                    {
                        giris1.Visible = false; panel1.Visible = false; ay
dinlatma1.Visible = false;
                        insanlar1.Visible = false; cihazlar1.Visible = tru
e; qtoplam1.Visible = false; kapasite1.Visible = true;
                    }
                }
                else if (giris1.q_var == true || Qt_kontrol == true)
                {
                    giris1.Visible = false; panel1.Visible = false; aydinl
atma1.Visible = false;
                    insanlar1.Visible = false; cihazlar1.Visible = true; q
toplam1.Visible = false; kapasite1.Visible = true;
                }
            }
        }
        //
        // Yapı Elemanları sekmesindeki ağaç yapısında adım adım ilerlemey
i sağlayan navigasyon kontrolleri
        private void treeView1_AfterSelect(object sender, TreeViewEventArg
s e)
        {
            if (treeView1.Nodes[0].IsSelected)
            {
```

```
                cephe11.Visible = true; cephe21.Visible = false; cephe31.V
isible = false;
                cephe41.Visible = false; cati1.Visible = false;
                if (cephe11.kontrol == true)
                {
                    cephe11.cephe_radyasyon_groupBox.Visible = true; cephe
11.duvar_Q_groupBox.Visible = false;
                    cephe11.pencere_Q_groupBox.Visible = false; cephe11.ka
pi_Q_groupBox.Visible = false;
                }
            }
            if (treeView1.Nodes[0].Nodes[0].IsSelected)
            {
                cephe11.Visible = true; cephe21.Visible = false; cephe31.V
isible = false;
                cephe41.Visible = false; cati1.Visible = false;
                cephe11.cephe_radyasyon_groupBox.Visible = false; cephe11.
duvar_Q_groupBox.Visible = true;
                cephe11.pencere_Q_groupBox.Visible = false; cephe11.kapi_Q
_groupBox.Visible = false;
            }
            if (treeView1.Nodes[0].Nodes[1].IsSelected)
            {
                cephe11.Visible = true; cephe21.Visible = false; cephe31.V
isible = false;
                cephe41.Visible = false; cati1.Visible = false;
                cephe11.cephe_radyasyon_groupBox.Visible = false; cephe11.
duvar_Q_groupBox.Visible = false;
                cephe11.pencere_Q_groupBox.Visible = true; cephe11.kapi_Q_
groupBox.Visible = false;
            }
            if (treeView1.Nodes[0].Nodes[2].IsSelected)
            {
                cephe11.Visible = true; cephe21.Visible = false; cephe31.V
isible = false;
                cephe41.Visible = false; cati1.Visible = false;
                cephe11.cephe_radyasyon_groupBox.Visible = false; cephe11.
duvar_Q_groupBox.Visible = false;
                cephe11.pencere_Q_groupBox.Visible = false; cephe11.kapi_Q
_groupBox.Visible = true;
            }
            if (treeView1.Nodes[1].IsSelected)
            {
                if (cephe11.kontrol == false) treeView1.SelectedNode = tre
eView1.Nodes[0];
                else
                {
                    cephe11.Visible = false; cephe21.Visible = true; cephe
31.Visible = false;
                    cephe41.Visible = false; cati1.Visible = false;
                    if (cephe21.kontrol == true)
                    {
                        cephe21.cephe_radyasyon_groupBox.Visible = true; c
ephe21.duvar_Q_groupBox.Visible = false;
                        cephe21.pencere_Q_groupBox.Visible = false; cephe2
1.kapi_Q_groupBox.Visible = false;
                    }
                }
            }
```

```
            if (treeView1.Nodes[1].Nodes[0].IsSelected)
            {
                cephe11.Visible = false; cephe21.Visible = true; cephe31.V
isible = false;
                cephe41.Visible = false; cati1.Visible = false;
                cephe21.cephe_radyasyon_groupBox.Visible = false; cephe21.
duvar_Q_groupBox.Visible = true;
                cephe21.pencere_Q_groupBox.Visible = false; cephe21.kapi_Q
_groupBox.Visible = false;
            }
            if (treeView1.Nodes[1].Nodes[1].IsSelected)
            {
                cephe11.Visible = false; cephe21.Visible = true; cephe31.V
isible = false;
                cephe41.Visible = false; cati1.Visible = false;
                cephe21.cephe_radyasyon_groupBox.Visible = false; cephe21.
duvar_Q_groupBox.Visible = false;
                cephe21.pencere_Q_groupBox.Visible = true; cephe21.kapi_Q_
groupBox.Visible = false;
            }
            if (treeView1.Nodes[1].Nodes[2].IsSelected)
            {
                cephe11.Visible = false; cephe21.Visible = true; cephe31.V
isible = false;
                cephe41.Visible = false; cati1.Visible = false;
                cephe21.cephe_radyasyon_groupBox.Visible = false; cephe21.
duvar_Q_groupBox.Visible = false;
                cephe21.pencere_Q_groupBox.Visible = false; cephe21.kapi_Q
_groupBox.Visible = true;
            }
            if (treeView1.Nodes[2].IsSelected)
            {
                if (cephe21.kontrol == false) treeView1.SelectedNode = tre
eView1.Nodes[1];
                else
                {
                    cephe11.Visible = false; cephe21.Visible = false; ceph
e31.Visible = true;
                    cephe41.Visible = false; cati1.Visible = false;
                    if (cephe31.kontrol == true)
                    {
                        cephe31.cephe_radyasyon_groupBox.Visible = true; c
ephe31.duvar_Q_groupBox.Visible = false;
                        cephe31.pencere_Q_groupBox.Visible = false; cephe3
1.kapi_Q_groupBox.Visible = false;
                    }
                }
            }
            if (treeView1.Nodes[2].Nodes[0].IsSelected)
            {
                cephe11.Visible = false; cephe21.Visible = false; cephe31.
Visible = true;
                cephe41.Visible = false; cati1.Visible = false;
                cephe31.cephe_radyasyon_groupBox.Visible = false; cephe31.
duvar_Q_groupBox.Visible = true;
                cephe31.pencere_Q_groupBox.Visible = false; cephe31.kapi_Q
_groupBox.Visible = false;
            }
            if (treeView1.Nodes[2].Nodes[1].IsSelected)
```

```
            {
                cephe11.Visible = false; cephe21.Visible = false; cephe31.
Visible = true;
                cephe41.Visible = false; cati1.Visible = false;
                cephe31.cephe_radyasyon_groupBox.Visible = false; cephe31.
duvar_Q_groupBox.Visible = false;
                cephe31.pencere_Q_groupBox.Visible = true; cephe31.kapi_Q_
groupBox.Visible = false;
            }
            if (treeView1.Nodes[2].Nodes[2].IsSelected)
            {
                cephe11.Visible = false; cephe21.Visible = false; cephe31.
Visible = true;
                cephe41.Visible = false; cati1.Visible = false;
                cephe31.cephe_radyasyon_groupBox.Visible = false; cephe31.
duvar_Q_groupBox.Visible = false;
                cephe31.pencere_Q_groupBox.Visible = false; cephe31.kapi_Q
_groupBox.Visible = true;
            }
            if (treeView1.Nodes[3].IsSelected)
            {
                if (cephe31.kontrol == false) treeView1.SelectedNode = tre
eView1.Nodes[2];
                else
                {
                    cephe11.Visible = false; cephe21.Visible = false; ceph
e31.Visible = false;
                    cephe41.Visible = true; cati1.Visible = false;
                    if (cephe41.kontrol == true)
                    {
                        cephe41.cephe_radyasyon_groupBox.Visible = true; c
ephe41.duvar_Q_groupBox.Visible = false;
                        cephe41.pencere_Q_groupBox.Visible = false; cephe4
1.kapi_Q_groupBox.Visible = false;
                    }
                }
            }
            if (treeView1.Nodes[3].Nodes[0].IsSelected)
            {
                cephe11.Visible = false; cephe21.Visible = false; cephe31.
Visible = false;
                cephe41.Visible = true; cati1.Visible = false;
                cephe41.cephe_radyasyon_groupBox.Visible = false; cephe41.
duvar_Q_groupBox.Visible = true;
                cephe41.pencere_Q_groupBox.Visible = false; cephe41.kapi_Q
_groupBox.Visible = false;
            }
            if (treeView1.Nodes[3].Nodes[1].IsSelected)
            {
                cephe11.Visible = false; cephe21.Visible = false; cephe31.
Visible = false;
                cephe41.Visible = true; cati1.Visible = false;
                cephe41.cephe_radyasyon_groupBox.Visible = false; cephe41.
duvar_Q_groupBox.Visible = false;
                cephe41.pencere_Q_groupBox.Visible = true; cephe41.kapi_Q_
groupBox.Visible = false;
            }
            if (treeView1.Nodes[3].Nodes[2].IsSelected)
            {
```

```
                cephe11.Visible = false; cephe21.Visible = false; cephe31.
Visible = false;
                cephe41.Visible = true; cati1.Visible = false;
                cephe41.cephe_radyasyon_groupBox.Visible = false; cephe41.
duvar_Q_groupBox.Visible = false;
                cephe41.pencere_Q_groupBox.Visible = false; cephe41.kapi_Q
_groupBox.Visible = true;
            }
            if (treeView1.Nodes[4].IsSelected)
            {
                if (cephe41.kontrol == false) treeView1.SelectedNode = tre
eView1.Nodes[3];
                else
                {
                    cephe11.Visible = false; cephe21.Visible = false; ceph
e31.Visible = false;
                    cephe41.Visible = false; cati1.Visible = true;
                }
            }
        }
        private void treeView1_AfterExpand(object sender, TreeViewEventArg
s e)
        {
            if (treeView1.Nodes[0].IsExpanded)
            {
                if (cephe11.kontrol == false) treeView1.Nodes[0].Collapse(
);
            }
            if (treeView1.Nodes[1].IsExpanded)
            {
                if (cephe21.kontrol == false) treeView1.Nodes[1].Collapse(
);
            }
            if (treeView1.Nodes[2].IsExpanded)
            {
                if (cephe31.kontrol == false) treeView1.Nodes[2].Collapse(
);
            }
            if (treeView1.Nodes[3].IsExpanded)
            {
                if (cephe41.kontrol == false) treeView1.Nodes[3].Collapse(
);
            }
        }
        //
    }
}
```

EK C.5 Giriş Ekranı Kaynak Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Giriş Ekranı Kaynak Kodları
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BuharlasmaliSogutucuHesapProgrami
{
    public partial class Giris : UserControl
    {
        public Giris()
        {
            InitializeComponent();
        }
        // Fonksiyon dosyasını bağlar.
        fonksiyon.Bolum1 Ch1 = new fonksiyon.Bolum1();
        fonksiyon.Bolum14 Ch14 = new fonksiyon.Bolum14();
        //
        // Değişken tanımları
        int rts_id;
        double E0, ET, Delta, z, LON, L, taub, taud, ab, ad, tic, tdis_top
, tdis_ort;
        double RelHum_dis, qt;
        public bool kontrol = false; // Hesapla butonu tıklama kontrol değ
işkeni
        public bool q_var = false; // Mevcut soğutma yükünün girilip giril
mediğinin kontrolü
        //
        void veri_cek() // Metin kutularından verileri çekip veritabanına
kaydeder.
        {
            E0 = Convert.ToDouble(E0_label.Text);
            ET = Convert.ToDouble(ET_label.Text);
            Delta = Convert.ToDouble(Delta_label.Text);
            z = Convert.ToDouble(z_textBox.Text);
            LON = Convert.ToDouble(LON_textBox.Text);
            L = Convert.ToDouble(L_textBox.Text);
            taub = Convert.ToDouble(taub_textBox.Text);
            taud = Convert.ToDouble(taud_textBox.Text);
            RelHum_dis = Convert.ToDouble(RelHum_textBox.Text) / 100;
            tic = Convert.ToDouble(tic_textBox.Text);
            rts_id = Convert.ToInt16(rts_id_label.Text);
            // Veritabanından verileri çeker.
            this.girisTableAdapter.Fill(this.dataDataSet.Giris);
            this.radyasyonTableAdapter.Fill(this.dataDataSet.Radyasyon);
```

```
            //
            this.dataDataSet.Giris.Rows[0][2] = tic;
            this.dataDataSet.Giris.Rows[0][3] = z;
            this.dataDataSet.Giris.Rows[0][4] = RelHum_dis;
            this.dataDataSet.Giris.Rows[0][6] = rts_id;
            this.dataDataSet.Radyasyon.Rows[23][3] = Convert.ToDouble(LST2
4_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[0][3] = Convert.ToDouble(LST1_
textBox.Text);
            this.dataDataSet.Radyasyon.Rows[1][3] = Convert.ToDouble(LST2_
textBox.Text);
            this.dataDataSet.Radyasyon.Rows[2][3] = Convert.ToDouble(LST3_
textBox.Text);
            this.dataDataSet.Radyasyon.Rows[3][3] = Convert.ToDouble(LST4_
textBox.Text);
            this.dataDataSet.Radyasyon.Rows[4][3] = Convert.ToDouble(LST5_
textBox.Text);
            this.dataDataSet.Radyasyon.Rows[5][3] = Convert.ToDouble(LST6_
textBox.Text);
            this.dataDataSet.Radyasyon.Rows[6][3] = Convert.ToDouble(LST7_
textBox.Text);
            this.dataDataSet.Radyasyon.Rows[7][3] = Convert.ToDouble(LST8_
textBox.Text);
            this.dataDataSet.Radyasyon.Rows[8][3] = Convert.ToDouble(LST9_
textBox.Text);
            this.dataDataSet.Radyasyon.Rows[9][3] = Convert.ToDouble(LST10
_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[10][3] = Convert.ToDouble(LST1
1_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[11][3] = Convert.ToDouble(LST1
2_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[12][3] = Convert.ToDouble(LST1
3_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[13][3] = Convert.ToDouble(LST1
4_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[14][3] = Convert.ToDouble(LST1
5_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[15][3] = Convert.ToDouble(LST1
6_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[16][3] = Convert.ToDouble(LST1
7_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[17][3] = Convert.ToDouble(LST1
8_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[18][3] = Convert.ToDouble(LST1
9_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[19][3] = Convert.ToDouble(LST2
0_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[20][3] = Convert.ToDouble(LST2
1_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[21][3] = Convert.ToDouble(LST2
2_textBox.Text);
            this.dataDataSet.Radyasyon.Rows[22][3] = Convert.ToDouble(LST2
3_textBox.Text);
            // Dış sıcaklık değerlerinin ortalamasını hesaplar ve veritaba
nına kaydeder.
            tdis_top = 0;
            for (int i = 0; i < 24; i++) tdis_top += Convert.ToDouble(this
.dataDataSet.Radyasyon.Rows[i][3]);
            tdis_ort = tdis_top / 24;
```

```
            this.dataDataSet.Giris.Rows[0][1] = tdis_ort;
            //
            this.girisTableAdapter.Update(this.dataDataSet.Giris); // Veri
tabanını günceller.
        }
        void veri_cek2() // Metin kutularından verileri çekip veritabanına
 kaydeder.
        {
            z = Convert.ToDouble(z_textBox2.Text);
            tdis_ort = Convert.ToDouble(tdis_textBox.Text);
            RelHum_dis = Convert.ToDouble(RelHum_textBox2.Text) / 100;
            tic = Convert.ToDouble(tic_textBox2.Text);
            qt = Convert.ToDouble(qt_textBox.Text);
            // Veritabanından verileri çeker.
            this.girisTableAdapter.Fill(this.dataDataSet.Giris);
            this.radyasyonTableAdapter.Fill(this.dataDataSet.Radyasyon);
            //
            this.dataDataSet.Giris.Rows[0][1] = tdis_ort;
            this.dataDataSet.Giris.Rows[0][2] = tic;
            this.dataDataSet.Giris.Rows[0][3] = z;
            this.dataDataSet.Giris.Rows[0][4] = RelHum_dis;
            this.dataDataSet.Giris.Rows[0][5] = qt;
            this.girisTableAdapter.Update(this.dataDataSet.Giris); // Veri
tabanını günceller.
        }
        void radyasyon_hesap() // Saatlik güneş ışınımı hesaplamalarını ya
par.
        {
            ab = Ch14.get_ab(taub, taud); // ab üssü
            ad = Ch14.get_ad(taub, taud); // ad üssü
            for (int i = 0; i < 24; i++)
            {
                double AST = Ch14.getAST(i + 1, ET, LON, 30); // Belirgin
güneş zamanı (Türkiye için LSM=15)
                double H_aci = Ch14.getH(AST); // Saat açısı
                double Beta = Ch14.getBeta(L, Delta, H_aci); // Güneş irti
fa açısı
                double Phi = Ch14.getPhi(L, Delta, H_aci, Beta); // Güneş
azimut açısı
                double m = Ch14.get_m(Beta); // Bağıl hava kütlesi
                double Eb, Ed;
                if (Beta < 0) { Eb = 0; Ed = 0; } // Gece zamanı ışınım de
ğerlerini sıfır olarak belirler.
                else
                {
                    Eb = Ch14.getEb(E0, m, taub, ab); // Direkt güneş ışın
ımı
                    Ed = Ch14.getEd(E0, m, taud, ad); // Yayılan güneş ışı
nımı
                }
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Radyasyon.Rows[i][4] = AST;
                this.dataDataSet.Radyasyon.Rows[i][5] = H_aci;
                this.dataDataSet.Radyasyon.Rows[i][6] = Beta;
                this.dataDataSet.Radyasyon.Rows[i][7] = Phi;
                this.dataDataSet.Radyasyon.Rows[i][8] = m;
                this.dataDataSet.Radyasyon.Rows[i][9] = Eb;
                this.dataDataSet.Radyasyon.Rows[i][10] = Ed;
            }
```

```
            // Veritabanını günceller.
            this.radyasyonTableAdapter.Update(this.dataDataSet.Radyasyon);
            this.girisTableAdapter.Update(this.dataDataSet.Giris);
            //
        }
        private void Giris_Load(object sender, EventArgs e)
        {
            // Veritabanından verileri çeker.
            this.astronomikTableAdapter.Fill(this.dataDataSet.Astronomik);
            this.rTS_YapiTableAdapter.Fill(this.dataDataSet.RTS_Yapi);
            this.rTS_HaliTableAdapter.Fill(this.dataDataSet.RTS_Hali);
            this.rTS_CamTableAdapter.Fill(this.dataDataSet.RTS_Cam);
            this.girisTableAdapter.Fill(this.dataDataSet.Giris);
            this.radyasyonTableAdapter.Fill(this.dataDataSet.Radyasyon);
            //
        }
        // Metin kutularına veri girişinde sadece sayı, virgül, del ve bac
kspace tuşlarını aktif eder.
        // Tüm metin kutularına bu kod referans verilmiştir.
        private void z_textBox_KeyPress(object sender, KeyPressEventArgs e
)
        {
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)44 && e.KeyChar != (char)127;
        }
        //
        // Soğutma yükü hesabı ile mevcut soğutma yükü girişinin yapıldığı
 ekranlar arasında geçiş sağlar.
        private void q_var_radioButton_CheckedChanged(object sender, Event
Args e)
        {
            if (q_var_radioButton.Checked)
            {
                psikrometrik_groupBox.Visible = true; iklim_groupBox.Visib
le = false; tdis_groupBox.Visible = false;
                rts_groupBox.Visible = false; tic_baslik.Visible = false;
tic_textBox.Visible = false;
                hesapla_button.Visible = false; radyasyon_groupBox.Visible
 = false;
            }
            else if (q_yok_radioButton.Checked)
            {
                psikrometrik_groupBox.Visible = false; iklim_groupBox.Visi
ble = true; tdis_groupBox.Visible = true;
                rts_groupBox.Visible = true; tic_baslik.Visible = true; ti
c_textBox.Visible = true;
                hesapla_button.Visible = true;
                if (kontrol == true) radyasyon_groupBox.Visible = true;
            }
        }
        //
        private void hesapla_button_Click(object sender, EventArgs e) // S
oğutma yükü hesaplanmak istendiğinde tıklanacak buton
        {
            veri_cek();
            // Bağıl nem değerinin 0-
%100 arasında girilip girilmediğini kontrol eder.
            if (RelHum_dis < 0 || RelHum_dis > 1)
            {
```

```
                MessageBox.Show("Bağıl nem %0 ile %100 arasında olmalıdır.
", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                kontrol = false;
            }
            //
            // Ortalama dış ortam sıcaklığının iç ortam sıcaklığından büyü
k olup olmadığını kontrol eder.
            else if (tic > tdis_ort)
            {
                MessageBox.Show("Ortalama dış ortam sıcaklığı iç ortam sıc
aklığından büyük olmalıdır.", "Uyarı", MessageBoxButtons.OK, MessageBoxIco
n.Warning);
                kontrol = false;
            }
            //
            else
            {
                radyasyon_hesap();
                this.AutoScroll = true; // Sayfa kaydırmayı aktif eder.
                radyasyon_groupBox.Visible = true; // Hesaplanan değerleri
 ekranda gösterir.
                kontrol = true; // Hesapla butonu tıklama kontrolü
                q_var = false;
            }
        }
        private void hesapla_button2_Click(object sender, EventArgs e) //
Mevcut soğutma yükü girildiğinde tıklanacak buton
        {
            veri_cek2();
            // Bağıl nem değerinin 0-
%100 arasında girilip girilmediğini kontrol eder.
            if (RelHum_dis < 0 || RelHum_dis > 1)
            {
                MessageBox.Show("Bağıl nem %0 ile %100 arasında olmalıdır.
", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                q_var = false;
            }
            //
            // Girilen dış ortam sıcaklığının iç ortam sıcaklığından büyük
 olup olmadığını kontrol eder.
            else if (tic > tdis_ort)
            {
                MessageBox.Show("Girilen dış ortam sıcaklığı iç ortam sıca
klığından büyük olmalıdır.", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon
.Warning);
                q_var = false;
            }
            //
            else q_var = true; // Hesapla butonu tıklama kontrolü
        }
    }
}
```

