

Wind/Solar Power Miniproject PDR

Group 5

Sept. 21 2004

To: Dr. Teare

Executive Summary

This Preliminary Design Review outlines a study that determines the feasibility of designing a livable house that will be powered exclusively by on-site wind and solar power generation.

The introductory section of the study is comprised of an Executive Summary which outlines the rest of this document, an Introduction which describes the project, and a Background section that lists key information pertaining to the project. The descriptive section of this document consists of a formal Project Description which lists the tasks and assignments necessary for the completion of this project, a deliverables section which explicitly identifies all deliverables and due dates, and a section outlining plans for Test and Evaluation.

Contents

| | | |
|----------|---|----------|
| 1 | Introduction | 4 |
| 2 | Background | 4 |
| 2.1 | Assumptions | 4 |
| 3 | Project Description | 4 |
| 3.1 | Deliverables | 4 |
| 4 | Testing and Evaluation | 5 |
| 5 | Specifications | 5 |
| 5.1 | System Requirements | 5 |
| 5.2 | System Specification | 5 |
| 6 | Results | 6 |
| 7 | Cost Analysis | 6 |
| 8 | Summary | 6 |
| 9 | Appendix: Source Code and Data Files | 7 |
| 9.1 | data_appliances.py | 7 |
| 9.2 | data_constants.py | 8 |
| 9.3 | data_hvac.py | 8 |
| 9.4 | data_power.py | 8 |
| 9.5 | helper.py | 8 |
| 9.6 | main.py | 8 |
| 9.7 | xpath.py | 10 |
| 9.8 | model_hvac.py | 12 |
| 9.9 | resources.xml | 13 |

List of Figures

| | | |
|---|---|----|
| 1 | MS Project Tracking Gantt Chart | 17 |
| 2 | Daily Energy Consumption | 17 |
| 3 | Daily Battery Level | 18 |
| 4 | Daily Wind and Solar Power Production | 18 |
| 5 | Indoor vs. Outdoor Temperature | 19 |
| 6 | Peak Power Consumption | 19 |
| 7 | HVAC Usage | 20 |
| 8 | Temperature Model | 20 |

1 Introduction

This project explores the plausibility of designing a livable house that will be able to exist off of the power grid. The home in question will be powered exclusively by wind and solar energy generation. Another important factor in this project will focus on the heating and cooling needs of the home which would be required to make the house livable.

The system is powered by windmills and solar cells, and features back-up batteries for use during periods of little sunlight and low wind. The project itself contains a test and evaluation phase which determines the effectiveness of the system under a specific weather pattern.

2 Background

The house we are designing will be set in New Mexico. The location of the house determines likely weather patterns, and the depth of the water table.

To complete this project, a few things must be considered. Assumptions concerning the lifestyle of the occupants and the types of appliances will be made. These assumptions will determine the energy used in the house. Thus they will determine the type of wind mill and solar panels we will use to generate energy. Ambient temperature will affect the house temperature. Therefore heating and cooling of the house will be function of the external temperature. Wind and sunshine will provide a means for generating power to the house.

2.1 Assumptions

The daily schedules of the people living in their house and the power consumptions for various appliances are listed in section 9.9.

3 Project Description

Figure 1 shows a schedule of time and personal responsibilities. Our team is composed of three electrical engineers: David Kadjo, Thomas fuller, and David Baird.

David Baird will be primarily working on a computer model of the house energy consumption; This model will be used to test the effectiveness of the house. Thomas Fuller will be searching the market for heating, cooling, solar, and wind power possibilities. He is in charge of finding windmill and solar panel systems to provide the energy needed in the house. David Kadjo will be getting information about weather patterns and testing the entire system.

3.1 Deliverables

The following items will be delivered as a result of this project:

- A computer model: the computer calculates energy usage and house temperature in 1 hour intervals; The computer model provides the temperature and the energy consumption depending of the activities in the house and the weather. The computer model serves as an energy effectiveness test.
- Specification for the electrical power system requirements; Commercial systems exist that are capable of meeting these specifications.
- A cost analysis: We will provide a cost investigation, showing the cost of the off-the-grid house compared to an on-the-grid house.

4 Testing and Evaluation

An important step in designing the house will be to test the system; We need to verify the correctness of our assumptions, and the energy use of the house. The way to test the system is to implement a computer model that simulates the energy consumption in the house. We input data to the computer model and analyze the output. Analysis of the output determines the effectiveness of the system and will help ameliorate the system. The inputs to the computer model is the weather pattern for a period of a month. A pattern of the four different seasons, summer, autumn, winter and spring will be collected. The main elements to take into account for those seasons are the wind, the sun, and temperature. The wind and the sun are the energy providers. Other inputs to the computer model are the energy consumptions of each of the appliances and the amount of water needed for the different activities in the house. Based on the inputs to the computer we get the following outputs: energy shortage, energy remaining in batteries, the amount of water pumped and the temperature inside the house. These output data are collected every hour and analyzed; Our analysis will give the effectiveness of the system. Plots of the different output are included in this Preliminary Design Review.

5 Specifications

5.1 System Requirements

- House require 24.26 kWh per day
- Peak usage is 12 kW
- Peak battery usage required is 8.0 kW

5.2 System Specification

- System is comprised of a 10 kWh solar subsystem and a 20 kWh windmill subsystem
- System can handle a peak value of 15 kW
- Total available battery storage is 13.5 kWh

6 Results

Figure 2 shows how much energy the appliances require every hour, every day. Figure 4 shows how much energy is being produced by solar panels and by wind power. Figure 6 shows the peak amount of power required within 1 hour intervals. Notice that on Sunday when laundry is being done that the power requirement peaks to its highest. Figure 5 shows the indoor and outdoor temperature.