EK C.6 Cephe1 Ekranı Kaynak Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Cephe1 Ekranı Kaynak Kodları
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BuharlasmaliSogutucuHesapProgrami
{
    public partial class Cephe1 : UserControl
    {
        public Cephe1()
        {
            InitializeComponent();
        }
        // Fonksiyon dosyasını bağlar.
        fonksiyon.Bolum14 Ch14 = new fonksiyon.Bolum14();
        fonksiyon.Bolum18 Ch18 = new fonksiyon.Bolum18();
        //
        // Değişken tanımları
        int rts_id, duvar_id, pencere_adet, kapi_adet;
        double t_ic, Psi, alphah0, duvar_en, duvar_boy, duvar_alan, Uduvar
, pencere_en, pencere_boy, pencere_alan, Upencere, kapi_en, kapi_boy, kapi
_alan, Ukapi;
        public bool kontrol = false; // Hesapla butonu tıklama kontrol değ
işkeni
        //
        void veri_cek() // Metin kutuları, radyo butonları ve veri tabanın
dan verileri çeker.
        {
            this.girisTableAdapter.Fill(this.dataDataSet.Giris);
            // Yüzey azimut açısını belirler.
            if (k_radioButton.Checked) Psi = Ch14.kuzey;
            else if (kd_radioButton.Checked) Psi = Ch14.kuzeydogu;
            else if (d_radioButton.Checked) Psi = Ch14.dogu;
            else if (gd_radioButton.Checked) Psi = Ch14.guneydogu;
            else if (g_radioButton.Checked) Psi = Ch14.guney;
            else if (gb_radioButton.Checked) Psi = Ch14.guneybati;
            else if (b_radioButton.Checked) Psi = Ch14.bati;
            else if (kb_radioButton.Checked) Psi = Ch14.kuzeybati;
            //
            // alpha/h0 değerini belirler.
            if (acik_radioButton.Checked) alphah0 = Ch14.acik_renk;
            else if (koyu_radioButton.Checked) alphah0 = Ch14.koyu_renk;
            //
```

```
            t_ic = Convert.ToDouble(this.dataDataSet.Giris.Rows[0][2]); //
 İç ortam sıcaklığı
            rts_id = Convert.ToInt16(this.dataDataSet.Giris.Rows[0][6]); /
/ RTS no
            duvar_id = Convert.ToInt16(duvar_alt_comboBox.Text); // Duvar
CTS no
            duvar_en = Convert.ToDouble(duvar_en_textBox.Text); // Duvar e
ni
            duvar_boy = Convert.ToDouble(duvar_boy_textBox.Text); // Duvar
 boyu
            Uduvar = Convert.ToDouble(this.dataDataSet.Duvar_Alt.Rows[duva
r_id - 1][2]); // Duvar ısı transfer katsayısı
            // Pencere adedi girilmezse pencere alanını sıfır olarak belir
ler.
            if (pencere_adet_textBox.Text == "") { pencere_adet = 0; pence
re_en = 0; pencere_boy = 0; }
            else
            {
                pencere_adet = Convert.ToInt16(pencere_adet_textBox.Text);
 // Pencere adedi
                pencere_en = Convert.ToDouble(pencere_en_textBox.Text); //
 Pencere eni
                pencere_boy = Convert.ToDouble(pencere_boy_textBox.Text);
// Pencere boyu
                Upencere = Convert.ToDouble(pencere_u_label.Text); // Penc
ere ısı transfer katsayısı

            }
            //
            // Kapı adedi girilmezse kapı alanını sıfır olarak belirler.
            if (kapi_adet_textBox.Text == "") { kapi_adet = 0; kapi_en = 0
; kapi_boy = 0; }
            else
            {
                kapi_adet = Convert.ToInt16(kapi_adet_textBox.Text); // Ka
pı adedi
                kapi_en = Convert.ToDouble(kapi_en_textBox.Text); // Kapı
eni
                kapi_boy = Convert.ToDouble(kapi_boy_textBox.Text); // Kap
ı boyu
                Ukapi = Convert.ToDouble(kapi_u_label.Text); // Kapı ısı t
ransfer katsayısı
            }
            //
        }
        void alan_hesapla() // Net yapı alanlarını hesaplar.
        {
            pencere_alan = pencere_adet * pencere_en * pencere_boy; // Pen
cere alanı
            kapi_alan = kapi_adet * kapi_en * kapi_boy; // Kapı alanı
            duvar_alan = duvar_en * duvar_boy -
 (pencere_alan + kapi_alan); // Net duvar alanı
            // Hesaplanan alanları ekrana yazdırır.
            duvar_alan_label.Text = string.Format("{0:0.##} m²", duvar_ala
n);
            pencere_alan_label.Text = string.Format("{0:0.##} m²", pencere
_alan);
            kapi_alan_label.Text = string.Format("{0:0.##} m²", kapi_alan)
;
```

```
            //
        }
        void cephe_radyasyon_hesapla() // Cepheye gelen saatlik güneş ışın
ımını hesaplar.
        {
            // Veritabanından verileri çeker.
            this.radyasyonTableAdapter.Fill(this.dataDataSet.Radyasyon);
            this.cephe1TableAdapter.Fill(this.dataDataSet.Cephe1);
            //
            for (int i = 0; i < 24; i++)
            {
                double Beta = Convert.ToDouble(this.dataDataSet.Radyasyon.
Rows[i][6]); // Güneş irtifa açısı
                double Phi = Convert.ToDouble(this.dataDataSet.Radyasyon.R
ows[i][7]); // Güneş azimut açısı
                double Eb = Convert.ToDouble(this.dataDataSet.Radyasyon.Ro
ws[i][9]); // Direkt güneş ışınımı
                double Ed = Convert.ToDouble(this.dataDataSet.Radyasyon.Ro
ws[i][10]); // Yayılan güneş ışınımı
                double Gamma = Ch14.getGamma(Phi, Psi); // Yüzey-
güneş azimut açısı
                double Theta = Ch14.getThetaVer(Beta, Gamma); // Geliş açı
sı
                double Etb = Ch14.getEtb(Eb, Theta); // Yüzeye gelen direk
t güneş ışınımı
                double Etd = Ch14.getEtd(Ed, Theta, 90); // Yüzeye yayılan
 güneş ışınımı
                double Etr = Ch14.getEtrVer(Eb, Beta, Ed); // Yerden yüzey
e yansıyan güneş ışınımı
                double Et = Ch14.GetEt(Etb, Etd, Etr); // Toplam güneş ışı
nımı
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Cephe1.Rows[i][2] = Gamma;
                this.dataDataSet.Cephe1.Rows[i][3] = Theta;
                this.dataDataSet.Cephe1.Rows[i][4] = Etb;
                this.dataDataSet.Cephe1.Rows[i][5] = Etd;
                this.dataDataSet.Cephe1.Rows[i][6] = Etr;
                this.dataDataSet.Cephe1.Rows[i][7] = Et;
                //
            }
            this.cephe1TableAdapter.Update(this.dataDataSet.Cephe1); // Ve
ritabanını günceller.
        }
        void duvar_Q_hesapla() // Duvardan saatlik toplam soğutma yükünü h
esaplar.
        {
            // Veritabanından verileri çeker.
            this.duvar1TableAdapter.Fill(this.dataDataSet.Duvar1);
            this.qtTableAdapter.Fill(this.dataDataSet.Qt);
            //
            for (int i = 0; i < 24; i++) // Saatlik eşdeğer sıcaklık ve ıs
ı girişini hesaplar.
            {
                double t_dis = Convert.ToDouble(this.dataDataSet.Radyasyon
.Rows[i][3]); // Dış ortam sıcaklığı
                double Et = Convert.ToDouble(this.dataDataSet.Cephe1.Rows[
i][7]); // Toplam güneş ışınımı
                double te = Ch18.get_teVer(t_dis, Et, alphah0); // Eşdeğer
 sıcaklık
```

```
                double qi = Ch18.get_qi(Uduvar, duvar_alan, te, t_ic); //
Isı girişi
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Duvar1.Rows[i][2] = t_dis;
                this.dataDataSet.Duvar1.Rows[i][3] = te;
                this.dataDataSet.Duvar1.Rows[i][4] = t_ic;
                this.dataDataSet.Duvar1.Rows[i][5] = qi;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için iletim zaman ser
ilerini uygular.
            {
                double qcond = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double c = Convert.ToDouble(this.dataDataSet.Duvar_CTS
.Rows[j][duvar_id]) / 100; // İletim zaman faktörü
                    double qi = Convert.ToDouble(this.dataDataSet.Duvar1.R
ows[saat][5]); // Isı girişi
                    qcond += c * qi; // İletim ısı kazancı
                    saat--;
                }
                double qconv = 0.54 * qcond; // Taşınımsal ısı kazancı
                double qrad = 0.46 * qcond; // Işınımsal ısı kazancı
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Duvar1.Rows[i][6] = Convert.ToDouble(this
.dataDataSet.Duvar_CTS.Rows[i][duvar_id]);
                this.dataDataSet.Duvar1.Rows[i][7] = qcond;
                this.dataDataSet.Duvar1.Rows[i][8] = qconv;
                this.dataDataSet.Duvar1.Rows[i][9] = qrad;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r = Convert.ToDouble(this.dataDataSet.RTS_Nonso
lar.Rows[j][rts_id]) / 100; // Işınım zaman faktörü
                    double qrad = Convert.ToDouble(this.dataDataSet.Duvar1
.Rows[saat][9]); // Işınımsal ısı kazancı
                    Qr += r * qrad; // Işınımsal soğutma yükü
                    saat--;
                }
                double qconv = Convert.ToDouble(this.dataDataSet.Duvar1.Ro
ws[i][8]); // Taşınımsal soğutma yükü
                double Qt = qconv + Qr; // Toplam saatlik soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Duvar1.Rows[i][10] = Convert.ToDouble(thi
s.dataDataSet.RTS_Nonsolar.Rows[i][rts_id]);
                this.dataDataSet.Duvar1.Rows[i][11] = Qr;
                this.dataDataSet.Duvar1.Rows[i][12] = Qt;
                this.dataDataSet.Qt.Rows[i][3] = Qt;
                //
            }
            // Veritabanını günceller.
```

```
            this.duvar1TableAdapter.Update(this.dataDataSet.Duvar1);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        void pencere_Q_hesapla() // Pencereden saatlik toplam soğutma yükü
nü hesaplar.
        {
            this.pencere1TableAdapter.Fill(this.dataDataSet.Pencere1); //
Veritabanından verileri çeker.
            int cam_id = Convert.ToInt16(pencere_ana_comboBox.SelectedInde
x);
            double shgc0 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.R
ows[cam_id][2]);
            double shgc40 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][3]);
            double shgc50 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][4]);
            double shgc60 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][5]);
            double shgc70 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][6]);
            double shgc80 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][7]);
            double shgc90 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][8]);
            double shgc_d = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][9]);
            for (int i = 0; i < 24; i++) // Saatlik ısı kazançlarını hesap
lar.
            {
                double shgc; // Güneş ısı kazanç katsayısı
                double Theta = Convert.ToDouble(this.dataDataSet.Cephe1.Ro
ws[i][3]); // Geliş açısı
                double t_dis = Convert.ToDouble(this.dataDataSet.Radyasyon
.Rows[i][3]); // Dış ortam sıcaklığı
                double Etb = Convert.ToDouble(this.dataDataSet.Cephe1.Rows
[i][4]); // Yüzeye gelen direkt güneş ışınımı
                double Etd = Convert.ToDouble(this.dataDataSet.Cephe1.Rows
[i][5]); // Yüzeye yayılan güneş ışınımı
                double Etr = Convert.ToDouble(this.dataDataSet.Cephe1.Rows
[i][6]); // Yerden yüzeye yansıyan güneş ışınımı
                // interpolasyon ile SHGC değerini belirler.
                if (Theta <= 0 && Theta > 40) shgc = shgc0 + (Theta -
 0) * (shgc40 - shgc0) / (40 - 0);
                else if (Theta >= 40 && Theta < 50) shgc = shgc40 + (Theta
 - 40) * (shgc50 - shgc40) / (50 - 40);
                else if (Theta >= 50 && Theta < 60) shgc = shgc50 + (Theta
 - 50) * (shgc60 - shgc50) / (60 - 50);
                else if (Theta >= 60 && Theta < 70) shgc = shgc60 + (Theta
 - 60) * (shgc70 - shgc60) / (70 - 60);
                else if (Theta >= 70 && Theta < 80) shgc = shgc70 + (Theta
 - 70) * (shgc80 - shgc70) / (80 - 70);
                else if (Theta >= 80 && Theta < 90) shgc = shgc80 + (Theta
 - 80) * (shgc90 - shgc80) / (90 - 80);
                else shgc = 0;
                //
                double qbeam = Ch18.get_qbeam(pencere_alan, Etb, shgc); //
 Direkt güneş ısı kazancı
```

```
                double qdiffuse = Ch18.get_qdiffuse(pencere_alan, Etd, Etr
, shgc_d); // Yayılan güneş ısı kazancı
                double qcond = Ch18.get_qi(Upencere, pencere_alan, t_dis,
t_ic); // İletim ısı kazancı
                double qconv, qrad;
                if (shgc_d <= 0.5)
                {
                    qconv = 0.54 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.46 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                else
                {
                    qconv = 0.67 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.33 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Pencere1.Rows[i][2] = t_dis;
                this.dataDataSet.Pencere1.Rows[i][3] = t_ic;
                this.dataDataSet.Pencere1.Rows[i][4] = Etb;
                this.dataDataSet.Pencere1.Rows[i][5] = shgc;
                this.dataDataSet.Pencere1.Rows[i][6] = qbeam;
                this.dataDataSet.Pencere1.Rows[i][9] = Etd;
                this.dataDataSet.Pencere1.Rows[i][10] = Etr;
                this.dataDataSet.Pencere1.Rows[i][11] = shgc_d;
                this.dataDataSet.Pencere1.Rows[i][12] = qdiffuse;
                this.dataDataSet.Pencere1.Rows[i][13] = qcond;
                this.dataDataSet.Pencere1.Rows[i][14] = qconv;
                this.dataDataSet.Pencere1.Rows[i][15] = qrad;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qb = 0; double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r_solar = Convert.ToDouble(this.dataDataSet.RTS
_Solar.Rows[j][rts_id]) / 100; // Güneşe bağlı ışınım zaman faktörü
                    double r_nonsolar = Convert.ToDouble(this.dataDataSet.
RTS_Nonsolar.Rows[j][rts_id]) / 100; // Güneşe bağlı olmayan ışınım zaman
faktörü
                    double qbeam = Convert.ToDouble(this.dataDataSet.Pence
re1.Rows[saat][6]); // Direkt güneş ısı kazancı
                    double qrad = Convert.ToDouble(this.dataDataSet.Pencer
e1.Rows[saat][15]); // Yayılan güneş+iletim ışınımsal ısı kazancı
                    Qb += r_solar * qbeam; // Direkt güneş soğutma yükü
                    Qr += r_nonsolar * qrad; // Yayılan güneş+iletim ışını
msal soğutma yükü
                    saat--;
                }
                double qconv = Convert.ToDouble(this.dataDataSet.Pencere1.
Rows[i][14]); // Yayılan güneş+iletim taşınımsal soğutma yükü
                double Qdc = qconv + Qr; // Yayılan güneş+iletim soğutma y
ükü
```

```
                double Qt = Qb + Qdc; // Toplam soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Pencere1.Rows[i][7] = Convert.ToDouble(th
is.dataDataSet.RTS_Solar[i][rts_id]);
                this.dataDataSet.Pencere1.Rows[i][16] = Convert.ToDouble(t
his.dataDataSet.RTS_Nonsolar[i][rts_id]);
                this.dataDataSet.Pencere1.Rows[i][8] = Qb;
                this.dataDataSet.Pencere1.Rows[i][17] = Qr;
                this.dataDataSet.Pencere1.Rows[i][18] = Qdc;
                this.dataDataSet.Pencere1.Rows[i][19] = Qt;
                this.dataDataSet.Qt.Rows[i][3] = Convert.ToDouble(this.dat
aDataSet.Qt.Rows[i][3]) + Qt;
                //
            }
            // Veritabanını günceller.
            this.pencere1TableAdapter.Update(this.dataDataSet.Pencere1);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        void kapi_Q_hesapla() //Kapıdan saatlik toplam soğutma yükünü esap
lar.
        {
            this.kapi1TableAdapter.Fill(this.dataDataSet.Kapi1); // Verita
banından verileri çeker.
            int cam_id = Convert.ToInt16(kapi_ana_comboBox.SelectedIndex);
            double shgc0 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.R
ows[cam_id][2]);
            double shgc40 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][3]);
            double shgc50 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][4]);
            double shgc60 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][5]);
            double shgc70 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][6]);
            double shgc80 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][7]);
            double shgc90 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][8]);
            double shgc_d = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][9]);
            for (int i = 0; i < 24; i++) // Saatlik ısı kazançlarını hesap
lar.
            {
                double shgc; // Güneş ısı kazanç katsayısı
                double Theta = Convert.ToDouble(this.dataDataSet.Cephe1.Ro
ws[i][3]); // Geliş açısı
                double t_dis = Convert.ToDouble(this.dataDataSet.Radyasyon
.Rows[i][3]); // Dış ortam sıcaklığı
                double Etb = Convert.ToDouble(this.dataDataSet.Cephe1.Rows
[i][4]); // Yüzeye gelen direkt güneş ışınımı
                double Etd = Convert.ToDouble(this.dataDataSet.Cephe1.Rows
[i][5]); // Yüzeye yayılan güneş ışınımı
                double Etr = Convert.ToDouble(this.dataDataSet.Cephe1.Rows
[i][6]); // Yerden yüzeye yansıyan güneş ışınımı
                // interpolasyon ile SHGC değerini belirler.
                if (Theta <= 0 && Theta > 40) shgc = shgc0 + (Theta -
 0) * (shgc40 - shgc0) / (40 - 0);
```

```
                else if (Theta >= 40 && Theta < 50) shgc = shgc40 + (Theta
 - 40) * (shgc50 - shgc40) / (50 - 40);
                else if (Theta >= 50 && Theta < 60) shgc = shgc50 + (Theta
 - 50) * (shgc60 - shgc50) / (60 - 50);
                else if (Theta >= 60 && Theta < 70) shgc = shgc60 + (Theta
 - 60) * (shgc70 - shgc60) / (70 - 60);
                else if (Theta >= 70 && Theta < 80) shgc = shgc70 + (Theta
 - 70) * (shgc80 - shgc70) / (80 - 70);
                else if (Theta >= 80 && Theta < 90) shgc = shgc80 + (Theta
 - 80) * (shgc90 - shgc80) / (90 - 80);
                else shgc = 0;
                //
                double qbeam = Ch18.get_qbeam(kapi_alan, Etb, shgc); // Di
rekt güneş ısı kazancı
                double qdiffuse = Ch18.get_qdiffuse(kapi_alan, Etd, Etr, s
hgc_d); // Yayılan güneş ısı kazancı
                double qcond = Ch18.get_qi(Ukapi, kapi_alan, t_dis, t_ic);
 // İletim ısı kazancı
                double qconv, qrad;
                if (shgc_d <= 0.5)
                {
                    qconv = 0.54 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.46 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                else
                {
                    qconv = 0.67 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.33 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Kapi1.Rows[i][2] = t_dis;
                this.dataDataSet.Kapi1.Rows[i][3] = t_ic;
                this.dataDataSet.Kapi1.Rows[i][4] = Etb;
                this.dataDataSet.Kapi1.Rows[i][5] = shgc;
                this.dataDataSet.Kapi1.Rows[i][6] = qbeam;
                this.dataDataSet.Kapi1.Rows[i][9] = Etd;
                this.dataDataSet.Kapi1.Rows[i][10] = Etr;
                this.dataDataSet.Kapi1.Rows[i][11] = shgc_d;
                this.dataDataSet.Kapi1.Rows[i][12] = qdiffuse;
                this.dataDataSet.Kapi1.Rows[i][13] = qcond;
                this.dataDataSet.Kapi1.Rows[i][14] = qconv;
                this.dataDataSet.Kapi1.Rows[i][15] = qrad;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qb = 0; double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r_solar = Convert.ToDouble(this.dataDataSet.RTS
_Solar.Rows[j][rts_id]) / 100; // Güneşe bağlı ışınım zaman faktörü
```

```
                    double r_nonsolar = Convert.ToDouble(this.dataDataSet.
RTS_Nonsolar.Rows[j][rts_id]) / 100; // Güneşe bağlı olmayan ışınım zaman
faktörü
                    double qbeam = Convert.ToDouble(this.dataDataSet.Kapi1
.Rows[saat][6]); // Direkt güneş ısı kazancı
                    double qrad = Convert.ToDouble(this.dataDataSet.Kapi1.
Rows[saat][15]); // Yayılan güneş+iletim ışınımsal ısı kazancı
                    Qb += r_solar * qbeam; // Direkt güneş soğutma yükü
                    Qr += r_nonsolar * qrad; // Yayılan güneş+iletim ışını
msal soğutma yükü
                    saat--;
                }
                double qconv = Convert.ToDouble(this.dataDataSet.Kapi1.Row
s[i][14]); // Yayılan güneş+iletim taşınımsal soğutma yükü
                double Qdc = qconv + Qr; // Yayılan güneş+iletim soğutma y
ükü
                double Qt = Qb + Qdc; // Toplam soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Kapi1.Rows[i][7] = Convert.ToDouble(this.
dataDataSet.RTS_Solar[i][rts_id]);
                this.dataDataSet.Kapi1.Rows[i][16] = Convert.ToDouble(this
.dataDataSet.RTS_Nonsolar[i][rts_id]);
                this.dataDataSet.Kapi1.Rows[i][8] = Qb;
                this.dataDataSet.Kapi1.Rows[i][17] = Qr;
                this.dataDataSet.Kapi1.Rows[i][18] = Qdc;
                this.dataDataSet.Kapi1.Rows[i][19] = Qt;
                this.dataDataSet.Qt.Rows[i][3] = Convert.ToDouble(this.dat
aDataSet.Qt.Rows[i][3]) + Qt;
                //
            }
            // Veritabanını günceller.
            this.kapi1TableAdapter.Update(this.dataDataSet.Kapi1);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        private void Cephe1_Load(object sender, EventArgs e)
        {
            // Veritabanından verileri çeker.
            this.duvar_AnaTableAdapter.Fill(this.dataDataSet.Duvar_Ana);
            this.duvar_AltTableAdapter.Fill(this.dataDataSet.Duvar_Alt);
            this.duvar_YapiTableAdapter.Fill(this.dataDataSet.Duvar_Yapi);
            this.pencere_AnaTableAdapter.Fill(this.dataDataSet.Pencere_Ana
);
            this.pencere_AltTableAdapter.Fill(this.dataDataSet.Pencere_Alt
);
            this.pencere_CerceveTableAdapter.Fill(this.dataDataSet.Pencere
_Cerceve);
            this.duvar_CTSTableAdapter.Fill(this.dataDataSet.Duvar_CTS);
            this.rTS_NonsolarTableAdapter.Fill(this.dataDataSet.RTS_Nonsol
ar);
            this.rTS_SolarTableAdapter.Fill(this.dataDataSet.RTS_Solar);
            //
        }
        // En ve boy girişinin yapıldığı metin kutularında sadece sayı, vi
rgül, del ve backspace tuşlarını aktif eder.
        private void duvar_en_textBox_KeyPress(object sender, KeyPressEven
tArgs e)
        {
```

```
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)44 && e.KeyChar != (char)127;
        }
        // Adet girişinin yapıldığı metin kutularında sadece sayı, del ve
backspace tuşlarını aktif eder.
        private void pencere_adet_textBox_KeyPress(object sender, KeyPress
EventArgs e)
        {
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)127;
        }
        //
        private void hesapla_button_Click(object sender, EventArgs e)
        {
            veri_cek();
            alan_hesapla();
            cephe_radyasyon_hesapla();
            duvar_Q_hesapla();
            if (pencere_alan != 0) pencere_Q_hesapla(); // Pencere alanı s
ıfırdan farklı ise soğutma yükü hesaplar.
            if (kapi_alan != 0) kapi_Q_hesapla(); // Kapı alanı sıfırdan f
arklı ise soğutma yükü hesaplar.
            this.AutoScroll = true; // Sayfa kaydırmayı aktif eder.
            cephe_radyasyon_groupBox.Visible = true; // Hesaplanan değerle
ri ekranda gösterir.
            kontrol = true; // Hesapla butonu tıklama kontrolü
        }
    }
}
```

EK C.7 Cephe2 Ekranı Kaynak Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Cephe2 Ekranı Kaynak Kodları
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BuharlasmaliSogutucuHesapProgrami
{
    public partial class Cephe2 : UserControl
    {
        public Cephe2()
        {
            InitializeComponent();
        }
        // Fonksiyon dosyasını bağlar.
        fonksiyon.Bolum14 Ch14 = new fonksiyon.Bolum14();
        fonksiyon.Bolum18 Ch18 = new fonksiyon.Bolum18();
        //
        // Değişken tanımları
        int rts_id, duvar_id, pencere_adet, kapi_adet;
        double t_ic, Psi, alphah0, duvar_en, duvar_boy, duvar_alan, Uduvar
, pencere_en, pencere_boy, pencere_alan, Upencere, kapi_en, kapi_boy, kapi
_alan, Ukapi;
        public bool kontrol = false; // Hesapla butonu tıklama kontrol değ
işkeni
        //
        void veri_cek() // Metin kutuları, radyo butonları ve veri tabanın
dan verileri çeker.
        {
            this.girisTableAdapter.Fill(this.dataDataSet.Giris);
            // Yüzey azimut açısını belirler.
            if (k_radioButton.Checked) Psi = Ch14.kuzey;
            else if (kd_radioButton.Checked) Psi = Ch14.kuzeydogu;
            else if (d_radioButton.Checked) Psi = Ch14.dogu;
            else if (gd_radioButton.Checked) Psi = Ch14.guneydogu;
            else if (g_radioButton.Checked) Psi = Ch14.guney;
            else if (gb_radioButton.Checked) Psi = Ch14.guneybati;
            else if (b_radioButton.Checked) Psi = Ch14.bati;
            else if (kb_radioButton.Checked) Psi = Ch14.kuzeybati;
            //
            // alpha/h0 değerini belirler.
            if (acik_radioButton.Checked) alphah0 = Ch14.acik_renk;
            else if (koyu_radioButton.Checked) alphah0 = Ch14.koyu_renk;
            //
```

```
            t_ic = Convert.ToDouble(this.dataDataSet.Giris.Rows[0][2]); //
 İç ortam sıcaklığı
            rts_id = Convert.ToInt16(this.dataDataSet.Giris.Rows[0][6]); /
/ RTS no
            duvar_id = Convert.ToInt16(duvar_alt_comboBox.Text); // Duvar
CTS no
            duvar_en = Convert.ToDouble(duvar_en_textBox.Text); // Duvar e
ni
            duvar_boy = Convert.ToDouble(duvar_boy_textBox.Text); // Duvar
 boyu
            Uduvar = Convert.ToDouble(this.dataDataSet.Duvar_Alt.Rows[duva
r_id - 1][2]); // Duvar ısı transfer katsayısı
            // Pencere adedi girilmezse pencere alanını sıfır olarak belir
ler.
            if (pencere_adet_textBox.Text == "") { pencere_adet = 0; pence
re_en = 0; pencere_boy = 0; }
            else
            {
                pencere_adet = Convert.ToInt16(pencere_adet_textBox.Text);
 // Pencere adedi
                pencere_en = Convert.ToDouble(pencere_en_textBox.Text); //
 Pencere eni
                pencere_boy = Convert.ToDouble(pencere_boy_textBox.Text);
// Pencere boyu
                Upencere = Convert.ToDouble(pencere_u_label.Text); // Penc
ere ısı transfer katsayısı

            }
            //
            // Kapı adedi girilmezse kapı alanını sıfır olarak belirler.
            if (kapi_adet_textBox.Text == "") { kapi_adet = 0; kapi_en = 0
; kapi_boy = 0; }
            else
            {
                kapi_adet = Convert.ToInt16(kapi_adet_textBox.Text); // Ka
pı adedi
                kapi_en = Convert.ToDouble(kapi_en_textBox.Text); // Kapı
eni
                kapi_boy = Convert.ToDouble(kapi_boy_textBox.Text); // Kap
ı boyu
                Ukapi = Convert.ToDouble(kapi_u_label.Text); // Kapı ısı t
ransfer katsayısı
            }
            //
        }
        void alan_hesapla() // Net yapı alanlarını hesaplar.
        {
            pencere_alan = pencere_adet * pencere_en * pencere_boy; // Pen
cere alanı
            kapi_alan = kapi_adet * kapi_en * kapi_boy; // Kapı alanı
            duvar_alan = duvar_en * duvar_boy -
 (pencere_alan + kapi_alan); // Net duvar alanı
            // Hesaplanan alanları ekrana yazdırır.
            duvar_alan_label.Text = string.Format("{0:0.##} m²", duvar_ala
n);
            pencere_alan_label.Text = string.Format("{0:0.##} m²", pencere
_alan);
            kapi_alan_label.Text = string.Format("{0:0.##} m²", kapi_alan)
;
```

```
            //
        }
        void cephe_radyasyon_hesapla() // Cepheye gelen saatlik güneş ışın
ımını hesaplar.
        {
            // Veritabanından verileri çeker.
            this.radyasyonTableAdapter.Fill(this.dataDataSet.Radyasyon);
            this.cephe2TableAdapter.Fill(this.dataDataSet.Cephe2);
            //
            for (int i = 0; i < 24; i++)
            {
                double Beta = Convert.ToDouble(this.dataDataSet.Radyasyon.
Rows[i][6]); // Güneş irtifa açısı
                double Phi = Convert.ToDouble(this.dataDataSet.Radyasyon.R
ows[i][7]); // Güneş azimut açısı
                double Eb = Convert.ToDouble(this.dataDataSet.Radyasyon.Ro
ws[i][9]); // Direkt güneş ışınımı
                double Ed = Convert.ToDouble(this.dataDataSet.Radyasyon.Ro
ws[i][10]); // Yayılan güneş ışınımı
                double Gamma = Ch14.getGamma(Phi, Psi); // Yüzey-
güneş azimut açısı
                double Theta = Ch14.getThetaVer(Beta, Gamma); // Geliş açı
sı
                double Etb = Ch14.getEtb(Eb, Theta); // Yüzeye gelen direk
t güneş ışınımı
                double Etd = Ch14.getEtd(Ed, Theta, 90); // Yüzeye yayılan
 güneş ışınımı
                double Etr = Ch14.getEtrVer(Eb, Beta, Ed); // Yerden yüzey
e yansıyan güneş ışınımı
                double Et = Ch14.GetEt(Etb, Etd, Etr); // Toplam güneş ışı
nımı
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Cephe2.Rows[i][2] = Gamma;
                this.dataDataSet.Cephe2.Rows[i][3] = Theta;
                this.dataDataSet.Cephe2.Rows[i][4] = Etb;
                this.dataDataSet.Cephe2.Rows[i][5] = Etd;
                this.dataDataSet.Cephe2.Rows[i][6] = Etr;
                this.dataDataSet.Cephe2.Rows[i][7] = Et;
                //
            }
            this.cephe2TableAdapter.Update(this.dataDataSet.Cephe2); // Ve
ritabanını günceller.
        }
        void duvar_Q_hesapla() // Duvardan saatlik toplam soğutma yükünü h
esaplar.
        {
            // Veritabanından verileri çeker.
            this.duvar2TableAdapter.Fill(this.dataDataSet.Duvar2);
            this.qtTableAdapter.Fill(this.dataDataSet.Qt);
            //
            for (int i = 0; i < 24; i++) // Saatlik eşdeğer sıcaklık ve ıs
ı girişini hesaplar.
            {
                double t_dis = Convert.ToDouble(this.dataDataSet.Radyasyon
.Rows[i][3]); // Dış ortam sıcaklığı
                double Et = Convert.ToDouble(this.dataDataSet.Cephe2.Rows[
i][7]); // Toplam güneş ışınımı
                double te = Ch18.get_teVer(t_dis, Et, alphah0); // Eşdeğer
 sıcaklık
```

```
                double qi = Ch18.get_qi(Uduvar, duvar_alan, te, t_ic); //
Isı girişi
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Duvar2.Rows[i][2] = t_dis;
                this.dataDataSet.Duvar2.Rows[i][3] = te;
                this.dataDataSet.Duvar2.Rows[i][4] = t_ic;
                this.dataDataSet.Duvar2.Rows[i][5] = qi;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için iletim zaman ser
ilerini uygular.
            {
                double qcond = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double c = Convert.ToDouble(this.dataDataSet.Duvar_CTS
.Rows[j][duvar_id]) / 100; // İletim zaman faktörü
                    double qi = Convert.ToDouble(this.dataDataSet.Duvar2.R
ows[saat][5]); // Isı girişi
                    qcond += c * qi; // İletim ısı kazancı
                    saat--;
                }
                double qconv = 0.54 * qcond; // Taşınımsal ısı kazancı
                double qrad = 0.46 * qcond; // Işınımsal ısı kazancı
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Duvar2.Rows[i][6] = Convert.ToDouble(this
.dataDataSet.Duvar_CTS.Rows[i][duvar_id]);
                this.dataDataSet.Duvar2.Rows[i][7] = qcond;
                this.dataDataSet.Duvar2.Rows[i][8] = qconv;
                this.dataDataSet.Duvar2.Rows[i][9] = qrad;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r = Convert.ToDouble(this.dataDataSet.RTS_Nonso
lar.Rows[j][rts_id]) / 100; // Işınım zaman faktörü
                    double qrad = Convert.ToDouble(this.dataDataSet.Duvar2
.Rows[saat][9]); // Işınımsal ısı kazancı
                    Qr += r * qrad; // Işınımsal soğutma yükü
                    saat--;
                }
                double qconv = Convert.ToDouble(this.dataDataSet.Duvar2.Ro
ws[i][8]); // Taşınımsal soğutma yükü
                double Qt = qconv + Qr; // Toplam saatlik soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Duvar2.Rows[i][10] = Convert.ToDouble(thi
s.dataDataSet.RTS_Nonsolar.Rows[i][rts_id]);
                this.dataDataSet.Duvar2.Rows[i][11] = Qr;
                this.dataDataSet.Duvar2.Rows[i][12] = Qt;
                this.dataDataSet.Qt.Rows[i][4] = Qt;
                //
            }
            // Veritabanını günceller.
```

```
            this.duvar2TableAdapter.Update(this.dataDataSet.Duvar2);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        void pencere_Q_hesapla() // Pencereden saatlik toplam soğutma yükü
nü hesaplar.
        {
            this.pencere2TableAdapter.Fill(this.dataDataSet.Pencere2); //
Veritabanından verileri çeker.
            int cam_id = Convert.ToInt16(pencere_ana_comboBox.SelectedInde
x);
            double shgc0 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.R
ows[cam_id][2]);
            double shgc40 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][3]);
            double shgc50 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][4]);
            double shgc60 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][5]);
            double shgc70 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][6]);
            double shgc80 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][7]);
            double shgc90 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][8]);
            double shgc_d = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][9]);
            for (int i = 0; i < 24; i++) // Saatlik ısı kazançlarını hesap
lar.
            {
                double shgc; // Güneş ısı kazanç katsayısı
                double Theta = Convert.ToDouble(this.dataDataSet.Cephe2.Ro
ws[i][3]); // Geliş açısı
                double t_dis = Convert.ToDouble(this.dataDataSet.Radyasyon
.Rows[i][3]); // Dış ortam sıcaklığı
                double Etb = Convert.ToDouble(this.dataDataSet.Cephe2.Rows
[i][4]); // Yüzeye gelen direkt güneş ışınımı
                double Etd = Convert.ToDouble(this.dataDataSet.Cephe2.Rows
[i][5]); // Yüzeye yayılan güneş ışınımı
                double Etr = Convert.ToDouble(this.dataDataSet.Cephe2.Rows
[i][6]); // Yerden yüzeye yansıyan güneş ışınımı
                // interpolasyon ile SHGC değerini belirler.
                if (Theta <= 0 && Theta > 40) shgc = shgc0 + (Theta -
 0) * (shgc40 - shgc0) / (40 - 0);
                else if (Theta >= 40 && Theta < 50) shgc = shgc40 + (Theta
 - 40) * (shgc50 - shgc40) / (50 - 40);
                else if (Theta >= 50 && Theta < 60) shgc = shgc50 + (Theta
 - 50) * (shgc60 - shgc50) / (60 - 50);
                else if (Theta >= 60 && Theta < 70) shgc = shgc60 + (Theta
 - 60) * (shgc70 - shgc60) / (70 - 60);
                else if (Theta >= 70 && Theta < 80) shgc = shgc70 + (Theta
 - 70) * (shgc80 - shgc70) / (80 - 70);
                else if (Theta >= 80 && Theta < 90) shgc = shgc80 + (Theta
 - 80) * (shgc90 - shgc80) / (90 - 80);
                else shgc = 0;
                //
                double qbeam = Ch18.get_qbeam(pencere_alan, Etb, shgc); //
 Direkt güneş ısı kazancı
```

```
                double qdiffuse = Ch18.get_qdiffuse(pencere_alan, Etd, Etr
, shgc_d); // Yayılan güneş ısı kazancı
                double qcond = Ch18.get_qi(Upencere, pencere_alan, t_dis,
t_ic); // İletim ısı kazancı
                double qconv, qrad;
                if (shgc_d <= 0.5)
                {
                    qconv = 0.54 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.46 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                else
                {
                    qconv = 0.67 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.33 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Pencere2.Rows[i][2] = t_dis;
                this.dataDataSet.Pencere2.Rows[i][3] = t_ic;
                this.dataDataSet.Pencere2.Rows[i][4] = Etb;
                this.dataDataSet.Pencere2.Rows[i][5] = shgc;
                this.dataDataSet.Pencere2.Rows[i][6] = qbeam;
                this.dataDataSet.Pencere2.Rows[i][9] = Etd;
                this.dataDataSet.Pencere2.Rows[i][10] = Etr;
                this.dataDataSet.Pencere2.Rows[i][11] = shgc_d;
                this.dataDataSet.Pencere2.Rows[i][12] = qdiffuse;
                this.dataDataSet.Pencere2.Rows[i][13] = qcond;
                this.dataDataSet.Pencere2.Rows[i][14] = qconv;
                this.dataDataSet.Pencere2.Rows[i][15] = qrad;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qb = 0; double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r_solar = Convert.ToDouble(this.dataDataSet.RTS
_Solar.Rows[j][rts_id]) / 100; // Güneşe bağlı ışınım zaman faktörü
                    double r_nonsolar = Convert.ToDouble(this.dataDataSet.
RTS_Nonsolar.Rows[j][rts_id]) / 100; // Güneşe bağlı olmayan ışınım zaman
faktörü
                    double qbeam = Convert.ToDouble(this.dataDataSet.Pence
re2.Rows[saat][6]); // Direkt güneş ısı kazancı
                    double qrad = Convert.ToDouble(this.dataDataSet.Pencer
e2.Rows[saat][15]); // Yayılan güneş+iletim ışınımsal ısı kazancı
                    Qb += r_solar * qbeam; // Direkt güneş soğutma yükü
                    Qr += r_nonsolar * qrad; // Yayılan güneş+iletim ışını
msal soğutma yükü
                    saat--;
                }
                double qconv = Convert.ToDouble(this.dataDataSet.Pencere2.
Rows[i][14]); // Yayılan güneş+iletim taşınımsal soğutma yükü
                double Qdc = qconv + Qr; // Yayılan güneş+iletim soğutma y
ükü
```

```
                double Qt = Qb + Qdc; // Toplam soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Pencere2.Rows[i][7] = Convert.ToDouble(th
is.dataDataSet.RTS_Solar[i][rts_id]);
                this.dataDataSet.Pencere2.Rows[i][16] = Convert.ToDouble(t
his.dataDataSet.RTS_Nonsolar[i][rts_id]);
                this.dataDataSet.Pencere2.Rows[i][8] = Qb;
                this.dataDataSet.Pencere2.Rows[i][17] = Qr;
                this.dataDataSet.Pencere2.Rows[i][18] = Qdc;
                this.dataDataSet.Pencere2.Rows[i][19] = Qt;
                this.dataDataSet.Qt.Rows[i][4] = Convert.ToDouble(this.dat
aDataSet.Qt.Rows[i][4]) + Qt;
                //
            }
            // Veritabanını günceller.
            this.pencere2TableAdapter.Update(this.dataDataSet.Pencere2);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        void kapi_Q_hesapla() //Kapıdan saatlik toplam soğutma yükünü esap
lar.
        {
            this.kapi2TableAdapter.Fill(this.dataDataSet.Kapi2); // Verita
banından verileri çeker.
            int cam_id = Convert.ToInt16(kapi_ana_comboBox.SelectedIndex);
            double shgc0 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.R
ows[cam_id][2]);
            double shgc40 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][3]);
            double shgc50 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][4]);
            double shgc60 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][5]);
            double shgc70 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][6]);
            double shgc80 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][7]);
            double shgc90 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][8]);
            double shgc_d = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][9]);
            for (int i = 0; i < 24; i++) // Saatlik ısı kazançlarını hesap
lar.
            {
                double shgc; // Güneş ısı kazanç katsayısı
                double Theta = Convert.ToDouble(this.dataDataSet.Cephe2.Ro
ws[i][3]); // Geliş açısı
                double t_dis = Convert.ToDouble(this.dataDataSet.Radyasyon
.Rows[i][3]); // Dış ortam sıcaklığı
                double Etb = Convert.ToDouble(this.dataDataSet.Cephe2.Rows
[i][4]); // Yüzeye gelen direkt güneş ışınımı
                double Etd = Convert.ToDouble(this.dataDataSet.Cephe2.Rows
[i][5]); // Yüzeye yayılan güneş ışınımı
                double Etr = Convert.ToDouble(this.dataDataSet.Cephe2.Rows
[i][6]); // Yerden yüzeye yansıyan güneş ışınımı
                // interpolasyon ile SHGC değerini belirler.
                if (Theta <= 0 && Theta > 40) shgc = shgc0 + (Theta -
 0) * (shgc40 - shgc0) / (40 - 0);
```

```
                else if (Theta >= 40 && Theta < 50) shgc = shgc40 + (Theta
 - 40) * (shgc50 - shgc40) / (50 - 40);
                else if (Theta >= 50 && Theta < 60) shgc = shgc50 + (Theta
 - 50) * (shgc60 - shgc50) / (60 - 50);
                else if (Theta >= 60 && Theta < 70) shgc = shgc60 + (Theta
 - 60) * (shgc70 - shgc60) / (70 - 60);
                else if (Theta >= 70 && Theta < 80) shgc = shgc70 + (Theta
 - 70) * (shgc80 - shgc70) / (80 - 70);
                else if (Theta >= 80 && Theta < 90) shgc = shgc80 + (Theta
 - 80) * (shgc90 - shgc80) / (90 - 80);
                else shgc = 0;
                //
                double qbeam = Ch18.get_qbeam(kapi_alan, Etb, shgc); // Di
rekt güneş ısı kazancı
                double qdiffuse = Ch18.get_qdiffuse(kapi_alan, Etd, Etr, s
hgc_d); // Yayılan güneş ısı kazancı
                double qcond = Ch18.get_qi(Ukapi, kapi_alan, t_dis, t_ic);
 // İletim ısı kazancı
                double qconv, qrad;
                if (shgc_d <= 0.5)
                {
                    qconv = 0.54 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.46 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                else
                {
                    qconv = 0.67 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.33 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Kapi2.Rows[i][2] = t_dis;
                this.dataDataSet.Kapi2.Rows[i][3] = t_ic;
                this.dataDataSet.Kapi2.Rows[i][4] = Etb;
                this.dataDataSet.Kapi2.Rows[i][5] = shgc;
                this.dataDataSet.Kapi2.Rows[i][6] = qbeam;
                this.dataDataSet.Kapi2.Rows[i][9] = Etd;
                this.dataDataSet.Kapi2.Rows[i][10] = Etr;
                this.dataDataSet.Kapi2.Rows[i][11] = shgc_d;
                this.dataDataSet.Kapi2.Rows[i][12] = qdiffuse;
                this.dataDataSet.Kapi2.Rows[i][13] = qcond;
                this.dataDataSet.Kapi2.Rows[i][14] = qconv;
                this.dataDataSet.Kapi2.Rows[i][15] = qrad;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qb = 0; double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r_solar = Convert.ToDouble(this.dataDataSet.RTS
_Solar.Rows[j][rts_id]) / 100; // Güneşe bağlı ışınım zaman faktörü
```

```
                    double r_nonsolar = Convert.ToDouble(this.dataDataSet.
RTS_Nonsolar.Rows[j][rts_id]) / 100; // Güneşe bağlı olmayan ışınım zaman
faktörü
                    double qbeam = Convert.ToDouble(this.dataDataSet.Kapi2
.Rows[saat][6]); // Direkt güneş ısı kazancı
                    double qrad = Convert.ToDouble(this.dataDataSet.Kapi2.
Rows[saat][15]); // Yayılan güneş+iletim ışınımsal ısı kazancı
                    Qb += r_solar * qbeam; // Direkt güneş soğutma yükü
                    Qr += r_nonsolar * qrad; // Yayılan güneş+iletim ışını
msal soğutma yükü
                    saat--;
                }
                double qconv = Convert.ToDouble(this.dataDataSet.Kapi2.Row
s[i][14]); // Yayılan güneş+iletim taşınımsal soğutma yükü
                double Qdc = qconv + Qr; // Yayılan güneş+iletim soğutma y
ükü
                double Qt = Qb + Qdc; // Toplam soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Kapi2.Rows[i][7] = Convert.ToDouble(this.
dataDataSet.RTS_Solar[i][rts_id]);
                this.dataDataSet.Kapi2.Rows[i][16] = Convert.ToDouble(this
.dataDataSet.RTS_Nonsolar[i][rts_id]);
                this.dataDataSet.Kapi2.Rows[i][8] = Qb;
                this.dataDataSet.Kapi2.Rows[i][17] = Qr;
                this.dataDataSet.Kapi2.Rows[i][18] = Qdc;
                this.dataDataSet.Kapi2.Rows[i][19] = Qt;
                this.dataDataSet.Qt.Rows[i][4] = Convert.ToDouble(this.dat
aDataSet.Qt.Rows[i][4]) + Qt;
                //
            }
            // Veritabanını günceller.
            this.kapi2TableAdapter.Update(this.dataDataSet.Kapi2);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        private void Cephe2_Load(object sender, EventArgs e)
        {
            // Veritabanından verileri çeker.
            this.duvar_AnaTableAdapter.Fill(this.dataDataSet.Duvar_Ana);
            this.duvar_AltTableAdapter.Fill(this.dataDataSet.Duvar_Alt);
            this.duvar_YapiTableAdapter.Fill(this.dataDataSet.Duvar_Yapi);
            this.pencere_AnaTableAdapter.Fill(this.dataDataSet.Pencere_Ana
);
            this.pencere_AltTableAdapter.Fill(this.dataDataSet.Pencere_Alt
);
            this.pencere_CerceveTableAdapter.Fill(this.dataDataSet.Pencere
_Cerceve);
            this.duvar_CTSTableAdapter.Fill(this.dataDataSet.Duvar_CTS);
            this.rTS_NonsolarTableAdapter.Fill(this.dataDataSet.RTS_Nonsol
ar);
            this.rTS_SolarTableAdapter.Fill(this.dataDataSet.RTS_Solar);
            //
        }
        // En ve boy girişinin yapıldığı metin kutularında sadece sayı, vi
rgül, del ve backspace tuşlarını aktif eder.
        private void duvar_en_textBox_KeyPress(object sender, KeyPressEven
tArgs e)
        {
```

```
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)44 && e.KeyChar != (char)127;
        }
        // Adet girişinin yapıldığı metin kutularında sadece sayı, del ve
backspace tuşlarını aktif eder.
        private void pencere_adet_textBox_KeyPress(object sender, KeyPress
EventArgs e)
        {
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)127;
        }
        //
        private void hesapla_button_Click(object sender, EventArgs e)
        {
            veri_cek();
            alan_hesapla();
            cephe_radyasyon_hesapla();
            duvar_Q_hesapla();
            if (pencere_alan != 0) pencere_Q_hesapla(); // Pencere alanı s
ıfırdan farklı ise soğutma yükü hesaplar.
            if (kapi_alan != 0) kapi_Q_hesapla(); // Kapı alanı sıfırdan f
arklı ise soğutma yükü hesaplar.
            this.AutoScroll = true; // Sayfa kaydırmayı aktif eder.
            cephe_radyasyon_groupBox.Visible = true; // Hesaplanan değerle
ri ekranda gösterir.
            kontrol = true; // Hesapla butonu tıklama kontrolü
        }
    }
}
```