If the system was optimized, then the batteries will oscillate between being fully charged to nearly discharged. Referring to figure 3, the batteries only discharge to nearly half of their capacity. The fact that the battery level remains capped off at its maximum for most of the time indicates that the system is producing more energy than can be used. Thus, it be advisable to cut down on the wind and solar power generation systems.

Referring to figure 7, the air conditioning system remains on for less than 1 percent of the day, or about 1-2 hours. This does not seem reasonable and needs to be analyzed more carefully before implementing this system.

Figure 8 shows the linear circuit model that is used to calculate the temperature of the house.

7 Cost Analysis

The power usage of the entire house is 730 kWh/month. A house receiving the same amount of power from a power company in New Mexico would pay \$51.00 every month. It should be noted that the power system of the house also provides fuel for the heating and cooling systems, hot water heaters, and the electricity for the household appliances. To put this into context, the average family in New Mexico pays approximately \$1,295 for their annual utility bills. The annual bill for the power system would be \$612.00 which is less than half of the annual bill for the average home. Another factor that should be taken into consideration is the initial cost of the power system equipment, which is a total one-time cost of \$160,000.

8 Summary

It is possible to build a house that does not use draw power from the power grid. Using only wind and solar power generation, a house can be fully powered without sacrificing modern conveniences such as comfortable environmental temperatures, and an assortment of appliances. However, such a system requires tremendous initial costs, and should only be considered if money is not a driving factor.

9 Appendix: Source Code and Data Files

9.1 data_appliances.py

```
import xpath
import data_constants
import math

class Appliance:
    def __init__(self, name, rate):
        self.name = name
        self.rate = float(rate)
    def __repr__(self):
        return self.__str__()
    def __str__(self):
        return "%s"/%f' % (self.name, self.rate)

data = xpath.XPathDict(file='../20040920.resources/resources.xml')

unit_to_watt = {
    'watt': 1.,
    'kWh/month': 1000./24/30,
    'kWh/year': 1000./24/365
}

appliances = {}
XPathDict2 = lambda nn: map(lambda n: xpath.XPathDict(doc=n.doc, node=n), nn)
for a in XPathDict2(
    data.nodes('//resources/appliances/appliance') + \
    data.nodes('//resources/periodic-appliances/appliance')):
    # I would have preferred this: data.nodes('*/*appliances/appliance')
    id = a['@id']
    name = a['name']
    rate = 0.
    if a['rate'] is not None:
        rate = a['rate/@value']
        assert(a['rate/@unit'] == 'watt')
    elif a['averate'] is not None:
        rate = float(a['averate/@value']) / float(a['averate/@dutycycle'])
        rate *= unit_to_watt[a['averate/@unit']]
    appliances[id] = Appliance(name, rate)

# Calculate the energy and power requirements for every hour of a week...
# hour 0 = 12am Monday morning
hours = xrange(7 * 24)
weekdays = range(0, 5*24, 24)
saturday = 5*24
sunday = 6*24
peakpower = [0. for h in hours]
energy = [0. for h in hours]
energy2 = {}
for id in appliances: energy2[id] = 0.
m = lambda s: int(s[-2:])
h = lambda s: int(s[:-2])
tohours = lambda s, h=h, m=m: h(s) + m(s)/60.
for a in XPathDict2(data.nodes('//resources/activities/schedule')):
    id = a['@id']
    power = 0.
    for b in XPathDict2(a.nodes('use')):
        ref = b['@ref']
        on = tohours(b['@from'])
        off = tohours(b['@to'])
        onrange = xrange(*map(int, [math.floor(on), math.ceil(off)+1]))
        count = int(b['@count'] or 1)
        power = appliances[ref].rate * count
        if id == 'saturday':
            r = [saturday]
        elif id == 'sunday':
            r = [sunday]
        elif id == 'weekday':
            r = weekdays
        for hh in r:
            for h in onrange:
                peakpower[h+hh] += power
        t1 = on
        while t1 < off:
            h = int(math.floor(t1))
            t2 = h + 1.
            if t2 > off: t2 = off
            e = power * (t2 - t1) * 3600.
            for hh in r:
                energy[h+hh] += e
                energy2[ref] += e
            t1 = t2
del m, h

for a in XPathDict2(\
    data.nodes('//resources/periodic-appliances/appliance')):
```

```

id = a['@id']
p = appliances[id].rate
e = appliances[id].rate * 3600. * float(a['@average/@duty'])
for h in hours:
    peakpower[h] += p
    energy[h] += e

if __name__ == '__main__':
    for h in hours:
        print h, h%24, energy[h], peakpower[h]
    joules = reduce(lambda a, b: a+b, energy)
    kwh = joules/3600./1000.
    print 'Total kWh =', kwh, kwh/7.
    for i in energy2:
        print i, energy2[i]/3600.