EK C.8 Cephe3 Ekranı Kaynak Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Cephe3 Ekranı Kaynak Kodları
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BuharlasmaliSogutucuHesapProgrami
{
    public partial class Cephe3 : UserControl
    {
        public Cephe3()
        {
            InitializeComponent();
        }
        // Fonksiyon dosyasını bağlar.
        fonksiyon.Bolum14 Ch14 = new fonksiyon.Bolum14();
        fonksiyon.Bolum18 Ch18 = new fonksiyon.Bolum18();
        //
        // Değişken tanımları
        int rts_id, duvar_id, pencere_adet, kapi_adet;
        double t_ic, Psi, alphah0, duvar_en, duvar_boy, duvar_alan, Uduvar
, pencere_en, pencere_boy, pencere_alan, Upencere, kapi_en, kapi_boy, kapi
_alan, Ukapi;
        public bool kontrol = false; // Hesapla butonu tıklama kontrol değ
işkeni
        //
        void veri_cek() // Metin kutuları, radyo butonları ve veri tabanın
dan verileri çeker.
        {
            this.girisTableAdapter.Fill(this.dataDataSet.Giris);
            // Yüzey azimut açısını belirler.
            if (k_radioButton.Checked) Psi = Ch14.kuzey;
            else if (kd_radioButton.Checked) Psi = Ch14.kuzeydogu;
            else if (d_radioButton.Checked) Psi = Ch14.dogu;
            else if (gd_radioButton.Checked) Psi = Ch14.guneydogu;
            else if (g_radioButton.Checked) Psi = Ch14.guney;
            else if (gb_radioButton.Checked) Psi = Ch14.guneybati;
            else if (b_radioButton.Checked) Psi = Ch14.bati;
            else if (kb_radioButton.Checked) Psi = Ch14.kuzeybati;
            //
            // alpha/h0 değerini belirler.
            if (acik_radioButton.Checked) alphah0 = Ch14.acik_renk;
            else if (koyu_radioButton.Checked) alphah0 = Ch14.koyu_renk;
            //
```

```
            t_ic = Convert.ToDouble(this.dataDataSet.Giris.Rows[0][2]); //
 İç ortam sıcaklığı
            rts_id = Convert.ToInt16(this.dataDataSet.Giris.Rows[0][6]); /
/ RTS no
            duvar_id = Convert.ToInt16(duvar_alt_comboBox.Text); // Duvar
CTS no
            duvar_en = Convert.ToDouble(duvar_en_textBox.Text); // Duvar e
ni
            duvar_boy = Convert.ToDouble(duvar_boy_textBox.Text); // Duvar
 boyu
            Uduvar = Convert.ToDouble(this.dataDataSet.Duvar_Alt.Rows[duva
r_id - 1][2]); // Duvar ısı transfer katsayısı
            // Pencere adedi girilmezse pencere alanını sıfır olarak belir
ler.
            if (pencere_adet_textBox.Text == "") { pencere_adet = 0; pence
re_en = 0; pencere_boy = 0; }
            else
            {
                pencere_adet = Convert.ToInt16(pencere_adet_textBox.Text);
 // Pencere adedi
                pencere_en = Convert.ToDouble(pencere_en_textBox.Text); //
 Pencere eni
                pencere_boy = Convert.ToDouble(pencere_boy_textBox.Text);
// Pencere boyu
                Upencere = Convert.ToDouble(pencere_u_label.Text); // Penc
ere ısı transfer katsayısı

            }
            //
            // Kapı adedi girilmezse kapı alanını sıfır olarak belirler.
            if (kapi_adet_textBox.Text == "") { kapi_adet = 0; kapi_en = 0
; kapi_boy = 0; }
            else
            {
                kapi_adet = Convert.ToInt16(kapi_adet_textBox.Text); // Ka
pı adedi
                kapi_en = Convert.ToDouble(kapi_en_textBox.Text); // Kapı
eni
                kapi_boy = Convert.ToDouble(kapi_boy_textBox.Text); // Kap
ı boyu
                Ukapi = Convert.ToDouble(kapi_u_label.Text); // Kapı ısı t
ransfer katsayısı
            }
            //
        }
        void alan_hesapla() // Net yapı alanlarını hesaplar.
        {
            pencere_alan = pencere_adet * pencere_en * pencere_boy; // Pen
cere alanı
            kapi_alan = kapi_adet * kapi_en * kapi_boy; // Kapı alanı
            duvar_alan = duvar_en * duvar_boy -
 (pencere_alan + kapi_alan); // Net duvar alanı
            // Hesaplanan alanları ekrana yazdırır.
            duvar_alan_label.Text = string.Format("{0:0.##} m²", duvar_ala
n);
            pencere_alan_label.Text = string.Format("{0:0.##} m²", pencere
_alan);
            kapi_alan_label.Text = string.Format("{0:0.##} m²", kapi_alan)
;
```

```
            //
        }
        void cephe_radyasyon_hesapla() // Cepheye gelen saatlik güneş ışın
ımını hesaplar.
        {
            // Veritabanından verileri çeker.
            this.radyasyonTableAdapter.Fill(this.dataDataSet.Radyasyon);
            this.cephe3TableAdapter.Fill(this.dataDataSet.Cephe3);
            //
            for (int i = 0; i < 24; i++)
            {
                double Beta = Convert.ToDouble(this.dataDataSet.Radyasyon.
Rows[i][6]); // Güneş irtifa açısı
                double Phi = Convert.ToDouble(this.dataDataSet.Radyasyon.R
ows[i][7]); // Güneş azimut açısı
                double Eb = Convert.ToDouble(this.dataDataSet.Radyasyon.Ro
ws[i][9]); // Direkt güneş ışınımı
                double Ed = Convert.ToDouble(this.dataDataSet.Radyasyon.Ro
ws[i][10]); // Yayılan güneş ışınımı
                double Gamma = Ch14.getGamma(Phi, Psi); // Yüzey-
güneş azimut açısı
                double Theta = Ch14.getThetaVer(Beta, Gamma); // Geliş açı
sı
                double Etb = Ch14.getEtb(Eb, Theta); // Yüzeye gelen direk
t güneş ışınımı
                double Etd = Ch14.getEtd(Ed, Theta, 90); // Yüzeye yayılan
 güneş ışınımı
                double Etr = Ch14.getEtrVer(Eb, Beta, Ed); // Yerden yüzey
e yansıyan güneş ışınımı
                double Et = Ch14.GetEt(Etb, Etd, Etr); // Toplam güneş ışı
nımı
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Cephe3.Rows[i][2] = Gamma;
                this.dataDataSet.Cephe3.Rows[i][3] = Theta;
                this.dataDataSet.Cephe3.Rows[i][4] = Etb;
                this.dataDataSet.Cephe3.Rows[i][5] = Etd;
                this.dataDataSet.Cephe3.Rows[i][6] = Etr;
                this.dataDataSet.Cephe3.Rows[i][7] = Et;
                //
            }
            this.cephe3TableAdapter.Update(this.dataDataSet.Cephe3); // Ve
ritabanını günceller.
        }
        void duvar_Q_hesapla() // Duvardan saatlik toplam soğutma yükünü h
esaplar.
        {
            // Veritabanından verileri çeker.
            this.duvar3TableAdapter.Fill(this.dataDataSet.Duvar3);
            this.qtTableAdapter.Fill(this.dataDataSet.Qt);
            //
            for (int i = 0; i < 24; i++) // Saatlik eşdeğer sıcaklık ve ıs
ı girişini hesaplar.
            {
                double t_dis = Convert.ToDouble(this.dataDataSet.Radyasyon
.Rows[i][3]); // Dış ortam sıcaklığı
                double Et = Convert.ToDouble(this.dataDataSet.Cephe3.Rows[
i][7]); // Toplam güneş ışınımı
                double te = Ch18.get_teVer(t_dis, Et, alphah0); // Eşdeğer
 sıcaklık
```

```
                double qi = Ch18.get_qi(Uduvar, duvar_alan, te, t_ic); //
Isı girişi
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Duvar3.Rows[i][2] = t_dis;
                this.dataDataSet.Duvar3.Rows[i][3] = te;
                this.dataDataSet.Duvar3.Rows[i][4] = t_ic;
                this.dataDataSet.Duvar3.Rows[i][5] = qi;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için iletim zaman ser
ilerini uygular.
            {
                double qcond = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double c = Convert.ToDouble(this.dataDataSet.Duvar_CTS
.Rows[j][duvar_id]) / 100; // İletim zaman faktörü
                    double qi = Convert.ToDouble(this.dataDataSet.Duvar3.R
ows[saat][5]); // Isı girişi
                    qcond += c * qi; // İletim ısı kazancı
                    saat--;
                }
                double qconv = 0.54 * qcond; // Taşınımsal ısı kazancı
                double qrad = 0.46 * qcond; // Işınımsal ısı kazancı
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Duvar3.Rows[i][6] = Convert.ToDouble(this
.dataDataSet.Duvar_CTS.Rows[i][duvar_id]);
                this.dataDataSet.Duvar3.Rows[i][7] = qcond;
                this.dataDataSet.Duvar3.Rows[i][8] = qconv;
                this.dataDataSet.Duvar3.Rows[i][9] = qrad;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r = Convert.ToDouble(this.dataDataSet.RTS_Nonso
lar.Rows[j][rts_id]) / 100; // Işınım zaman faktörü
                    double qrad = Convert.ToDouble(this.dataDataSet.Duvar3
.Rows[saat][9]); // Işınımsal ısı kazancı
                    Qr += r * qrad; // Işınımsal soğutma yükü
                    saat--;
                }
                double qconv = Convert.ToDouble(this.dataDataSet.Duvar3.Ro
ws[i][8]); // Taşınımsal soğutma yükü
                double Qt = qconv + Qr; // Toplam saatlik soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Duvar3.Rows[i][10] = Convert.ToDouble(thi
s.dataDataSet.RTS_Nonsolar.Rows[i][rts_id]);
                this.dataDataSet.Duvar3.Rows[i][11] = Qr;
                this.dataDataSet.Duvar3.Rows[i][12] = Qt;
                this.dataDataSet.Qt.Rows[i][5] = Qt;
                //
            }
            // Veritabanını günceller.
```

```
            this.duvar3TableAdapter.Update(this.dataDataSet.Duvar3);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        void pencere_Q_hesapla() // Pencereden saatlik toplam soğutma yükü
nü hesaplar.
        {
            this.pencere3TableAdapter.Fill(this.dataDataSet.Pencere3); //
Veritabanından verileri çeker.
            int cam_id = Convert.ToInt16(pencere_ana_comboBox.SelectedInde
x);
            double shgc0 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.R
ows[cam_id][2]);
            double shgc40 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][3]);
            double shgc50 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][4]);
            double shgc60 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][5]);
            double shgc70 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][6]);
            double shgc80 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][7]);
            double shgc90 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][8]);
            double shgc_d = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][9]);
            for (int i = 0; i < 24; i++) // Saatlik ısı kazançlarını hesap
lar.
            {
                double shgc; // Güneş ısı kazanç katsayısı
                double Theta = Convert.ToDouble(this.dataDataSet.Cephe3.Ro
ws[i][3]); // Geliş açısı
                double t_dis = Convert.ToDouble(this.dataDataSet.Radyasyon
.Rows[i][3]); // Dış ortam sıcaklığı
                double Etb = Convert.ToDouble(this.dataDataSet.Cephe3.Rows
[i][4]); // Yüzeye gelen direkt güneş ışınımı
                double Etd = Convert.ToDouble(this.dataDataSet.Cephe3.Rows
[i][5]); // Yüzeye yayılan güneş ışınımı
                double Etr = Convert.ToDouble(this.dataDataSet.Cephe3.Rows
[i][6]); // Yerden yüzeye yansıyan güneş ışınımı
                // interpolasyon ile SHGC değerini belirler.
                if (Theta <= 0 && Theta > 40) shgc = shgc0 + (Theta -
 0) * (shgc40 - shgc0) / (40 - 0);
                else if (Theta >= 40 && Theta < 50) shgc = shgc40 + (Theta
 - 40) * (shgc50 - shgc40) / (50 - 40);
                else if (Theta >= 50 && Theta < 60) shgc = shgc50 + (Theta
 - 50) * (shgc60 - shgc50) / (60 - 50);
                else if (Theta >= 60 && Theta < 70) shgc = shgc60 + (Theta
 - 60) * (shgc70 - shgc60) / (70 - 60);
                else if (Theta >= 70 && Theta < 80) shgc = shgc70 + (Theta
 - 70) * (shgc80 - shgc70) / (80 - 70);
                else if (Theta >= 80 && Theta < 90) shgc = shgc80 + (Theta
 - 80) * (shgc90 - shgc80) / (90 - 80);
                else shgc = 0;
                //
                double qbeam = Ch18.get_qbeam(pencere_alan, Etb, shgc); //
 Direkt güneş ısı kazancı
```

```
                double qdiffuse = Ch18.get_qdiffuse(pencere_alan, Etd, Etr
, shgc_d); // Yayılan güneş ısı kazancı
                double qcond = Ch18.get_qi(Upencere, pencere_alan, t_dis,
t_ic); // İletim ısı kazancı
                double qconv, qrad;
                if (shgc_d <= 0.5)
                {
                    qconv = 0.54 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.46 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                else
                {
                    qconv = 0.67 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.33 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Pencere3.Rows[i][2] = t_dis;
                this.dataDataSet.Pencere3.Rows[i][3] = t_ic;
                this.dataDataSet.Pencere3.Rows[i][4] = Etb;
                this.dataDataSet.Pencere3.Rows[i][5] = shgc;
                this.dataDataSet.Pencere3.Rows[i][6] = qbeam;
                this.dataDataSet.Pencere3.Rows[i][9] = Etd;
                this.dataDataSet.Pencere3.Rows[i][10] = Etr;
                this.dataDataSet.Pencere3.Rows[i][11] = shgc_d;
                this.dataDataSet.Pencere3.Rows[i][12] = qdiffuse;
                this.dataDataSet.Pencere3.Rows[i][13] = qcond;
                this.dataDataSet.Pencere3.Rows[i][14] = qconv;
                this.dataDataSet.Pencere3.Rows[i][15] = qrad;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qb = 0; double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r_solar = Convert.ToDouble(this.dataDataSet.RTS
_Solar.Rows[j][rts_id]) / 100; // Güneşe bağlı ışınım zaman faktörü
                    double r_nonsolar = Convert.ToDouble(this.dataDataSet.
RTS_Nonsolar.Rows[j][rts_id]) / 100; // Güneşe bağlı olmayan ışınım zaman
faktörü
                    double qbeam = Convert.ToDouble(this.dataDataSet.Pence
re3.Rows[saat][6]); // Direkt güneş ısı kazancı
                    double qrad = Convert.ToDouble(this.dataDataSet.Pencer
e3.Rows[saat][15]); // Yayılan güneş+iletim ışınımsal ısı kazancı
                    Qb += r_solar * qbeam; // Direkt güneş soğutma yükü
                    Qr += r_nonsolar * qrad; // Yayılan güneş+iletim ışını
msal soğutma yükü
                    saat--;
                }
                double qconv = Convert.ToDouble(this.dataDataSet.Pencere3.
Rows[i][14]); // Yayılan güneş+iletim taşınımsal soğutma yükü
                double Qdc = qconv + Qr; // Yayılan güneş+iletim soğutma y
ükü
```

```
                double Qt = Qb + Qdc; // Toplam soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Pencere3.Rows[i][7] = Convert.ToDouble(th
is.dataDataSet.RTS_Solar[i][rts_id]);
                this.dataDataSet.Pencere3.Rows[i][16] = Convert.ToDouble(t
his.dataDataSet.RTS_Nonsolar[i][rts_id]);
                this.dataDataSet.Pencere3.Rows[i][8] = Qb;
                this.dataDataSet.Pencere3.Rows[i][17] = Qr;
                this.dataDataSet.Pencere3.Rows[i][18] = Qdc;
                this.dataDataSet.Pencere3.Rows[i][19] = Qt;
                this.dataDataSet.Qt.Rows[i][5] = Convert.ToDouble(this.dat
aDataSet.Qt.Rows[i][5]) + Qt;
                //
            }
            // Veritabanını günceller.
            this.pencere3TableAdapter.Update(this.dataDataSet.Pencere3);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        void kapi_Q_hesapla() //Kapıdan saatlik toplam soğutma yükünü esap
lar.
        {
            this.kapi3TableAdapter.Fill(this.dataDataSet.Kapi3); // Verita
banından verileri çeker.
            int cam_id = Convert.ToInt16(kapi_ana_comboBox.SelectedIndex);
            double shgc0 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.R
ows[cam_id][2]);
            double shgc40 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][3]);
            double shgc50 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][4]);
            double shgc60 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][5]);
            double shgc70 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][6]);
            double shgc80 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][7]);
            double shgc90 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][8]);
            double shgc_d = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][9]);
            for (int i = 0; i < 24; i++) // Saatlik ısı kazançlarını hesap
lar.
            {
                double shgc; // Güneş ısı kazanç katsayısı
                double Theta = Convert.ToDouble(this.dataDataSet.Cephe3.Ro
ws[i][3]); // Geliş açısı
                double t_dis = Convert.ToDouble(this.dataDataSet.Radyasyon
.Rows[i][3]); // Dış ortam sıcaklığı
                double Etb = Convert.ToDouble(this.dataDataSet.Cephe3.Rows
[i][4]); // Yüzeye gelen direkt güneş ışınımı
                double Etd = Convert.ToDouble(this.dataDataSet.Cephe3.Rows
[i][5]); // Yüzeye yayılan güneş ışınımı
                double Etr = Convert.ToDouble(this.dataDataSet.Cephe3.Rows
[i][6]); // Yerden yüzeye yansıyan güneş ışınımı
                // interpolasyon ile SHGC değerini belirler.
                if (Theta <= 0 && Theta > 40) shgc = shgc0 + (Theta -
 0) * (shgc40 - shgc0) / (40 - 0);
```

```
                else if (Theta >= 40 && Theta < 50) shgc = shgc40 + (Theta
 - 40) * (shgc50 - shgc40) / (50 - 40);
                else if (Theta >= 50 && Theta < 60) shgc = shgc50 + (Theta
 - 50) * (shgc60 - shgc50) / (60 - 50);
                else if (Theta >= 60 && Theta < 70) shgc = shgc60 + (Theta
 - 60) * (shgc70 - shgc60) / (70 - 60);
                else if (Theta >= 70 && Theta < 80) shgc = shgc70 + (Theta
 - 70) * (shgc80 - shgc70) / (80 - 70);
                else if (Theta >= 80 && Theta < 90) shgc = shgc80 + (Theta
 - 80) * (shgc90 - shgc80) / (90 - 80);
                else shgc = 0;
                //
                double qbeam = Ch18.get_qbeam(kapi_alan, Etb, shgc); // Di
rekt güneş ısı kazancı
                double qdiffuse = Ch18.get_qdiffuse(kapi_alan, Etd, Etr, s
hgc_d); // Yayılan güneş ısı kazancı
                double qcond = Ch18.get_qi(Ukapi, kapi_alan, t_dis, t_ic);
 // İletim ısı kazancı
                double qconv, qrad;
                if (shgc_d <= 0.5)
                {
                    qconv = 0.54 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.46 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                else
                {
                    qconv = 0.67 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.33 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Kapi3.Rows[i][2] = t_dis;
                this.dataDataSet.Kapi3.Rows[i][3] = t_ic;
                this.dataDataSet.Kapi3.Rows[i][4] = Etb;
                this.dataDataSet.Kapi3.Rows[i][5] = shgc;
                this.dataDataSet.Kapi3.Rows[i][6] = qbeam;
                this.dataDataSet.Kapi3.Rows[i][9] = Etd;
                this.dataDataSet.Kapi3.Rows[i][10] = Etr;
                this.dataDataSet.Kapi3.Rows[i][11] = shgc_d;
                this.dataDataSet.Kapi3.Rows[i][12] = qdiffuse;
                this.dataDataSet.Kapi3.Rows[i][13] = qcond;
                this.dataDataSet.Kapi3.Rows[i][14] = qconv;
                this.dataDataSet.Kapi3.Rows[i][15] = qrad;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qb = 0; double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r_solar = Convert.ToDouble(this.dataDataSet.RTS
_Solar.Rows[j][rts_id]) / 100; // Güneşe bağlı ışınım zaman faktörü
```

```
                    double r_nonsolar = Convert.ToDouble(this.dataDataSet.
RTS_Nonsolar.Rows[j][rts_id]) / 100; // Güneşe bağlı olmayan ışınım zaman
faktörü
                    double qbeam = Convert.ToDouble(this.dataDataSet.Kapi3
.Rows[saat][6]); // Direkt güneş ısı kazancı
                    double qrad = Convert.ToDouble(this.dataDataSet.Kapi3.
Rows[saat][15]); // Yayılan güneş+iletim ışınımsal ısı kazancı
                    Qb += r_solar * qbeam; // Direkt güneş soğutma yükü
                    Qr += r_nonsolar * qrad; // Yayılan güneş+iletim ışını
msal soğutma yükü
                    saat--;
                }
                double qconv = Convert.ToDouble(this.dataDataSet.Kapi3.Row
s[i][14]); // Yayılan güneş+iletim taşınımsal soğutma yükü
                double Qdc = qconv + Qr; // Yayılan güneş+iletim soğutma y
ükü
                double Qt = Qb + Qdc; // Toplam soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Kapi3.Rows[i][7] = Convert.ToDouble(this.
dataDataSet.RTS_Solar[i][rts_id]);
                this.dataDataSet.Kapi3.Rows[i][16] = Convert.ToDouble(this
.dataDataSet.RTS_Nonsolar[i][rts_id]);
                this.dataDataSet.Kapi3.Rows[i][8] = Qb;
                this.dataDataSet.Kapi3.Rows[i][17] = Qr;
                this.dataDataSet.Kapi3.Rows[i][18] = Qdc;
                this.dataDataSet.Kapi3.Rows[i][19] = Qt;
                this.dataDataSet.Qt.Rows[i][5] = Convert.ToDouble(this.dat
aDataSet.Qt.Rows[i][5]) + Qt;
                //
            }
            // Veritabanını günceller.
            this.kapi3TableAdapter.Update(this.dataDataSet.Kapi3);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        private void Cephe3_Load(object sender, EventArgs e)
        {
            // Veritabanından verileri çeker.
            this.duvar_AnaTableAdapter.Fill(this.dataDataSet.Duvar_Ana);
            this.duvar_AltTableAdapter.Fill(this.dataDataSet.Duvar_Alt);
            this.duvar_YapiTableAdapter.Fill(this.dataDataSet.Duvar_Yapi);
            this.pencere_AnaTableAdapter.Fill(this.dataDataSet.Pencere_Ana
);
            this.pencere_AltTableAdapter.Fill(this.dataDataSet.Pencere_Alt
);
            this.pencere_CerceveTableAdapter.Fill(this.dataDataSet.Pencere
_Cerceve);
            this.duvar_CTSTableAdapter.Fill(this.dataDataSet.Duvar_CTS);
            this.rTS_NonsolarTableAdapter.Fill(this.dataDataSet.RTS_Nonsol
ar);
            this.rTS_SolarTableAdapter.Fill(this.dataDataSet.RTS_Solar);
            //
        }
        // En ve boy girişinin yapıldığı metin kutularında sadece sayı, vi
rgül, del ve backspace tuşlarını aktif eder.
        private void duvar_en_textBox_KeyPress(object sender, KeyPressEven
tArgs e)
        {
```