```

9.2 data_constants.py

```

hours = range(24 * 30)

# Initial values:
house_temperature = 25. # deg C
battery_level = 0. # joules

```

9.3 data_hvac.py

```

import helper

R_to_ft2_degC_W = 6.14
R30 = 30 * R_to_ft2_degC_W # ft^2 * C / W
R15 = 15 * R_to_ft2_degC_W
wall_SA = 40*8*2 + 50*8*2 + 2000
floor_SA = 2000

theta_walls = R30 / wall_SA # deg C per watt
theta_floor = R30 / floor_SA
Pac_elec = 3000. # watts
SEER_ac = 19.5
Pheater_elec = 3000. # watts
SEER_heater = 17.9
Tground = helper.FtoC(85)
heat_capacity = 350000 # joules per kelvin
Thousemax = helper.FtoC(78)
Thousemin = helper.FtoC(70)

```

9.4 data_power.py

```

wind_min = 7 # mph
wind_max = 25 # mph
wind_cost = 3. # dollars/watt
windmill_count = 1

peak_solar_power_density = 7. * 1000 / 24 # watts/m^2
solar_efficiency = .30
solar_cost = 10. # dollars/watt
# watts = SA * solar_efficiency * peak_surface_solar_power * insolation
solar_area = 100.
solar_power_max = solar_efficiency * peak_solar_power_density * solar_area
battery_capacity = 12 * 225 * 3600 * 5 # V * AH * J/VAH * count

def wind_to_power(mph, min=wind_min, max=wind_max):
    if mph < min: return 0.
    if mph > max: mph = max
    return 2377 - 698.8 * mph + 52.27 * (mph ** 2.)

def insolation_to_power(insolation, maxpower=solar_power_max):
    return insolation/100. * maxpower

if __name__ == '__main__':
    print wind_to_power(wind_min * 1.00001)
    print wind_to_power(wind_max)
    print 'solar-power-max', solar_power_max

```

9.5 helper.py

```

CtoF = lambda C: (9./5.)*C + 32
FtoC = lambda F: (5./9.)*(F - 32)
Wh_per_joule = 1. / 3600.

```

9.6 main.py

```

import data_weather
import data_constants
import data_appliances
import data_hvac

```

```

import model_hvac
import data_power
from helper import CtoF, Wh_per_joule

def main():
    house = model_hvac.House(
        data_hvac.theta_walls,
        data_hvac.theta_floor,
        data_hvac.Pac_elec,
        data_hvac.SEER_ac,
        data_hvac.Pheater_elec,
        data_hvac.SEER_heater,
        data_hvac.Tground,
        data_hvac.heat_capacity,
        data_hvac.Thousemax,
        data_hvac.Thousemin
    )
    house_temp = data_constants.house_temperature
    battery_level = data_constants.battery_level
    total_energy_needed = 0.
    total_energy_shortage = 0.
    datafile = open('data', 'w')
    for hour in data_constants.hours:
        energy_usage = 0.
        peakpower = 0.
        energy_source = 0.
        energy_unused = 0.
        energy_shortage = 0.
        battery_delta = 0.
        hour_of_week = hour % (24*7)
        # Get the weather data for this hour:
        insolation = data_weather.insolation[hour]
        wind = data_weather.wind[hour]
        Toutside = data_weather.temperature[hour]
        # Perform the HVAC modelling for this hour:
        time_c = model_hvac.cooler_time(house, house_temp, Toutside)
        time_h = model_hvac.heater_time(house, house_temp, Toutside)
        house_temp = model_hvac.recalc.temperature(
            house, house_temp, Toutside, time_c, time_h
        )
        energy_usage += time_c * data_hvac.Pac_elec
        energy_usage += time_h * data_hvac.Pheater_elec
        if time_c: peakpower += data_hvac.Pac_elec
        if time_h: peakpower += data_hvac.Pheater_elec
        # Do appliance modeling:
        energy_usage += data_appliances.energy[hour_of_week]
        peakpower += data_appliances.peakpower[hour_of_week]
        # Do power system modeling:
        wind_power = data_power.wind_to_power(wind) * data_power.windmill_count
        solar_power = data_power.insolation_to_power(insolation)
        energy_source = (wind_power + solar_power) * 3600.
        battery_delta = energy_source - energy_usage
        battery_level += battery_delta
        if battery_level > data_power.battery_capacity:
            energy_unused = battery_level - data_power.battery_capacity
            battery_level = data_power.battery_capacity
        elif battery_level < 0.:
            energy_shortage = -battery_level
            battery_level = 0
        # print simulation results:
        print '>>datafile, hour, hour%24, time_c, time_h, CtoF(house_temp), \
            energy_usage, peakpower, energy_source, battery_level, \
            energy_unused, energy_shortage, wind_power, solar_power, \
            CtoF(Toutside)
        # Calculate aggregates:
        total_energy_needed += energy_usage
        total_energy_shortage += energy_shortage
    print 'total_energy_needed (Wh) =', total_energy_needed * Wh_per_joule
    print 'total_energy_shortage (Wh) =', total_energy_shortage * Wh_per_joule

colmap = {
    'hour': (1, ''),
    'time_c': (3, 'cooler time'),
    'time_h': (4, 'heater time'),
    'tempF': (5, 'indoor temp F'),
    'e-use': (6, 'e_needed'),
    'peakpower': (7, 'peakpower'),
    'e_source': (8, 'e_source'),
    'b_level': (9, 'battery level'),
    'e_unused': (10, 'e_unused'),
    'e_shortage': (11, 'e_shortage'),
    'wind_power': (12, 'wind_power'),
    'solar_power': (13, 'solar_power'),
    'Toutside': (14, 'outdoor temp F')
}

plots = [
    ('watt hours', Wh_per_joule, ('e-use', 'e_shortage')),
    ('watt hours', Wh_per_joule, ('b_level',)),
    ('watt hours', Wh_per_joule, ('wind_power', 'solar_power')),
    ('deg F', 1., ('tempF', 'Toutside')),