```
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)44 && e.KeyChar != (char)127;
        }
        // Adet girişinin yapıldığı metin kutularında sadece sayı, del ve
backspace tuşlarını aktif eder.
        private void pencere_adet_textBox_KeyPress(object sender, KeyPress
EventArgs e)
        {
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)127;
        }
        //
        private void hesapla_button_Click(object sender, EventArgs e)
        {
            veri_cek();
            alan_hesapla();
            cephe_radyasyon_hesapla();
            duvar_Q_hesapla();
            if (pencere_alan != 0) pencere_Q_hesapla(); // Pencere alanı s
ıfırdan farklı ise soğutma yükü hesaplar.
            if (kapi_alan != 0) kapi_Q_hesapla(); // Kapı alanı sıfırdan f
arklı ise soğutma yükü hesaplar.
            this.AutoScroll = true; // Sayfa kaydırmayı aktif eder.
            cephe_radyasyon_groupBox.Visible = true; // Hesaplanan değerle
ri ekranda gösterir.
            kontrol = true; // Hesapla butonu tıklama kontrolü
        }
    }
}
```

EK C.9 Cephe4 Ekranı Kaynak Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Cephe4 Ekranı Kaynak Kodları
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BuharlasmaliSogutucuHesapProgrami
{
    public partial class Cephe4 : UserControl
    {
        public Cephe4()
        {
            InitializeComponent();
        }
        // Fonksiyon dosyasını bağlar.
        fonksiyon.Bolum14 Ch14 = new fonksiyon.Bolum14();
        fonksiyon.Bolum18 Ch18 = new fonksiyon.Bolum18();
        //
        // Değişken tanımları
        int rts_id, duvar_id, pencere_adet, kapi_adet;
        double t_ic, Psi, alphah0, duvar_en, duvar_boy, duvar_alan, Uduvar
, pencere_en, pencere_boy, pencere_alan, Upencere, kapi_en, kapi_boy, kapi
_alan, Ukapi;
        public bool kontrol = false; // Hesapla butonu tıklama kontrol değ
işkeni
        //
        void veri_cek() // Metin kutuları, radyo butonları ve veri tabanın
dan verileri çeker.
        {
            this.girisTableAdapter.Fill(this.dataDataSet.Giris);
            // Yüzey azimut açısını belirler.
            if (k_radioButton.Checked) Psi = Ch14.kuzey;
            else if (kd_radioButton.Checked) Psi = Ch14.kuzeydogu;
            else if (d_radioButton.Checked) Psi = Ch14.dogu;
            else if (gd_radioButton.Checked) Psi = Ch14.guneydogu;
            else if (g_radioButton.Checked) Psi = Ch14.guney;
            else if (gb_radioButton.Checked) Psi = Ch14.guneybati;
            else if (b_radioButton.Checked) Psi = Ch14.bati;
            else if (kb_radioButton.Checked) Psi = Ch14.kuzeybati;
            //
            // alpha/h0 değerini belirler.
            if (acik_radioButton.Checked) alphah0 = Ch14.acik_renk;
            else if (koyu_radioButton.Checked) alphah0 = Ch14.koyu_renk;
            //
```

```
            t_ic = Convert.ToDouble(this.dataDataSet.Giris.Rows[0][2]); //
 İç ortam sıcaklığı
            rts_id = Convert.ToInt16(this.dataDataSet.Giris.Rows[0][6]); /
/ RTS no
            duvar_id = Convert.ToInt16(duvar_alt_comboBox.Text); // Duvar
CTS no
            duvar_en = Convert.ToDouble(duvar_en_textBox.Text); // Duvar e
ni
            duvar_boy = Convert.ToDouble(duvar_boy_textBox.Text); // Duvar
 boyu
            Uduvar = Convert.ToDouble(this.dataDataSet.Duvar_Alt.Rows[duva
r_id - 1][2]); // Duvar ısı transfer katsayısı
            // Pencere adedi girilmezse pencere alanını sıfır olarak belir
ler.
            if (pencere_adet_textBox.Text == "") { pencere_adet = 0; pence
re_en = 0; pencere_boy = 0; }
            else
            {
                pencere_adet = Convert.ToInt16(pencere_adet_textBox.Text);
 // Pencere adedi
                pencere_en = Convert.ToDouble(pencere_en_textBox.Text); //
 Pencere eni
                pencere_boy = Convert.ToDouble(pencere_boy_textBox.Text);
// Pencere boyu
                Upencere = Convert.ToDouble(pencere_u_label.Text); // Penc
ere ısı transfer katsayısı

            }
            //
            // Kapı adedi girilmezse kapı alanını sıfır olarak belirler.
            if (kapi_adet_textBox.Text == "") { kapi_adet = 0; kapi_en = 0
; kapi_boy = 0; }
            else
            {
                kapi_adet = Convert.ToInt16(kapi_adet_textBox.Text); // Ka
pı adedi
                kapi_en = Convert.ToDouble(kapi_en_textBox.Text); // Kapı
eni
                kapi_boy = Convert.ToDouble(kapi_boy_textBox.Text); // Kap
ı boyu
                Ukapi = Convert.ToDouble(kapi_u_label.Text); // Kapı ısı t
ransfer katsayısı
            }
            //
        }
        void alan_hesapla() // Net yapı alanlarını hesaplar.
        {
            pencere_alan = pencere_adet * pencere_en * pencere_boy; // Pen
cere alanı
            kapi_alan = kapi_adet * kapi_en * kapi_boy; // Kapı alanı
            duvar_alan = duvar_en * duvar_boy -
 (pencere_alan + kapi_alan); // Net duvar alanı
            // Hesaplanan alanları ekrana yazdırır.
            duvar_alan_label.Text = string.Format("{0:0.##} m²", duvar_ala
n);
            pencere_alan_label.Text = string.Format("{0:0.##} m²", pencere
_alan);
            kapi_alan_label.Text = string.Format("{0:0.##} m²", kapi_alan)
;
```

```
            //
        }
        void cephe_radyasyon_hesapla() // Cepheye gelen saatlik güneş ışın
ımını hesaplar.
        {
            // Veritabanından verileri çeker.
            this.radyasyonTableAdapter.Fill(this.dataDataSet.Radyasyon);
            this.cephe4TableAdapter.Fill(this.dataDataSet.Cephe4);
            //
            for (int i = 0; i < 24; i++)
            {
                double Beta = Convert.ToDouble(this.dataDataSet.Radyasyon.
Rows[i][6]); // Güneş irtifa açısı
                double Phi = Convert.ToDouble(this.dataDataSet.Radyasyon.R
ows[i][7]); // Güneş azimut açısı
                double Eb = Convert.ToDouble(this.dataDataSet.Radyasyon.Ro
ws[i][9]); // Direkt güneş ışınımı
                double Ed = Convert.ToDouble(this.dataDataSet.Radyasyon.Ro
ws[i][10]); // Yayılan güneş ışınımı
                double Gamma = Ch14.getGamma(Phi, Psi); // Yüzey-
güneş azimut açısı
                double Theta = Ch14.getThetaVer(Beta, Gamma); // Geliş açı
sı
                double Etb = Ch14.getEtb(Eb, Theta); // Yüzeye gelen direk
t güneş ışınımı
                double Etd = Ch14.getEtd(Ed, Theta, 90); // Yüzeye yayılan
 güneş ışınımı
                double Etr = Ch14.getEtrVer(Eb, Beta, Ed); // Yerden yüzey
e yansıyan güneş ışınımı
                double Et = Ch14.GetEt(Etb, Etd, Etr); // Toplam güneş ışı
nımı
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Cephe4.Rows[i][2] = Gamma;
                this.dataDataSet.Cephe4.Rows[i][3] = Theta;
                this.dataDataSet.Cephe4.Rows[i][4] = Etb;
                this.dataDataSet.Cephe4.Rows[i][5] = Etd;
                this.dataDataSet.Cephe4.Rows[i][6] = Etr;
                this.dataDataSet.Cephe4.Rows[i][7] = Et;
                //
            }
            this.cephe4TableAdapter.Update(this.dataDataSet.Cephe4); // Ve
ritabanını günceller.
        }
        void duvar_Q_hesapla() // Duvardan saatlik toplam soğutma yükünü h
esaplar.
        {
            // Veritabanından verileri çeker.
            this.duvar4TableAdapter.Fill(this.dataDataSet.Duvar4);
            this.qtTableAdapter.Fill(this.dataDataSet.Qt);
            //
            for (int i = 0; i < 24; i++) // Saatlik eşdeğer sıcaklık ve ıs
ı girişini hesaplar.
            {
                double t_dis = Convert.ToDouble(this.dataDataSet.Radyasyon
.Rows[i][3]); // Dış ortam sıcaklığı
                double Et = Convert.ToDouble(this.dataDataSet.Cephe4.Rows[
i][7]); // Toplam güneş ışınımı
                double te = Ch18.get_teVer(t_dis, Et, alphah0); // Eşdeğer
 sıcaklık
```

```
                double qi = Ch18.get_qi(Uduvar, duvar_alan, te, t_ic); //
Isı girişi
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Duvar4.Rows[i][2] = t_dis;
                this.dataDataSet.Duvar4.Rows[i][3] = te;
                this.dataDataSet.Duvar4.Rows[i][4] = t_ic;
                this.dataDataSet.Duvar4.Rows[i][5] = qi;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için iletim zaman ser
ilerini uygular.
            {
                double qcond = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double c = Convert.ToDouble(this.dataDataSet.Duvar_CTS
.Rows[j][duvar_id]) / 100; // İletim zaman faktörü
                    double qi = Convert.ToDouble(this.dataDataSet.Duvar4.R
ows[saat][5]); // Isı girişi
                    qcond += c * qi; // İletim ısı kazancı
                    saat--;
                }
                double qconv = 0.54 * qcond; // Taşınımsal ısı kazancı
                double qrad = 0.46 * qcond; // Işınımsal ısı kazancı
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Duvar4.Rows[i][6] = Convert.ToDouble(this
.dataDataSet.Duvar_CTS.Rows[i][duvar_id]);
                this.dataDataSet.Duvar4.Rows[i][7] = qcond;
                this.dataDataSet.Duvar4.Rows[i][8] = qconv;
                this.dataDataSet.Duvar4.Rows[i][9] = qrad;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r = Convert.ToDouble(this.dataDataSet.RTS_Nonso
lar.Rows[j][rts_id]) / 100; // Işınım zaman faktörü
                    double qrad = Convert.ToDouble(this.dataDataSet.Duvar4
.Rows[saat][9]); // Işınımsal ısı kazancı
                    Qr += r * qrad; // Işınımsal soğutma yükü
                    saat--;
                }
                double qconv = Convert.ToDouble(this.dataDataSet.Duvar4.Ro
ws[i][8]); // Taşınımsal soğutma yükü
                double Qt = qconv + Qr; // Toplam saatlik soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Duvar4.Rows[i][10] = Convert.ToDouble(thi
s.dataDataSet.RTS_Nonsolar.Rows[i][rts_id]);
                this.dataDataSet.Duvar4.Rows[i][11] = Qr;
                this.dataDataSet.Duvar4.Rows[i][12] = Qt;
                this.dataDataSet.Qt.Rows[i][6] = Qt;
                //
            }
            // Veritabanını günceller.
```

```
            this.duvar4TableAdapter.Update(this.dataDataSet.Duvar4);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        void pencere_Q_hesapla() // Pencereden saatlik toplam soğutma yükü
nü hesaplar.
        {
            this.pencere4TableAdapter.Fill(this.dataDataSet.Pencere4); //
Veritabanından verileri çeker.
            int cam_id = Convert.ToInt16(pencere_ana_comboBox.SelectedInde
x);
            double shgc0 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.R
ows[cam_id][2]);
            double shgc40 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][3]);
            double shgc50 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][4]);
            double shgc60 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][5]);
            double shgc70 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][6]);
            double shgc80 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][7]);
            double shgc90 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][8]);
            double shgc_d = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][9]);
            for (int i = 0; i < 24; i++) // Saatlik ısı kazançlarını hesap
lar.
            {
                double shgc; // Güneş ısı kazanç katsayısı
                double Theta = Convert.ToDouble(this.dataDataSet.Cephe4.Ro
ws[i][3]); // Geliş açısı
                double t_dis = Convert.ToDouble(this.dataDataSet.Radyasyon
.Rows[i][3]); // Dış ortam sıcaklığı
                double Etb = Convert.ToDouble(this.dataDataSet.Cephe4.Rows
[i][4]); // Yüzeye gelen direkt güneş ışınımı
                double Etd = Convert.ToDouble(this.dataDataSet.Cephe4.Rows
[i][5]); // Yüzeye yayılan güneş ışınımı
                double Etr = Convert.ToDouble(this.dataDataSet.Cephe4.Rows
[i][6]); // Yerden yüzeye yansıyan güneş ışınımı
                // interpolasyon ile SHGC değerini belirler.
                if (Theta <= 0 && Theta > 40) shgc = shgc0 + (Theta -
 0) * (shgc40 - shgc0) / (40 - 0);
                else if (Theta >= 40 && Theta < 50) shgc = shgc40 + (Theta
 - 40) * (shgc50 - shgc40) / (50 - 40);
                else if (Theta >= 50 && Theta < 60) shgc = shgc50 + (Theta
 - 50) * (shgc60 - shgc50) / (60 - 50);
                else if (Theta >= 60 && Theta < 70) shgc = shgc60 + (Theta
 - 60) * (shgc70 - shgc60) / (70 - 60);
                else if (Theta >= 70 && Theta < 80) shgc = shgc70 + (Theta
 - 70) * (shgc80 - shgc70) / (80 - 70);
                else if (Theta >= 80 && Theta < 90) shgc = shgc80 + (Theta
 - 80) * (shgc90 - shgc80) / (90 - 80);
                else shgc = 0;
                //
                double qbeam = Ch18.get_qbeam(pencere_alan, Etb, shgc); //
 Direkt güneş ısı kazancı
```

```
                double qdiffuse = Ch18.get_qdiffuse(pencere_alan, Etd, Etr
, shgc_d); // Yayılan güneş ısı kazancı
                double qcond = Ch18.get_qi(Upencere, pencere_alan, t_dis,
t_ic); // İletim ısı kazancı
                double qconv, qrad;
                if (shgc_d <= 0.5)
                {
                    qconv = 0.54 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.46 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                else
                {
                    qconv = 0.67 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.33 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Pencere4.Rows[i][2] = t_dis;
                this.dataDataSet.Pencere4.Rows[i][3] = t_ic;
                this.dataDataSet.Pencere4.Rows[i][4] = Etb;
                this.dataDataSet.Pencere4.Rows[i][5] = shgc;
                this.dataDataSet.Pencere4.Rows[i][6] = qbeam;
                this.dataDataSet.Pencere4.Rows[i][9] = Etd;
                this.dataDataSet.Pencere4.Rows[i][10] = Etr;
                this.dataDataSet.Pencere4.Rows[i][11] = shgc_d;
                this.dataDataSet.Pencere4.Rows[i][12] = qdiffuse;
                this.dataDataSet.Pencere4.Rows[i][13] = qcond;
                this.dataDataSet.Pencere4.Rows[i][14] = qconv;
                this.dataDataSet.Pencere4.Rows[i][15] = qrad;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qb = 0; double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r_solar = Convert.ToDouble(this.dataDataSet.RTS
_Solar.Rows[j][rts_id]) / 100; // Güneşe bağlı ışınım zaman faktörü
                    double r_nonsolar = Convert.ToDouble(this.dataDataSet.
RTS_Nonsolar.Rows[j][rts_id]) / 100; // Güneşe bağlı olmayan ışınım zaman
faktörü
                    double qbeam = Convert.ToDouble(this.dataDataSet.Pence
re4.Rows[saat][6]); // Direkt güneş ısı kazancı
                    double qrad = Convert.ToDouble(this.dataDataSet.Pencer
e4.Rows[saat][15]); // Yayılan güneş+iletim ışınımsal ısı kazancı
                    Qb += r_solar * qbeam; // Direkt güneş soğutma yükü
                    Qr += r_nonsolar * qrad; // Yayılan güneş+iletim ışını
msal soğutma yükü
                    saat--;
                }
                double qconv = Convert.ToDouble(this.dataDataSet.Pencere4.
Rows[i][14]); // Yayılan güneş+iletim taşınımsal soğutma yükü
                double Qdc = qconv + Qr; // Yayılan güneş+iletim soğutma y
ükü
```

```
                double Qt = Qb + Qdc; // Toplam soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Pencere4.Rows[i][7] = Convert.ToDouble(th
is.dataDataSet.RTS_Solar[i][rts_id]);
                this.dataDataSet.Pencere4.Rows[i][16] = Convert.ToDouble(t
his.dataDataSet.RTS_Nonsolar[i][rts_id]);
                this.dataDataSet.Pencere4.Rows[i][8] = Qb;
                this.dataDataSet.Pencere4.Rows[i][17] = Qr;
                this.dataDataSet.Pencere4.Rows[i][18] = Qdc;
                this.dataDataSet.Pencere4.Rows[i][19] = Qt;
                this.dataDataSet.Qt.Rows[i][6] = Convert.ToDouble(this.dat
aDataSet.Qt.Rows[i][6]) + Qt;
                //
            }
            // Veritabanını günceller.
            this.pencere4TableAdapter.Update(this.dataDataSet.Pencere4);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        void kapi_Q_hesapla() //Kapıdan saatlik toplam soğutma yükünü esap
lar.
        {
            this.kapi4TableAdapter.Fill(this.dataDataSet.Kapi4); // Verita
banından verileri çeker.
            int cam_id = Convert.ToInt16(kapi_ana_comboBox.SelectedIndex);
            double shgc0 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.R
ows[cam_id][2]);
            double shgc40 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][3]);
            double shgc50 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][4]);
            double shgc60 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][5]);
            double shgc70 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][6]);
            double shgc80 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][7]);
            double shgc90 = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][8]);
            double shgc_d = Convert.ToDouble(this.dataDataSet.Pencere_Ana.
Rows[cam_id][9]);
            for (int i = 0; i < 24; i++) // Saatlik ısı kazançlarını hesap
lar.
            {
                double shgc; // Güneş ısı kazanç katsayısı
                double Theta = Convert.ToDouble(this.dataDataSet.Cephe4.Ro
ws[i][3]); // Geliş açısı
                double t_dis = Convert.ToDouble(this.dataDataSet.Radyasyon
.Rows[i][3]); // Dış ortam sıcaklığı
                double Etb = Convert.ToDouble(this.dataDataSet.Cephe4.Rows
[i][4]); // Yüzeye gelen direkt güneş ışınımı
                double Etd = Convert.ToDouble(this.dataDataSet.Cephe4.Rows
[i][5]); // Yüzeye yayılan güneş ışınımı
                double Etr = Convert.ToDouble(this.dataDataSet.Cephe4.Rows
[i][6]); // Yerden yüzeye yansıyan güneş ışınımı
                // interpolasyon ile SHGC değerini belirler.
                if (Theta <= 0 && Theta > 40) shgc = shgc0 + (Theta -
 0) * (shgc40 - shgc0) / (40 - 0);
```

```
                else if (Theta >= 40 && Theta < 50) shgc = shgc40 + (Theta
 - 40) * (shgc50 - shgc40) / (50 - 40);
                else if (Theta >= 50 && Theta < 60) shgc = shgc50 + (Theta
 - 50) * (shgc60 - shgc50) / (60 - 50);
                else if (Theta >= 60 && Theta < 70) shgc = shgc60 + (Theta
 - 60) * (shgc70 - shgc60) / (70 - 60);
                else if (Theta >= 70 && Theta < 80) shgc = shgc70 + (Theta
 - 70) * (shgc80 - shgc70) / (80 - 70);
                else if (Theta >= 80 && Theta < 90) shgc = shgc80 + (Theta
 - 80) * (shgc90 - shgc80) / (90 - 80);
                else shgc = 0;
                //
                double qbeam = Ch18.get_qbeam(kapi_alan, Etb, shgc); // Di
rekt güneş ısı kazancı
                double qdiffuse = Ch18.get_qdiffuse(kapi_alan, Etd, Etr, s
hgc_d); // Yayılan güneş ısı kazancı
                double qcond = Ch18.get_qi(Ukapi, kapi_alan, t_dis, t_ic);
 // İletim ısı kazancı
                double qconv, qrad;
                if (shgc_d <= 0.5)
                {
                    qconv = 0.54 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.46 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                else
                {
                    qconv = 0.67 * (qdiffuse + qcond); // Taşınımsal ısı k
azancı
                    qrad = 0.33 * (qdiffuse + qcond); // Işınımsal ısı kaz
ancı
                }
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Kapi4.Rows[i][2] = t_dis;
                this.dataDataSet.Kapi4.Rows[i][3] = t_ic;
                this.dataDataSet.Kapi4.Rows[i][4] = Etb;
                this.dataDataSet.Kapi4.Rows[i][5] = shgc;
                this.dataDataSet.Kapi4.Rows[i][6] = qbeam;
                this.dataDataSet.Kapi4.Rows[i][9] = Etd;
                this.dataDataSet.Kapi4.Rows[i][10] = Etr;
                this.dataDataSet.Kapi4.Rows[i][11] = shgc_d;
                this.dataDataSet.Kapi4.Rows[i][12] = qdiffuse;
                this.dataDataSet.Kapi4.Rows[i][13] = qcond;
                this.dataDataSet.Kapi4.Rows[i][14] = qconv;
                this.dataDataSet.Kapi4.Rows[i][15] = qrad;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qb = 0; double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r_solar = Convert.ToDouble(this.dataDataSet.RTS
_Solar.Rows[j][rts_id]) / 100; // Güneşe bağlı ışınım zaman faktörü
```

```
                    double r_nonsolar = Convert.ToDouble(this.dataDataSet.
RTS_Nonsolar.Rows[j][rts_id]) / 100; // Güneşe bağlı olmayan ışınım zaman
faktörü
                    double qbeam = Convert.ToDouble(this.dataDataSet.Kapi4
.Rows[saat][6]); // Direkt güneş ısı kazancı
                    double qrad = Convert.ToDouble(this.dataDataSet.Kapi4.
Rows[saat][15]); // Yayılan güneş+iletim ışınımsal ısı kazancı
                    Qb += r_solar * qbeam; // Direkt güneş soğutma yükü
                    Qr += r_nonsolar * qrad; // Yayılan güneş+iletim ışını
msal soğutma yükü
                    saat--;
                }
                double qconv = Convert.ToDouble(this.dataDataSet.Kapi4.Row
s[i][14]); // Yayılan güneş+iletim taşınımsal soğutma yükü
                double Qdc = qconv + Qr; // Yayılan güneş+iletim soğutma y
ükü
                double Qt = Qb + Qdc; // Toplam soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Kapi4.Rows[i][7] = Convert.ToDouble(this.
dataDataSet.RTS_Solar[i][rts_id]);
                this.dataDataSet.Kapi4.Rows[i][16] = Convert.ToDouble(this
.dataDataSet.RTS_Nonsolar[i][rts_id]);
                this.dataDataSet.Kapi4.Rows[i][8] = Qb;
                this.dataDataSet.Kapi4.Rows[i][17] = Qr;
                this.dataDataSet.Kapi4.Rows[i][18] = Qdc;
                this.dataDataSet.Kapi4.Rows[i][19] = Qt;
                this.dataDataSet.Qt.Rows[i][6] = Convert.ToDouble(this.dat
aDataSet.Qt.Rows[i][6]) + Qt;
                //
            }
            // Veritabanını günceller.
            this.kapi4TableAdapter.Update(this.dataDataSet.Kapi4);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        private void Cephe4_Load(object sender, EventArgs e)
        {
            // Veritabanından verileri çeker.
            this.duvar_AnaTableAdapter.Fill(this.dataDataSet.Duvar_Ana);
            this.duvar_AltTableAdapter.Fill(this.dataDataSet.Duvar_Alt);
            this.duvar_YapiTableAdapter.Fill(this.dataDataSet.Duvar_Yapi);
            this.pencere_AnaTableAdapter.Fill(this.dataDataSet.Pencere_Ana
);
            this.pencere_AltTableAdapter.Fill(this.dataDataSet.Pencere_Alt
);
            this.pencere_CerceveTableAdapter.Fill(this.dataDataSet.Pencere
_Cerceve);
            this.duvar_CTSTableAdapter.Fill(this.dataDataSet.Duvar_CTS);
            this.rTS_NonsolarTableAdapter.Fill(this.dataDataSet.RTS_Nonsol
ar);
            this.rTS_SolarTableAdapter.Fill(this.dataDataSet.RTS_Solar);
            //
        }
        // En ve boy girişinin yapıldığı metin kutularında sadece sayı, vi
rgül, del ve backspace tuşlarını aktif eder.
        private void duvar_en_textBox_KeyPress(object sender, KeyPressEven
tArgs e)
        {
```

```
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)44 && e.KeyChar != (char)127;
        }
        // Adet girişinin yapıldığı metin kutularında sadece sayı, del ve
backspace tuşlarını aktif eder.
        private void pencere_adet_textBox_KeyPress(object sender, KeyPress
EventArgs e)
        {
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)127;
        }
        //
        private void hesapla_button_Click(object sender, EventArgs e)
        {
            veri_cek();
            alan_hesapla();
            cephe_radyasyon_hesapla();
            duvar_Q_hesapla();
            if (pencere_alan != 0) pencere_Q_hesapla(); // Pencere alanı s
ıfırdan farklı ise soğutma yükü hesaplar.
            if (kapi_alan != 0) kapi_Q_hesapla(); // Kapı alanı sıfırdan f
arklı ise soğutma yükü hesaplar.
            this.AutoScroll = true; // Sayfa kaydırmayı aktif eder.
            cephe_radyasyon_groupBox.Visible = true; // Hesaplanan değerle
ri ekranda gösterir.
            kontrol = true; // Hesapla butonu tıklama kontrolü
        }
    }
}
```

EK C.10 Çatı Ekranı Kaynak Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Çatı Ekranı Kaynak Kodları
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BuharlasmaliSogutucuHesapProgrami
{
    public partial class Cati : UserControl
    {
        public Cati()
        {
            InitializeComponent();
        }
        // Fonksiyon dosyasını bağlar.
        fonksiyon.Bolum14 Ch14 = new fonksiyon.Bolum14();
        fonksiyon.Bolum18 Ch18 = new fonksiyon.Bolum18();
        //
        // Değişken tanımları
        int rts_id, cati_id;
        double t_ic, alphah0, en, boy, alan, Ucati;
        public bool kontrol = false; // Hesapla butonu tıklama kontrol değ
işkeni
        //
        void veri_cek() // Metin kutuları, radyo butonları ve veri tabanın
dan verileri çeker.
        {
            this.girisTableAdapter.Fill(this.dataDataSet.Giris);
            // alpha/h0 değerini belirler.
            if (acik_radioButton.Checked) alphah0 = Ch14.acik_renk;
            else if (koyu_radioButton.Checked) alphah0 = Ch14.koyu_renk;
            //
            t_ic = Convert.ToDouble(this.dataDataSet.Giris.Rows[0][2]);
            rts_id = Convert.ToInt16(this.dataDataSet.Giris.Rows[0][6]);
            cati_id = Convert.ToInt16(cati_alt_comboBox.Text);
            Ucati = Convert.ToDouble(this.dataDataSet.Cati_Alt.Rows[cati_i
d - 1][2]);
            en = Convert.ToDouble(en_textBox.Text);
            boy = Convert.ToDouble(boy_textBox.Text);
        }
        void alan_hesapla() // Çatı alanını hesaplar.
        {
            alan = en * boy;
            alan_label.Text = string.Format("Çatı Alanı: {0:0.##} m²", ala
n); // Çatı alanını ekrana yazdırır.
```

```
        }
        void radyasyon_hesapla() // Cepheye gelen saatlik güneş ışınımını
hesaplar.
        {
            // Veritabanından verileri çeker.
            this.radyasyonTableAdapter.Fill(this.dataDataSet.Radyasyon);
            this.cephe_CatiTableAdapter.Fill(this.dataDataSet.Cephe_Cati);
            //
            for (int i = 0; i < 24; i++)
            {
                double Beta = Convert.ToDouble(this.dataDataSet.Radyasyon.
Rows[i][6]); // Güneş irtifa açısı
                double Phi = Convert.ToDouble(this.dataDataSet.Radyasyon.R
ows[i][7]); // Güneş azimut açısı
                double Eb = Convert.ToDouble(this.dataDataSet.Radyasyon.Ro
ws[i][9]); // Direkt güneş ışınımı
                double Ed = Convert.ToDouble(this.dataDataSet.Radyasyon.Ro
ws[i][10]); // Yayılan güneş ışınımı
                double Theta = Ch14.getThetaHor(Beta); // Geliş açısı
                double Etb = Ch14.getEtb(Eb, Theta); // Yüzeye gelen direk
t güneş ışınımı
                double Etd = Ch14.getEtd(Ed, Theta, 0); // Yüzeye yayılan
güneş ışınımı
                double Etr = Ch14.getEtrHor; // Yerden yüzeye yansıyan gün
eş ışınımı (Yatay yüzeyler için 0)
                double Et = Ch14.GetEt(Etb, Etd, Etr); // Toplam güneş ışı
nımı
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Cephe_Cati.Rows[i][2] = Theta;
                this.dataDataSet.Cephe_Cati.Rows[i][3] = Etb;
                this.dataDataSet.Cephe_Cati.Rows[i][4] = Etd;
                this.dataDataSet.Cephe_Cati.Rows[i][5] = Etr;
                this.dataDataSet.Cephe_Cati.Rows[i][6] = Et;
                //
            }
            this.cephe_CatiTableAdapter.Update(this.dataDataSet.Cephe_Cati
); // Veritabanını günceller.
        }
        void Q_hesapla() // Çatıdan saatlik toplam soğutma yükünü hesaplar
.
        {
            // Veritabanından verileri çeker.
            this.catiTableAdapter.Fill(this.dataDataSet.Cati);
            this.qtTableAdapter.Fill(this.dataDataSet.Qt);
            //
            for (int i = 0; i < 24; i++) // Saatlik eşdeğer sıcaklık ve ıs
ı girişini hesaplar.
            {
                double t_dis = Convert.ToDouble(this.dataDataSet.Radyasyon
.Rows[i][3]); // Dış ortam sıcaklığı
                double Et = Convert.ToDouble(this.dataDataSet.Cephe_Cati.R
ows[i][6]); // Toplam güneş ışınımı
                double te = Ch18.get_teHor(t_dis, Et, alphah0); // Eşdeğer
 sıcaklık
                double qi = Ch18.get_qi(Ucati, alan, te, t_ic); // Isı gir
işi
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Cati.Rows[i][2] = t_dis;
                this.dataDataSet.Cati.Rows[i][3] = te;
```

```
                this.dataDataSet.Cati.Rows[i][4] = t_ic;
                this.dataDataSet.Cati.Rows[i][5] = qi;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için iletim zaman ser
ilerini uygular.
            {
                double qcond = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double c = Convert.ToDouble(this.dataDataSet.Cati_CTS.
Rows[j][cati_id]) / 100; // İletim zaman faktörü
                    double qi = Convert.ToDouble(this.dataDataSet.Cati.Row
s[saat][5]); // Isı girişi
                    qcond += c * qi; // İletim ısı kazancı
                    saat--;
                }
                double qconv = 0.4 * qcond; // Taşınımsal ısı kazancı
                double qrad = 0.6 * qcond; // Işınımsal ısı kazancı
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Cati.Rows[i][6] = Convert.ToDouble(this.d
ataDataSet.Cati_CTS.Rows[i][cati_id]);
                this.dataDataSet.Cati.Rows[i][7] = qcond;
                this.dataDataSet.Cati.Rows[i][8] = qconv;
                this.dataDataSet.Cati.Rows[i][9] = qrad;
                //
            }
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qr = 0; int saat = i;
                for(int j=0;j<24;j++)
                {
                    if (saat == -1) saat += 24;
                    double r = Convert.ToDouble(this.dataDataSet.RTS_Nonso
lar.Rows[j][rts_id]);
                    double qrad = Convert.ToDouble(this.dataDataSet.Cati.R
ows[saat][9]);
                    Qr += r * qrad; // Işınımsal soğutma yükü
                    saat--;
                }
                double qconv = Convert.ToDouble(this.dataDataSet.Cati.Rows
[i][8]); // Taşınımsal soğutma yükü
                double Qt = qconv + Qr; // Toplam saatlik soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.Cati.Rows[i][10] = Convert.ToDouble(this.
dataDataSet.RTS_Nonsolar.Rows[i][rts_id]);
                this.dataDataSet.Cati.Rows[i][11] = Qr;
                this.dataDataSet.Cati.Rows[i][12] = Qt;
                this.dataDataSet.Qt.Rows[i][7] = Qt;
                //
            }
            // Veritabanını günceller.
            this.catiTableAdapter.Update(this.dataDataSet.Cati);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
```

```
        private void Cati_Load(object sender, EventArgs e)
        {
            // Veritabanından verileri çeker.
            this.cati_AnaTableAdapter.Fill(this.dataDataSet.Cati_Ana);
            this.cati_AltTableAdapter.Fill(this.dataDataSet.Cati_Alt);
            this.cati_CTSTableAdapter.Fill(this.dataDataSet.Cati_CTS);
            this.cati_YapiTableAdapter.Fill(this.dataDataSet.Cati_Yapi);
            this.rTS_NonsolarTableAdapter.Fill(this.dataDataSet.RTS_Nonsol
ar);
            //
        }
        // En ve boy girişinin yapıldığı metin kutularında sadece sayı, vi
rgül, del ve backspace tuşlarını aktif eder.
        private void en_textBox_KeyPress(object sender, KeyPressEventArgs
e)
        {
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)44 && e.KeyChar != (char)127;
        }
        //
        private void hesapla_button_Click(object sender, EventArgs e)
        {
            veri_cek();
            alan_hesapla();
            radyasyon_hesapla();
            Q_hesapla();
            this.AutoScroll = true; // Sayfa kaydırmayı aktif eder.
            cati_tabControl.Visible = true; // Hesaplanan değerleri ekrand
a gösterir.
            kontrol = true; // Hesapla butonu tıklama kontrolü
        }
    }
}
```

EK C.11 Aydınlatma Ekranı Kaynak Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Aydınlatma Ekranı Kaynak Kodları
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BuharlasmaliSogutucuHesapProgrami
{
    public partial class Aydinlatma : UserControl
    {
        public Aydinlatma()
        {
            InitializeComponent();
        }
        // Değişken tanımları
        int rts_id, baslangic, bitis;
        double en, boy, alan, LPD, conv, rad, qi, qconv, qrad;
        public bool kontrol = false; // Hesapla butonu tıklama kontrol değ
işkeni
        //
        void veri_cek() // Metin kutuları ve veri tabanından verileri çeke
r.
        {
            this.girisTableAdapter.Fill(this.dataDataSet.Giris);
            rts_id = Convert.ToInt16(this.dataDataSet.Giris.Rows[0][6]);
            en = Convert.ToDouble(en_textBox.Text);
            boy = Convert.ToDouble(boy_textBox.Text);
            LPD = Convert.ToDouble(LPD_label.Text); // Aydınlatma güç yoğu
nluğu
            conv = Convert.ToDouble(conv_label.Text); // Taşınım yüzdesi
            rad = Convert.ToDouble(rad_label.Text); // Işınım yüzdesi
            baslangic = Convert.ToInt16(baslangic_comboBox.SelectedIndex)
- 1; // Aydonlatma başlangıç zamanı
            bitis = Convert.ToInt16(bitis_comboBox.SelectedIndex) -
 1; // Aydınlatma bitiş zamanı
        }
        void alan_hesapla() // Bina kullanım alanını hesaplar.
        {
            alan = en * boy;
            alan_label.Text = string.Format("Kullanım Alanı: {0:0.##} m²",
 alan); // Hesaplanan değeri ekrana yazdırır.
        }
        void Q_hesapla() // Aydınlatmadan kaynaklı saatlik toplam soğutma
yükünü hesaplar.
        {
```

```
            qi = LPD * alan; // Isı kazancı
            qconv = conv * qi; // Taşınımsal ısı kazancı
            qrad = rad * qi; // Işınımsal ısı kazancı
            // Veritabanından verileri çeker.
            aydinlatmaTableAdapter.Fill(this.dataDataSet.Aydinlatma);
            qtTableAdapter.Fill(this.dataDataSet.Qt);
            //
            if (baslangic == -1) baslangic += 24;
            if (bitis == -1) bitis += 24;
            for (int i = 0; i < 24; i++)
            {
                this.dataDataSet.Aydinlatma.Rows[i][2] = 0;
                this.dataDataSet.Aydinlatma.Rows[i][3] = 0;
                this.dataDataSet.Aydinlatma.Rows[i][4] = 0;
            }
            // Hesaplanan değerleri veritabanına kaydeder.
            if (baslangic < bitis)
            {
                for (int i = baslangic; i < bitis; i++)
                {
                    this.dataDataSet.Aydinlatma.Rows[i][2] = qi;
                    this.dataDataSet.Aydinlatma.Rows[i][3] = qconv;
                    this.dataDataSet.Aydinlatma.Rows[i][4] = qrad;
                }
            }
            else
            {
                for (int i = baslangic; i < 24; i++)
                {
                    this.dataDataSet.Aydinlatma.Rows[i][2] = qi;
                    this.dataDataSet.Aydinlatma.Rows[i][3] = qconv;
                    this.dataDataSet.Aydinlatma.Rows[i][4] = qrad;
                }
                for (int i = 0; i < bitis; i++)
                {
                    this.dataDataSet.Aydinlatma.Rows[i][2] = qi;
                    this.dataDataSet.Aydinlatma.Rows[i][3] = qconv;
                    this.dataDataSet.Aydinlatma.Rows[i][4] = qrad;
                }
            }
            //
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r = Convert.ToDouble(this.dataDataSet.RTS_Nonso
lar.Rows[j][rts_id]) / 100;
                    double q_rad = Convert.ToDouble(this.dataDataSet.Aydin
latma.Rows[saat][4]);
                    Qr += r * q_rad; // Işınımsal soğutma yükü
                    saat--;
                }
                double q_conv = Convert.ToDouble(this.dataDataSet.Aydinlat
ma.Rows[i][3]); // Taşınımsal soğutma yükü
                double Qt = q_conv + Qr; // Toplam soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
```

```
                this.dataDataSet.Aydinlatma.Rows[i][5] = Convert.ToDouble(
this.dataDataSet.RTS_Nonsolar.Rows[i][rts_id]);
                this.dataDataSet.Aydinlatma.Rows[i][6] = Qr;
                this.dataDataSet.Aydinlatma.Rows[i][7] = Qt;
                this.dataDataSet.Qt.Rows[i][8] = Qt;
                //
            }
            // Veritabanını günceller.
            this.aydinlatmaTableAdapter.Update(this.dataDataSet.Aydinlatma
);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        private void Aydinlatma_Load(object sender, EventArgs e)
        {
            // Veritabanından verileri çeker.
            this.rTS_NonsolarTableAdapter.Fill(this.dataDataSet.RTS_Nonsol
ar);
            this.lPDTableAdapter.Fill(this.dataDataSet.LPD);
            this.aydinlatma_KatsayiTableAdapter.Fill(this.dataDataSet.Aydi
nlatma_Katsayi);
            //
            // Başlangıç ve bitiş saati kutularına ilk değer atar.
            baslangic_comboBox.SelectedIndex = 0;
            bitis_comboBox.SelectedIndex = 0;
            //
        }
        // En ve boy girişinin yapıldığı metin kutularında sadece sayı, vi
rgül, del ve backspace tuşlarını aktif eder.
        private void en_textBox_KeyPress(object sender, KeyPressEventArgs
e)
        {
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)44 && e.KeyChar != (char)127;
        }
        //
        private void hesapla_button_Click(object sender, EventArgs e)
        {
            // Saat aralığının doğru girilip girilmediğini kontrol eder.
            if (baslangic_comboBox.SelectedIndex == bitis_comboBox.Selecte
dIndex)
            {
                MessageBox.Show("Lütfen doğru bir saat aralığı giriniz.",
"Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                kontrol = false;
            }
            //
            else
            {
                veri_cek();
                alan_hesapla();
                Q_hesapla();
                this.AutoScroll = true; // Sayfa kaydırmayı aktif eder.
                Q_groupBox.Visible = true; // Hesaplanan değerleri ekranda
 gösterir.
                kontrol = true; // Hesapla butonu tıklama kontrolü
            }
        }
    } }
```

EK C.12 İnsanlar Ekranı Kaynak Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// İnsanlar Ekranı Kaynak Kodları
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BuharlasmaliSogutucuHesapProgrami
{
    public partial class Insanlar : UserControl
    {
        public Insanlar()
        {
            InitializeComponent();
        }
        // Değişken tanımları
        int rts_id, adet, baslangic, bitis;
        double q, qi, qconv, qrad;
        public bool kontrol = false; // Hesapla butonu tıklama kontrol değ
işkeni
        //
        void veri_cek() // Metin kutuları ve veri tabanından verileri çeke
r.
        {
            this.girisTableAdapter.Fill(this.dataDataSet.Giris);
            rts_id = Convert.ToInt16(this.dataDataSet.Giris.Rows[0][6]);
            adet = Convert.ToInt16(adet_textBox.Text);
            q = Convert.ToDouble(q_label.Text); // İnsan başına duyulur ıs
ı kazancı
            baslangic = Convert.ToInt16(baslangic_comboBox.SelectedIndex)
- 1; // insanların faaliyet başlangıç zamanı
            bitis = Convert.ToInt16(bitis_comboBox.SelectedIndex) -
 1; // İnsanların faaliyet bitiş zamanı
        }
        void Q_hesapla() // İnsanlardan kaynaklanan saatlik soğutma yükünü
 hesaplar.
        {
            // Veritabanından verileri çeker.
            İnsanlarTableAdapter.Fill(this.dataDataSet.İnsanlar);
            qtTableAdapter.Fill(this.dataDataSet.Qt);
            //
            qi = adet * q; // Isı kazancı
            qconv = 0.4 * qi; // Taşınımsal ısı kazancı
            qrad = 0.6 * qi; // Işınımsal ısı kazancı
            if (baslangic == -1) baslangic += 24;
            if (bitis == -1) bitis += 24;
```

```
            for (int i = 0; i < 24; i++)
            {
                this.dataDataSet.İnsanlar.Rows[i][2] = 0;
                this.dataDataSet.İnsanlar.Rows[i][3] = 0;
                this.dataDataSet.İnsanlar.Rows[i][4] = 0;
            }
            // Hesaplanan değerleri veritabanına kaydeder.
            if (baslangic < bitis)
            {
                for (int i = baslangic; i < bitis; i++)
                {
                    this.dataDataSet.İnsanlar.Rows[i][2] = qi;
                    this.dataDataSet.İnsanlar.Rows[i][3] = qconv;
                    this.dataDataSet.İnsanlar.Rows[i][4] = qrad;
                }
            }
            else
            {
                for (int i = baslangic; i < 24; i++)
                {
                    this.dataDataSet.İnsanlar.Rows[i][2] = qi;
                    this.dataDataSet.İnsanlar.Rows[i][3] = qconv;
                    this.dataDataSet.İnsanlar.Rows[i][4] = qrad;
                }
                for (int i = 0; i < bitis; i++)
                {
                    this.dataDataSet.İnsanlar.Rows[i][2] = qi;
                    this.dataDataSet.İnsanlar.Rows[i][3] = qconv;
                    this.dataDataSet.İnsanlar.Rows[i][4] = qrad;
                }
            }
            //
            for (int i = 0; i < 24; i++) // Her saat için ışınım zaman ser
ilerini uygular.
            {
                double Qr = 0; int saat = i;
                for (int j = 0; j < 24; j++)
                {
                    if (saat == -1) saat += 24;
                    double r = Convert.ToDouble(this.dataDataSet.RTS_Nonso
lar.Rows[j][rts_id]) / 100;
                    double q_rad = Convert.ToDouble(this.dataDataSet.İnsan
lar.Rows[saat][4]);
                    Qr += r * q_rad; // Işınımsal soğutma yükü
                    saat--;
                }
                double q_conv = Convert.ToDouble(this.dataDataSet.İnsanlar
.Rows[i][3]); // Taşınımsal soğutma yükü
                double Qt = q_conv + Qr; // Toplam soğutma yükü
                // Hesaplanan değerleri veritabanına kaydeder.
                this.dataDataSet.İnsanlar.Rows[i][5] = Convert.ToDouble(th
is.dataDataSet.RTS_Nonsolar.Rows[i][rts_id]);
                this.dataDataSet.İnsanlar.Rows[i][6] = Qr;
                this.dataDataSet.İnsanlar.Rows[i][7] = Qt;
                this.dataDataSet.Qt.Rows[i][9] = Qt;
                //
            }
            // Veritabanını günceller.
            this.İnsanlarTableAdapter.Update(this.dataDataSet.İnsanlar);
```

```
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        private void Insanlar_Load(object sender, EventArgs e)
        {
            // Veritabanından verileri çeker.
            this.rTS_NonsolarTableAdapter.Fill(this.dataDataSet.RTS_Nonsol
ar);
            this.İnsanlar_FaaliyetTableAdapter.Fill(this.dataDataSet.İnsan
lar_Faaliyet);
            //
            // Başlangıç ve bitiş saati kutularına ilk değer atar.
            baslangic_comboBox.SelectedIndex = 0;
            bitis_comboBox.SelectedIndex = 0;
            //
        }
        // Adet girişinin yapıldığı metin kutusunda sadece sayı, del ve ba
ckspace tuşlarını aktif eder.
        private void adet_textBox_KeyPress(object sender, KeyPressEventArg
s e)
        {
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)127;
        }
        //
        private void hesapla_button_Click(object sender, EventArgs e)
        {
            // Saat aralığının doğru girilip girilmediğini kontrol eder.
            if (baslangic_comboBox.SelectedIndex == bitis_comboBox.Selecte
dIndex)
            {
                MessageBox.Show("Lütfen doğru bir saat aralığı giriniz.",
"Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                kontrol = false;
            }
            //
            else
            {
                veri_cek();
                Q_hesapla();
                this.AutoScroll = true; // Sayfa kaydırmayı aktif eder.
                Q_groupBox.Visible = true; // Hesaplanan değerleri ekranda
 gösterir.
                kontrol = true; // Hesapla butonu tıklama kontrolü
            }
        }
    }
}
```

EK C.13 Cihazlar Ekranı Kaynak Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Cihazlar Ekranı Kaynak Kodları
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BuharlasmaliSogutucuHesapProgrami
{
    public partial class Cihazlar : UserControl
    {
        public Cihazlar()
        {
            InitializeComponent();
        }
        fonksiyon.Bolum18 Ch18 = new fonksiyon.Bolum18(); // Fonksiyon dos
yasını bağlar.
        // Değiken tanımları
        int rts_id, adet, baslangic, bitis;
        double p, verim, qi, qconv, qrad;
        public bool kontrol = false; // Hesapla butonu tıklama kontrol değ
işkeni
        //
        void veri_cek() // Metin kutuları ve veri tabanından verileri çeke
r.
        {
            girisTableAdapter.Fill(this.dataDataSet.Giris);
            rts_id = Convert.ToInt16(this.dataDataSet.Giris.Rows[0][6]);
            // Cihaz adedi girilmezse değerleri sıfır olarak belirler.
            if (adet_textBox.Text == "")
            {
                adet = 0; p = 0; verim = 0;
            }
            //
            else
            {
                adet = Convert.ToInt16(adet_textBox.Text);
                p = Convert.ToDouble(guc_textBox.Text) * 1000; // Birim ma
kine gücü [kW]
                verim = Convert.ToDouble(verim_textBox.Text) / 100; // Mak
ine verimi
                baslangic = Convert.ToInt16(baslangic_comboBox.SelectedInd
ex) - 1; // Makinelerin çalışma başlangıç saati
                bitis = Convert.ToInt16(bitis_comboBox.SelectedIndex) -
 1; // Makinelerin çalışma bitiş saati
            }
```

```
        }
        void Q_hesapla() // Cihazlardan kaynaklanan saatlik soğutma yükünü
 hesaplar.
        {
            // Veritabanından verileri çeker.
            this.cihazTableAdapter.Fill(this.dataDataSet.Cihaz);
            this.qtTableAdapter.Fill(this.dataDataSet.Qt);
            //
            // Cihaz adedi sıfır ise soğutma yükünü sıfır olarak belirler.
            if (adet == 0)
            {
                for (int i = 0; i < 24; i++)
                {
                    this.dataDataSet.Cihaz.Rows[i][7] = 0;
                    this.dataDataSet.Qt.Rows[i][10] = 0;
                    qi_label.Text = string.Format("Isı Kazancı: {0:0.####}
 W", 0);
                }
            }
            //
            else
            {
                // Makinelerden kaynaklanan ısı kazancını hesaplar.
                if (p1_radioButton.Checked) qi = adet * Ch18.get_qem1(p, v
erim);
                else if (p2_radioButton.Checked) qi = adet * Ch18.get_qem2
(p);
                else if (p3_radioButton.Checked) qi = adet * Ch18.get_qem3
(p, verim);
                //
                qi_label.Text = string.Format("Isı Kazancı: {0:0.####} W",
 qi); // Hesaplanan ısı kazancını ekrana yazdırır.
                qconv = 0.5 * qi; // Taşınımsal ısı kazancı
                qrad = 0.5 * qi; // Işınımsal ısı kazancı
                if (baslangic == -1) baslangic += 24;
                if (bitis == -1) bitis += 24;
                for (int i = 0; i < 24; i++)
                {
                    this.dataDataSet.Cihaz.Rows[i][2] = 0;
                    this.dataDataSet.Cihaz.Rows[i][3] = 0;
                    this.dataDataSet.Cihaz.Rows[i][4] = 0;
                }
                // Hesaplanan değerleri veritabanına kaydeder.
                if (baslangic < bitis)
                {
                    for (int i = baslangic; i < bitis; i++)
                    {
                        this.dataDataSet.Cihaz.Rows[i][2] = qi;
                        this.dataDataSet.Cihaz.Rows[i][3] = qconv;
                        this.dataDataSet.Cihaz.Rows[i][4] = qrad;
                    }
                }
                else
                {
                    for (int i = baslangic; i < 24; i++)
                    {
                        this.dataDataSet.Cihaz.Rows[i][2] = qi;
                        this.dataDataSet.Cihaz.Rows[i][3] = qconv;
                        this.dataDataSet.Cihaz.Rows[i][4] = qrad;
```

```
                    }
                    for (int i = 0; i < bitis; i++)
                    {
                        this.dataDataSet.Cihaz.Rows[i][2] = qi;
                        this.dataDataSet.Cihaz.Rows[i][3] = qconv;
                        this.dataDataSet.Cihaz.Rows[i][4] = qrad;
                    }
                }
                //
                for (int i = 0; i < 24; i++) // Her saat için ışınım zaman
 serilerini uygular.
                {
                    double Qr = 0; int saat = i;
                    for (int j = 0; j < 24; j++)
                    {
                        if (saat == -1) saat += 24;
                        double r = Convert.ToDouble(this.dataDataSet.RTS_N
onsolar.Rows[j][rts_id]) / 100;
                        double q_rad = Convert.ToDouble(this.dataDataSet.C
ihaz.Rows[saat][4]);
                        Qr += r * q_rad; // Işınımsal soğutma yükü
                        saat--;
                    }
                    double q_conv = Convert.ToDouble(this.dataDataSet.Ciha
z.Rows[i][4]); // Taşınımsal soğutma yükü
                    double Qt = q_conv + Qr; // Toplam soğutma yükü
                    // Hesaplanan değerleri veritabanına kaydeder.
                    this.dataDataSet.Cihaz.Rows[i][5] = Convert.ToDouble(t
his.dataDataSet.RTS_Nonsolar.Rows[i][rts_id]);
                    this.dataDataSet.Cihaz.Rows[i][6] = Qr;
                    this.dataDataSet.Cihaz.Rows[i][7] = Qt;
                    this.dataDataSet.Qt.Rows[i][10] = Qt;
                    //
                }
            }
            // Veritabanını günceller.
            this.cihazTableAdapter.Update(this.dataDataSet.Cihaz);
            this.qtTableAdapter.Update(this.dataDataSet.Qt);
            //
        }
        private void Cihazlar_Load(object sender, EventArgs e)
        {
            this.rTS_NonsolarTableAdapter.Fill(this.dataDataSet.RTS_Nonsol
ar); // Veritabanından verileri çeker.
            // Başlangıç ve bitiş saati kutularına ilk değer atar.
            baslangic_comboBox.SelectedIndex = 0;
            bitis_comboBox.SelectedIndex = 0;
            //
        }
        // Adet girişinin yapıldığı metin kutusunda sadece sayı, del ve ba
ckspace tuşlarını aktif eder.
        private void adet_textBox_KeyPress(object sender, KeyPressEventArg
s e)
        {
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)127;
        }
        //
```

```
        // Güç ve verim girişinin yapıldığı metin kutularında sadece sayı,
 del ve backspace tuşlarını aktif eder.
        private void guc_textBox_KeyPress(object sender, KeyPressEventArgs
 e)
        {
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)44 && e.KeyChar != (char)127;
        }
        //
        private void hesapla_button_Click(object sender, EventArgs e)
        {
            veri_cek();
            if (adet != 0 && (verim < 0 || verim > 1)) // Verim değerinin
0-%100 arasında girilip girilmediğini kontrol eder.
            {
                MessageBox.Show("Verim %0 ile %100 arasında olmalıdır.", "
Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                kontrol = false;
            }
            // Saat aralığının doğru girilip girilmediğini kontrol eder.
            else if (adet != 0 && baslangic_comboBox.SelectedIndex == biti
s_comboBox.SelectedIndex)
            {
                MessageBox.Show("Lütfen doğru bir saat aralığı giriniz.",
"Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                kontrol = false;
            }
            //
            else
            {
                Q_hesapla();
                this.AutoScroll = true; // Sayfa kaydırmayı aktif eder.
                Q_groupBox.Visible = true; // Hesaplanan değerleri ekranda
 gösterir.
                kontrol = true; // Hesapla butonu tıklama kontrolü
            }
        }
    }
}
```