```

```

('watts', 1., ('peakpower',)),
('% of time', 100./3600., ('time_c', 'time_h')),
]

output_str = '''
set term postscript eps
#set term png
set output '%s.eps'
set size 1.7,1.2
'''

grid_str = '''
set xtics border (%s)
set grid xtics
'''

gpfiler = open('gpfiler', 'w')

for i in range(len(plots)):
    ylabel = plots[i][0]
    scale = plots[i][1]
    cols = plots[i][2]
    print >>gpfiler, output_str % (str(i))
    print >>gpfiler, grid_str % ', '.join(["'%s' %s" % (x, str(x*24)) for x in range(30)])
    print >>gpfiler, "set ylabel '%s'" % ylabel
    print >>gpfiler, "set xlabel 'day'"
    plot = ', '.join(["data u ($1):(%s)*%e t '%s' w steps" % \
                      (colmap[c][0], scale, colmap[c][1]) for c in cols])
    print >>gpfiler, 'plot %s' % plot

main()

```

9.7 xpath.py

```

"""
This file is NOT the original work of David Baird. It is borrowed/adapted from
http://www.decafbad.com/blog/2004/09/01/xpath-based-python-dictionaries-on_loan

$Header: /cvsroot/dbagg3/lib/dbagg3/xmlutils.py,v 1.7 2004/09/13 13:04:44 deusx Exp $

Example:

feed_xd = XPathDict(file="sample-atom.xml")
for entry_node in feed_xd.nodes("//atom:entry"):
    entry = XPathDict(doc=entry_node.doc, node=entry_node)
    print "Title: " % entry['atom:title']
    if 'atom:author' in entry:
        print "Author: " % entry['atom:author/atom:name']

xml = '''
<dbagg3:user xmlns="http://purl.org/atom/ns#"
xmlns:dbagg3="http://decafbad.com/2004/07/dbagg3/">
<name>deusx</name>
<email>deus_x@pobox.com</email>
<url>http://www.decafbad.com/</url>
<dbagg3:prefs>
<dbagg3:pref name="foo">bar</dbagg3:pref>
</dbagg3:prefs>
</dbagg3:author>
'''

map = (
    ('userName', 'a:name'),
    ('userEmail', 'a:email'),
    ('fooPref', "dbagg3:prefs/dbagg3:pref[@name='foo']")
)

xd = XPathDict(xml=xml)
xd.cd("/dbagg3:user")
print xd.extract(map)

# {'userName' : 'deusx',
#   'userEmail' : 'deus_x@pobox.com',
#   'fooPref' : 'bar'}

"""

import sys, time, pickle, traceback, libxml2

from xml.sax import saxutils, make_parser, SAXParseException
from xml.sax.handler import feature_namespaces, feature_namespace_prefixes
from xml.sax import saxutils
from xml.sax.xmlreader import AttributesImpl
from StringIO import StringIO
import xml.sax._exceptions
import xml.sax.handler

INDENT = ' '
ATOM_VERSION = '0.3'

namespaces = {}

```

```

class XPathDict:
    namespaces=namespaces

    def __init__(self, doc=None, xml=None, file=None, node=None):
        self.ctx = None
        self.free_doc = True
        if xml:
            self.doc = libxml2.parseDoc(xml)
        elif file: self.doc = libxml2.parseFile(file)
        else:
            self.doc = doc
            self.free_doc = False
        ctx = self.doc.xpathNewContext()
        ctx.xpathRegisterNs('', 'http://purl.org/atom/ns#')
        ctx.xpathRegisterNs('dbagg3', 'http://decafbad.com/2004/07/dbagg3/')
        ctx.xpathRegisterNs('atom', 'http://purl.org/atom/ns#')
        ctx.xpathRegisterNs('a', 'http://purl.org/atom/ns#')
        for k,v in self.namespaces.items():
            ctx.xpathRegisterNs(k,v)
        self.ctx = ctx
        if node is not None:
            self.root = node
        else:
            self.root = self.doc.getRootElement()
        self.ctx.setContextNode(node)

    def __del__(self):
        if self.ctx:
            self.ctx.xpathFreeContext()
        if self.doc and self.free_doc:
            self.doc.freeDoc()

    def __getitem__(self, xpath):
        try:
            return self.nodes(xpath)[0].content
        except:
            return None

    def __delitem__(self, xpath):
        nodes = self.nodes(xpath)
        for n in nodes:
            n.unlinkNode()
            n.freeNode()

    def __contains__(self, xpath):
        rs = self.nodes(xpath)
        return rs is not None and len(rs)>0

    def get(self, key, default=None):
        return self.__getitem__(self, key, default)

    def has_key(self, key):
        return self.__contains__(key)

    def nodes(self, xpath):
        return self.ctx.xpathEval(xpath)

    def cd(self, xpath=None, node=None):
        if node is None:
            node = self.ctx.xpathEval(xpath)[0]
        self.root = node
        self.ctx.setContextNode(node)

    def xml(self):
        return self.root.serialize(format=1)

    def extract(self, extract_keys, extract_xml=False):
        data = dict([(k, self[p]) for k,p in extract_keys \
                    if self.has_key(p)])
        if extract_xml:
            data['xml'] = self.root.serialize(format=1)
        return data

class XMLGenerator(saxutils.XMLGenerator):
    INDENT=INDENT
    ATOM_VERSION=ATOM_VERSION
    namespaces=namespaces

    def __init__(self, encoding=None, stream=None, base_href='http://localhost'):
        if encoding == None: encoding = 'utf-8'
        if stream == None: stream = sys.stdout

        self.stream = stream
        saxutils.XMLGenerator.__init__(self, stream, encoding=encoding)
        self.base_href = base_href
        self.encoding = encoding
        self.reset()

    def reset(self):
        self._indent = 0