EK C.14 Toplam Soğutma Yükü Ekranı

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Toplam Soğutma Yükü Ekranı Kaynak Kodları
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BuharlasmaliSogutucuHesapProgrami
{
    public partial class Qtoplam : UserControl
    {
        public Qtoplam()
        {
            InitializeComponent();
        }
        // Değişken tanımları
        double[] Qt = new double[24];
        string[] saat = new string[24];
        double Qmax;
        string pik_saat;
        //
        private void Qtoplam_VisibleChanged(object sender, EventArgs e)
        {
            if (this.Visible == true)
            {
                // Veritabanından verileri çeker.
                this.qtTableAdapter.Fill(this.dataDataSet.Qt);
                this.girisTableAdapter.Fill(this.dataDataSet.Giris);
                //
                // Toplam soğutma yükünü hesaplar.
                for (int i = 0; i < 24; i++)
                {
                    double cephe1 = Convert.ToDouble(this.dataDataSet.Qt.R
ows[i][3]);
                    double cephe2 = Convert.ToDouble(this.dataDataSet.Qt.R
ows[i][4]);
                    double cephe3 = Convert.ToDouble(this.dataDataSet.Qt.R
ows[i][5]);
                    double cephe4 = Convert.ToDouble(this.dataDataSet.Qt.R
ows[i][6]);
                    double cati = Convert.ToDouble(this.dataDataSet.Qt.Row
s[i][7]);
                    double aydinlatma = Convert.ToDouble(this.dataDataSet.
Qt.Rows[i][8]);
                    double insanlar = Convert.ToDouble(this.dataDataSet.Qt
.Rows[i][9]);
```

```
                    double cihazlar = Convert.ToDouble(this.dataDataSet.Qt
.Rows[i][10]);
                    double Qtoplam = cephe1 + cephe2 + cephe3 + cephe4 + c
ati + aydinlatma + insanlar + cihazlar;
                    this.dataDataSet.Qt.Rows[i][11] = Qtoplam; // Toplam s
oğutma yükünü veritabanına kaydeder.
                    Qt[i] = Qtoplam;
                    saat[i] = this.dataDataSet.Qt[i][1].ToString();
                }
                this.qtTableAdapter.Update(this.dataDataSet.Qt); // Verita
banını günceller.
                //
                Qmax = Qt.Max(); // Pik yükü belirler.
                if (Qmax <= 0) Qmax = 0; // Soğutma yükü negatif çıkarsa s
ıfır olarak alır.
                pik_saat = saat[Qt.ToList().IndexOf(Qmax)]; // Pik yükün o
luştuğu saati belirler.
                this.dataDataSet.Giris.Rows[0][5] = Qmax; // Pik soğutma y
ükünü veritabanına yazdırır.
                this.girisTableAdapter.Update(this.dataDataSet.Giris); //
Veritabanını günceller.
                // Soğutma yükü negatif çıkarsa uyarı verir.
                if (Qmax <= 0) MessageBox.Show("Bu bina için soğutmaya ger
ek yoktur.", "Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                //
                // Değerleri ekrana yazdırır.
                else
                {
                    pik_saat_label.Text = pik_saat.ToString();
                    Qmax_label.Text = string.Format("{0:0.####} W", Qmax);
                }
                //
            }
        }
    }
}
```

EK C.15 Soğutucu Kapasitesi Ekranı Kaynak Kodları

```
// Buharlaşmalı Soğutucu Hesap Programı
// Oğuz ÇALIŞKAN
// Yüksek Lisans Tezi
// Denizli, 2015
// Soğutucu Kapasite Ekranı Kaynak Kodları
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BuharlasmaliSogutucuHesapProgrami
{
    public partial class Kapasite : UserControl
    {
        public Kapasite()
        {
            InitializeComponent();
        }
        fonksiyon.Bolum1 Ch1 = new fonksiyon.Bolum1(); // Fonksiyon dosyas
ını bağlar.
        // Değişken tanımları
        double z, t_giris, RelHum_giris, Pdoyma_giris, Pv_giris, t_ciy_gir
is, w_giris, v_giris;
        double t_ic, t_cikis, RelHum_cikis, Pdoyma_cikis, Pv_cikis, t_ciy_
cikis, w_cikis, v_cikis;
        double Qt, Patm, h, t_yas, density_ratio;
        double cihaz_hava, ped_alan, verim;
        double v_hava, cihaz_su, toplam_hava, toplam_su, adet, hava_hizi;
        bool tamam_kontrol = false; // Tamam butonu tıklama kontrol değişk
eni
        //
        void veri_cek() // Metin kutuları ve veritabanından verileri çeker
.
        {
            t_giris = Convert.ToDouble(this.dataDataSet.Giris.Rows[0][1]);
            t_ic = Convert.ToDouble(this.dataDataSet.Giris.Rows[0][2]);
            z = Convert.ToDouble(this.dataDataSet.Giris.Rows[0][3]);
            RelHum_giris = Convert.ToDouble(this.dataDataSet.Giris.Rows[0]
[4]);
            Qt = Convert.ToDouble(this.dataDataSet.Giris.Rows[0][5]);
            cihaz_hava = Convert.ToDouble(hava_textBox.Text);
            ped_alan = Convert.ToDouble(ped_alan_textBox.Text);
            verim = Convert.ToDouble(verim_textBox.Text) / 100;
        }
        void veri_cek2() // Metin kutusundan veri çeker.
        {
            density_ratio = Convert.ToDouble(density_ratio_textBox.Text);
        }
```

```
        void giris_psikrometrik_hesap() // Soğutucu girişindeki havanın ps
ikrometrik hesaplamalarını yapar.
        {
            Patm = Ch1.getPatm(z); // Atmosfer basıncı
            Pdoyma_giris = Ch1.getPsat(t_giris); // Soğutucu girişi doyma
basıncı
            Pv_giris = Ch1.getPvap(Pdoyma_giris, RelHum_giris); // Soğutuc
u girişi kısmi buhar basıncı
            w_giris = Ch1.get_w(Patm, Pv_giris); // Soğutucu girişi özgül
nem
            h = Ch1.get_h(t_giris, w_giris); // Özgül entalpi
            t_ciy_giris = Ch1.get_tdew(Pv_giris); // Soğutucu girişi çiy n
oktası sıcaklığı
            v_giris = Ch1.get_v(t_giris, Patm, w_giris); // Soğutucu giriş
i özgül hacim
            t_yas = Ch1.get_twet(t_giris, t_ciy_giris, Patm, w_giris); //
Yaş termometre sıcaklığı
        }
        void t_cikis_hesapla() // Soğutucu çıkış sıcaklığını hesaplar.
        {
            t_cikis = t_giris - (verim * (t_giris - t_yas));
        }
        void cikis_psikrometrik_hesap() // Soğutucu çıkışındaki havanın ps
ikrometrik hesaplamalarını yapar.
        {
            Pdoyma_cikis = Ch1.getPsat(t_cikis); // Soğutucu çıkışı doyma
basıncı
            w_cikis = Ch1.get_w_from_h(t_cikis, h); // Soğutucu çıkışı özg
ül nem
            Pv_cikis = Ch1.getPvap_from_w(Patm, w_cikis); // Soğutucu çıkı
şı kısmi buhar basıncı
            t_ciy_cikis = Ch1.get_tdew(Pv_cikis); // Soğutucu çıkışı çiy n
oktası sıcaklığı
            v_cikis = Ch1.get_v(t_cikis, Patm, w_cikis); // Soğutucu çıkış
ı özgül hacim
            RelHum_cikis = Ch1.getRelHum(Pv_cikis, Pdoyma_cikis); // Soğut
ucu çıkışı bağıl nem
        }
        void kapasite_hesapla() // Soğutucu cihaz kapasitesini hesaplar.
        {
            hava_hizi = cihaz_hava / (3600 * ped_alan); // Cihaz hava çıkı
ş hızını hesaplar.
            toplam_hava = 2.9281 * Qt / ((t_ic -
 t_cikis) * density_ratio); // Gereken toplam hava debisini hesaplar.
            v_hava = (v_giris + v_cikis) / 2; // Nemli havanın özgül hacmi
            cihaz_su = cihaz_hava * (w_cikis -
 w_giris) * 0.001003 / v_hava; // Cihaz başına beslenmesi gereken su debis
ini hesaplar.
            adet = Math.Ceiling(toplam_hava / cihaz_hava); // Gereken topl
am cihaz sayısını hesaplar.
            toplam_su = adet * cihaz_su; // Cihazlara beslenmesi gereken t
oplam su debisini hesaplar.
        }
        void psikrometrik_veri_yaz() // Hesaplanan psikrometrik verileri e
krana yazdırır.
        {
            t_giris_label.Text = string.Format("{0:0.####} °C", t_giris);
            RelHum_giris_label.Text = string.Format("% {0:0.####}", RelHum
_giris * 100);
```

```
            Patm_giris_label.Text = string.Format("{0:0.####} kPa", Patm /
 1000);
            Pdoy_giris_label.Text = string.Format("{0:0.####} kPa", Pdoyma
_giris / 1000);
            Pv_giris_label.Text = string.Format("{0:0.####} kPa", Pv_giris
 / 1000);
            h_giris_label.Text = string.Format("{0:0.####} kJ/kg", h);
            w_giris_label.Text = string.Format("{0:0.####} kg H2O/kg kuru
hava", w_giris);
            t_ciy_giris_label.Text = string.Format("{0:0.####} °C", t_ciy_
giris);
            vgiris_label.Text = string.Format("{0:0.####} m³/kg", v_giris)
;
            t_yas_giris_label.Text = string.Format("{0:0.####} °C", t_yas)
;
            t_cikis_label.Text = string.Format("{0:0.####} °C", t_cikis);
            RelHum_cikis_label.Text = string.Format("% {0:0.####}", RelHum
_cikis * 100);
            Patm_cikis_label.Text = string.Format("{0:0.####} kPa", Patm /
 1000);
            Pdoy_cikis_label.Text = string.Format("{0:0.####} kPa", Pdoyma
_cikis / 1000);
            Pv_cikis_label.Text = string.Format("{0:0.####} kPa", Pv_cikis
 / 1000);
            h_cikis_label.Text = string.Format("{0:0.####} kJ/kg", h);
            w_cikis_label.Text = string.Format("{0:0.####} kg H2O/kg kuru
hava", w_cikis);
            t_ciy_cikis_label.Text = string.Format("{0:0.####} °C", t_ciy_
cikis);
            vcikis_label.Text = string.Format("{0:0.####} m³/kg", v_cikis)
;
            t_yas_cikis_label.Text = string.Format("{0:0.####} °C", t_yas)
;
        }
        void kapasite_veri_yaz() // Hesaplanan kapasite verilerini ekrana
yazdırır.
        {
            ped_alan_label.Text = string.Format("{0:0.####} m²", ped_alan)
;
            verim_label.Text = string.Format("% {0:0.####}", verim * 100);
            hava_hizi_label.Text = string.Format("{0:0.####} m/s", hava_hi
zi);
            cihaz_hava_label.Text = string.Format("{0:0.##} m³/h", cihaz_h
ava);
            cihaz_su_label.Text = string.Format("{0:0.####} m³/h", cihaz_s
u);
            Qt_label.Text = string.Format("{0:0.####} W", Qt);
            toplam_hava_label.Text = string.Format("{0:0.####} m³/h", topl
am_hava);
            toplam_su_label.Text = string.Format("{0:0.####} m³/h", toplam
_su);
            adet_label.Text = string.Format("{0:0}", adet);
        }
        private void Kapasite_Load(object sender, EventArgs e)
        {
            this.hava_Yogunluk_OraniTableAdapter.Fill(this.dataDataSet.Hav
a_Yogunluk_Orani); // Veritabanından verileri çeker.
        }
```

```
        // Metin kutularına veri girişinde sadece sayı, virgül, del ve bac
kspace tuşlarını aktif eder.
        // Tüm metin kutularına bu kod referans verilmiştir.
        private void density_ratio_textBox_KeyPress(object sender, KeyPres
sEventArgs e)
        {
            e.Handled = !char.IsDigit(e.KeyChar) && e.KeyChar != (char)8 &
& e.KeyChar != (char)44 && e.KeyChar != (char)127;
        }
        private void tamam_button_Click(object sender, EventArgs e)
        {
            this.girisTableAdapter.Fill(this.dataDataSet.Giris); // Verita
banından verileri çeker.
            veri_cek();
            // Doyma veriminin %0-
100 arasında girilip girilmediğini kontrol eder.
            if (verim < 0 || verim > 1)
            {
                MessageBox.Show("Doyma verimi %0 ile %100 arasında olmalıd
ır.", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                tamam_kontrol = false;
            }
            //
            else
            {
                giris_psikrometrik_hesap();
                t_cikis_hesapla();
                cikis_psikrometrik_hesap();
                // Cihaz çıkış sıcaklığı iç ortam sıcaklığından yüksek vey
a çiy noktası sıcaklığından düşük çıktığında hesaplamayı durdurur.
                if (t_cikis > t_ic || t_cikis < t_ciy_cikis)
                {
                    MessageBox.Show("Girilen iç ortam sıcaklığına buharlaş
malı soğutma ile ulaşılamaz.", "Uyarı", MessageBoxButtons.OK, MessageBoxIc
on.Warning);
                    tamam_kontrol = false;
                }
                //
                else
                {
                    psikrometrik_veri_yaz();
                    tamam_kontrol = true;
                }
            }
        }
        private void hesapla_button_Click(object sender, EventArgs e)
        {
            // Tamam butonuna basılıp basılmadığını kontrol eder.
            if (tamam_kontrol == false) MessageBox.Show("Lütfen cihaz bilg
ilerini girip Tamam butonuna basınız.", "Uyarı", MessageBoxButtons.OK, Mes
sageBoxIcon.Warning);
            //
            else
            {
                veri_cek2();
                // Hava yoğunluk oranının 0-
1 arasında girilip girilmediğini kontrol eder.
```

```
                if (density_ratio < 0 || density_ratio > 1) MessageBox.Sho
w("Hava yoğunluk oranı 0 ile 1 arasında olmalıdır.", "Uyarı", MessageBoxBu
ttons.OK, MessageBoxIcon.Warning);
                //
                else
                {
                    kapasite_hesapla();
                    kapasite_veri_yaz();
                }
            }
        }
    }
}
```

EK D Denizli İline Ait İklim Verileri

Tablo D.1: Denizli iline ait 2014 yılı 21 Mayıs-21 Eylül tarihleri arası sıcaklık ve bağıl nem değerleri (Meteoroloji Genel Müdürlüğü'nden özel görüşme ile temin edilmiştir.).

	21 Mayıs	**21 Haziran**	**21 Temmuz**	**21 Ağustos**	**21 Eylül**
Bağıl Nem [%]	37,3	50,8	30,1	25,8	30,2
Yaz Saati	**Sıcaklık [°C]**				
0	21	22,1	27,1	27	19,9
1	19,8	20,7	25,6	26,3	19,8
2	18,1	20,1	25,1	25,3	18,9
3	19,2	18,2	24,1	24,5	16,5
4	18,5	17,5	23,4	23,8	16,7
5	18,2	17,3	22,4	23,5	16,3
6	17,6	17,8	21,7	22,4	15
7	16,7	18,2	22	22	14,1
8	17,4	21,6	24,6	25,7	17,5
9	21,8	21,7	25,4	27,9	20,1
10	22,9	22,3	26,6	28,7	21,1
11	24,1	23,8	28,4	30,5	22,8
12	24,9	25,3	29,3	32,2	24,6
13	26,1	26,2	30,8	33,2	26,6
14	27,4	26,8	32,2	34,9	28,4
15	28,6	27,9	32,9	36,5	29,7
16	28,6	28,8	33,8	36,9	30
17	29,5	29,2	33,9	36,2	29,5
18	27,2	27,9	34,1	34,6	28,5
19	25,9	26,9	33,1	32,8	25,6
20	24,7	25,7	31,2	31,1	23,3
21	23,8	24,5	29,7	29,4	22,1
22	22,8	22,9	29	28,6	21,5
23	21,4	22,6	28,2	27,9	21,2

9. ÖZGEÇMİŞ

Adı Soyadı	:	Oğuz ÇALIŞKAN
Doğum Yeri ve Tarihi	:	İzmir - 21.11.1989
Lisans Üniversite	:	Pamukkale Üniversitesi
Elektronik posta	:	oguzcaliskan@selcuk.edu.tr
İletişim Adresi	:	Selçuk Üniversitesi Mühendislik Fakültesi Makine Mühendisliği Bölümü Aleaddin Keykubat Kampüsü Selçuklu/KONYA