```

```

        self._emitted_NS = False
def omitNS(self):
    self._emitted_NS = True
def setNS(self, ns):
    self.namespaces = ns
def doIndent(self):
    for x in range(self._indent):
        self.characters(self.INDENT)
def elemStart(self, name, attr={}):
    self.doIndent()

    if not self._emitted_NS:
        ns_attr = {
            'xml:base'      : self.base_href,
            'xmlns'        : 'http://purl.org/atom/ns#',
            'xmlns:dbagg3'  : 'http://decafbad.com/2004/07/dbagg3/',
            'xmlns:xlink'   : 'http://www.w3.org/1999/xlink'
        }
        for k,v in self.namespaces.items():
            ns_attr['xmlns:%s'%k]=v
        ns_attr.update(attr)
        attr = ns_attr
        self._emitted_NS = True

    self.startElement(name, AttributesImpl(attr))
    self.characters('\n')
    self._indent += 1
def elemEnd(self, name):
    self._indent -= 1
    self.doIndent()
    self.endElement(name)
    self.characters('\n')
def elemWithContent(self, name, content, attr={}):
    self.doIndent()
    attr = dict( map( lambda k: (k, attr[k].encode(self.encoding, 'xmlcharrefreplace')), attr.keys() ) )
    self.startElement(name, AttributesImpl(attr))
    if content is not None:
        self.characters(content.encode(self.encoding, 'xmlcharrefreplace'))
    self.endElement(name)
    self.characters('\n')
def linkBuild(self, rel, href, type, title):
    attr = { 'rel'      : rel,
            'href'     : href,
            'type'     : type,
            'title'    : title }
    self.elemWithContent('link', None, attr=attr)

```

9.8 model_hvac.py

```

from math import exp, log as ln
class House:
    def __init__(self,
                 theta1, theta2,
                 Pac_elec, SEER_ac,
                 Pheater_elec, SEER_heater,
                 Tground,
                 heat_capacity,
                 Thousemax,
                 Thousemin):
        ,,,
        All units:
        temperature in deg C
        heat capacity in joules per kelvin
        power in watts
        thermal resistance in deg C per watt
        SEER is dimensionless
        ,,,
        self.theta1 = theta1
        self.theta2 = theta2
        self.Pac_elec = Pac_elec
        self.SEER_ac = SEER_ac
        self.Pheater_elec = Pheater_elec
        self.SEER_heater = SEER_heater
        self.Tground = Tground
        self.heat_capacity = heat_capacity
        self.Thousemax = Thousemax
        self.Thousemin = Thousemin
    def theta(self):
        t1, t2 = map(float, [self.theta1, self.theta2])
        return (t1 * t2) / (t1 + t2)
    def Text(self, Toutside):
        t1, t2 = map(float, [self.theta1, self.theta2])

```

```

        return (t1 * self.Tground + t2 * Toutside) / (t1 + t2)
def tau(self):
    return self.heat_capacity * self.theta()
def Pac(self):
    return self.Pac_elec * self.SEER_ac * .293
def Pheater(self):
    return self.Pheater_elec * self.SEER_heater * .293

def cooler_timeA(house, Tinside, Toutside, t_period = 3600.):
    """
    Something seems fishy about this method of calculating the cooler
    time... did I do my math correctly?
    """
    tau = house.tau()
    Text = house.Text()
    theta = house.theta()
    Pac = house.Pac()
    Tmax = house.Thousemax
    Tmin = house.Thousemin
    t1 = tau * ln( (Text - Tinside) / (Text + Tmax - 2*Tinside) )
    t_ac = (Text - Tmax) / theta / Pac * (t_period - t1)
    if t_ac < 0:
        t_ac = 0.
    return t_ac

def cooler_timeB(house, Tinside, Toutside, t_period = 3600.):
    Pac = house.Pac()
    Tmax = house.Thousemax
    c = house.heat_capacity
    # forecasted_T = what would the temperature be without cooling or heating?
    forecasted_T = recalc_temperature(house, Tinside, Toutside, 0, 0, t_period)
    t_ac = 0.
    if forecasted_T > Tmax:
        excess_energy = (forecasted_T - Tmax) * c
        t_ac = excess_energy / Pac
    else:
        t_ac = 0.
    if t_ac > t_period:
        t_ac = t_period
    return t_ac

def heater_timeB(house, Tinside, Toutside, t_period = 3600.):
    Pac = house.Pac()
    Tmin = house.Thousemin
    c = house.heat_capacity
    # forecasted_T = what would the temperature be without cooling or heating?
    forecasted_T = recalc_temperature(house, Tinside, Toutside, 0, 0, t_period)
    t_heater = 0.
    if forecasted_T < Tmin:
        energy_shortage = (Tmin - forecasted_T) * c
        t_heater = energy_shortage / Pac
    else:
        t_heater = 0.
    if t_heater > t_period:
        t_heater = t_period
    return t_heater

cooler_time = cooler_timeB # use method B for cooler_time
heater_time = heater_timeB

def recalc_temperature(house, Tinside, Toutside, cooler_time, heater_time,
                       t_period = 3600):
    tau = house.tau()
    Text = house.Text(Toutside)
    Pac = house.Pac()
    Pheat = house.Pheater()
    c = house.heat_capacity
    new_Tinside = Tinside + (1 - exp(-t_period / tau)) * (Text - Tinside) \
        + (heater_time * Pheat - cooler_time * Pac) / c
    return new_Tinside

```

9.9 resources.xml

```

<resources>
<!--
    hvac:
        insulation
        heater
        ecooler
        scooler
    appliance-rates
        rate
    activities
        activity
            euse - electricity usage
            wuse - water usage
-->
<hvac-appliances>
<!-- r30 = 30 feet*degF*hour/btu -->

```

```

<insulation id="r30" value="30" unit="feet*degF*hour/BTU">
  <name>R30 insulation</name>
  <note>30 \frac{feet \cdot degF \cdot hour}{btu}</note>
</insulation>
<hvac id="heatpump">
  <name>heatpump</name>
  <note>17.9 SEER</note>
</hvac>
<hvac id="refrigair">
  <name>refrig. air</name>
  <note>19.5 SEER</note>
</hvac>
<hvac id="swamp1">
  <name>PNM swamp</name>
  <note>250kWh/month, 775 gal/month</note>
</hvac>
<hvac id="refrigair2">
  <name>PNM refriger. air</name>
  <note>850 kWh/month</note>
</hvac>
</hvac-appliances>
<appliances>
  <!-- Fuller -->
  <appliance id="hairdryer">
    <rate value="1600" unit="watt"/>
    <name>Hair Dryer</name>
  </appliance>
  <appliance id="curlingiron">
    <rate value="85" unit="watt"/>
    <name>Curling Iron</name>
  </appliance>
  <!-- Baird -->
  <appliance id="tv">
    <rate value="150" unit="watt"/>
    <name>28in TV</name>
  </appliance>
  <appliance id="laptop">
    <rate value="50" unit="watt"/>
    <name>Laptop</name>
  </appliance>
  <appliance id="dvd">
    <rate value="30" unit="watt"/>
    <name>DVD player</name>
  </appliance>
  <appliance id="stereo">
    <rate value="140" unit="watt"/>
    <name>Stereo</name>
  </appliance>
  <appliance id="satellite">
    <rate value="30" unit="watt"/>
    <name>Satellite</name>
  </appliance>
  <appliance id="light">
    <rate value="60" unit="watt"/>
    <name>Light bulb, 60W</name>
  </appliance>
  <appliance id="desktop">
    <rate value="330" unit="watt"/>
    <name>Desktop computer</name>
  </appliance>
  <appliance id="monitor">
    <rate value="200" unit="watt"/>
    <name>Computer monitor</name>
  </appliance>
  <appliance id="printer">
    <rate value="200" unit="watt"/>
    <name>Computer printer</name>
  </appliance>
  <!-- Kadjo -->
  <!--
  <appliance id="washer">
    <rate value="296" unit="kWh/year"/>
    <name>Washer</name>
  </appliance>
  -->
  <appliance id="washer">
    <rate value="1900" unit="watt"/> <!-- 296000./24/365*(24*7)/3 -->
    <name>Washer</name>
  </appliance>
  <appliance id="dryer">
    <rate value="5400" unit="watt"/>
    <name>Dryer</name>
  </appliance>
  <appliance id="iron">
    <!-- 1200 watt -->
    <rate value="150" unit="watt"/>
    <name>Clothes iron</name>
  </appliance>
  <appliance id="microwave">
    <rate value="1800" unit="watt"/>

```

```

        <name>Microwave</name>
    </appliance>
    <appliance id="oven">
        <rate value="1200" unit="watt"/>
        <name>Oven and Range</name>
    </appliance>
    <appliance id="slowcooker">
        <rate value="650" unit="watt"/>
        <name>Slow Cooker</name>
    </appliance>
</appliances>

<periodic-appliances>
    <!--
        You can specify an average rate or peek rate for each appliance
        in this list ...
        peakrate = averate / dutycycle
        averate = peakrate * dutycycle
        Optionally, you can also specify a period.
    -->
    <!-- Kadjo -->
    <appliance id="fridge">
        <averate value="39" unit="kWh/month" dutycycle="0.30"/>
        <name>Refridgerator</name>
    </appliance>
    <!-- Baird -->
    <appliance id="waterheater">
        <!-- Sears 30 gal heater, assuming 1650 watts when on -->
        <averate value="4721" unit="kWh/year" dutycycle="0.33"/>
        <name>Water Heater</name>
    </appliance>
</periodic-appliances>

<activities>
    <weekly>
        <activity id="hairdrying">
            <time value="4" unit="min"/>
            <use ref="hairdryer"/>
        </activity>
        <activity id="shaving">
            <time value="4" unit="min"/>
            <use ref="faucet"/>
        </activity>
        <activity id="brushteeth">
            <time value="8" unit="min"/>
            <use ref="faucet"/>
        </activity>
    </weekly>
    <alwayson>
        <use ref="fridge"/>
    </alwayson>
    <schedule id="weekday">
        <name>Weekday Schedule</name>
        <!-- Baird -->
        <use ref="light" count="6" from="600" to="800"/>
        <use ref="light" count="6" from="1800" to="2200"/>
        <use ref="stereo" from="1800" to="1900"/>
        <use ref="satellite" from="1800" to="1900"/>
        <use ref="tv" from="1800" to="1900"/>
        <use ref="desktop" from="1900" to="2200"/>
        <use ref="monitor" from="1900" to="2200"/>
        <use ref="laptop" from="1900" to="2200"/>
        <!-- Kadjo -->
        <use ref="microwave" from="1900" to="1930"/>
    </schedule>
    <schedule id="saturday">
        <name>Saturday Schedule</name>
        <!-- Baird -->
        <use ref="light" count="6" from="600" to="900"/>
        <use ref="light" count="6" from="1800" to="2200"/>
        <use ref="stereo" from="1200" to="2100"/>
        <use ref="tv" from="1800" to="2100"/>
        <use ref="dvd" from="1800" to="2100"/>
        <use ref="desktop" from="2100" to="2200"/>
        <use ref="monitor" from="2100" to="2200"/>
        <use ref="printer" from="2100" to="2200"/>
        <!-- Kadjo -->
        <use ref="oven" from="900" to="1100"/>
        <use ref="oven" from="1800" to="2000"/>
        <use ref="slowcooker" from="900" to="1900"/>
    </schedule>
    <schedule id="sunday">
        <name>Sunday Schedule</name>
        <!-- Baird -->
        <use ref="light" count="6" from="600" to="800"/>
        <use ref="light" count="6" from="1800" to="2200"/>
        <!-- Kadjo -->
        <use ref="washer" from="900" to="1200"/>
        <use ref="dryer" from="900" to="1200"/>
        <use ref="iron" from="1900" to="1930"/>
        <use ref="oven" from="1800" to="2000"/>
    </schedule>

```

```
</schedule>  
</activities>  
</resources>
```

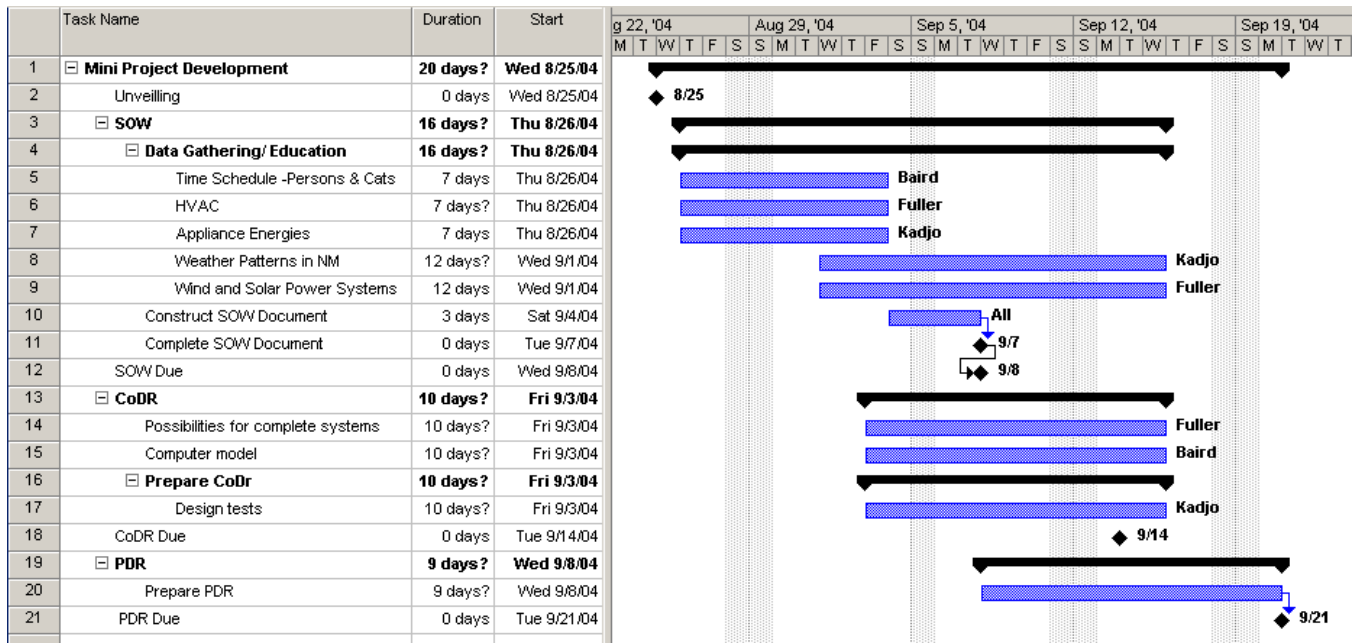


Figure 1: MS Project Tracking Gantt Chart

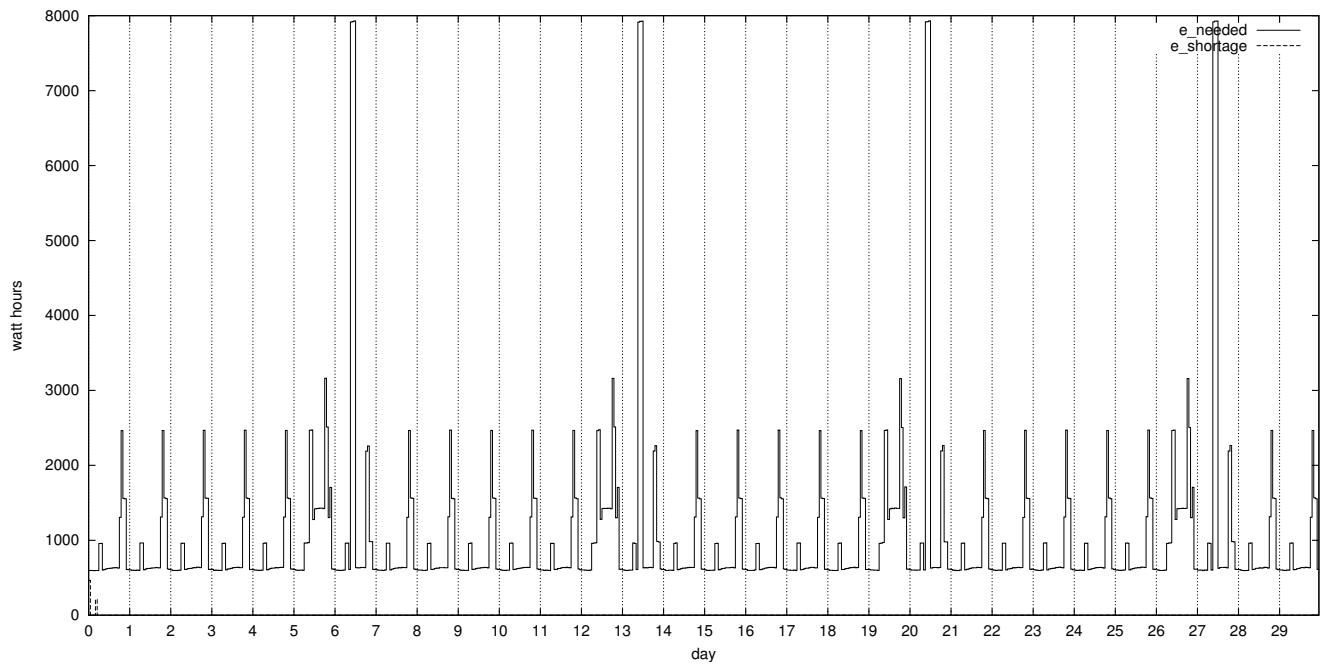


Figure 2: Daily Energy Consumption

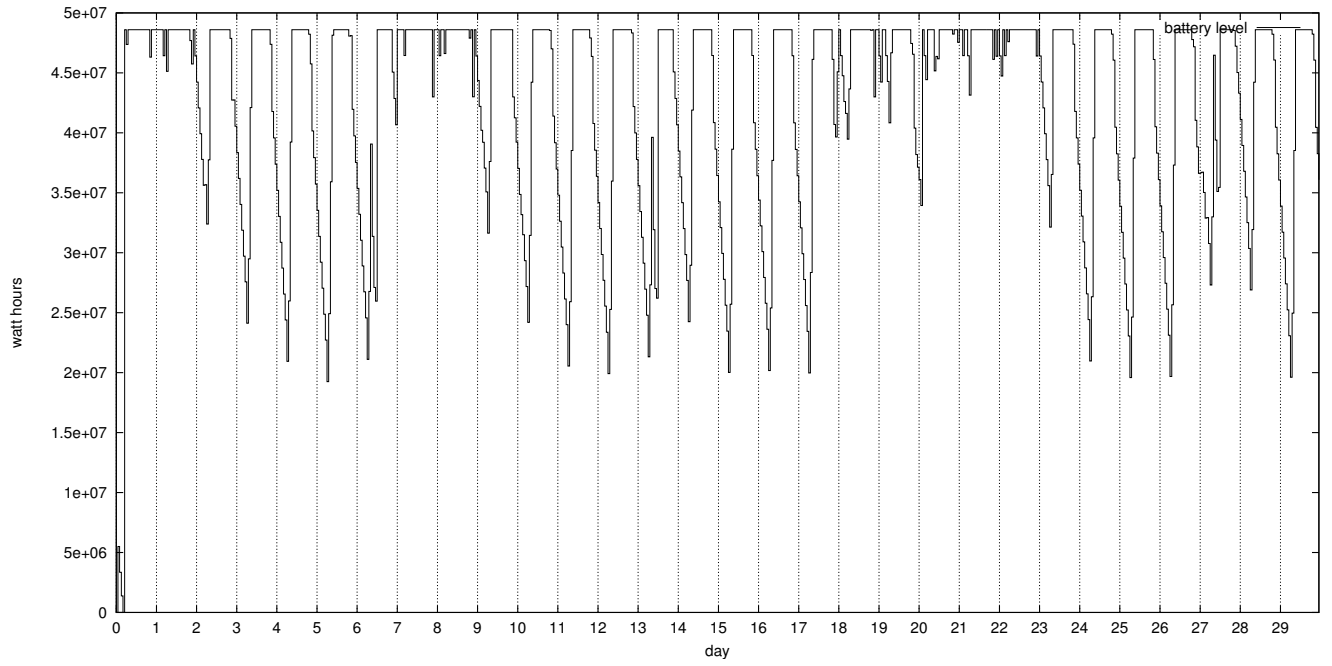


Figure 3: Daily Battery Level

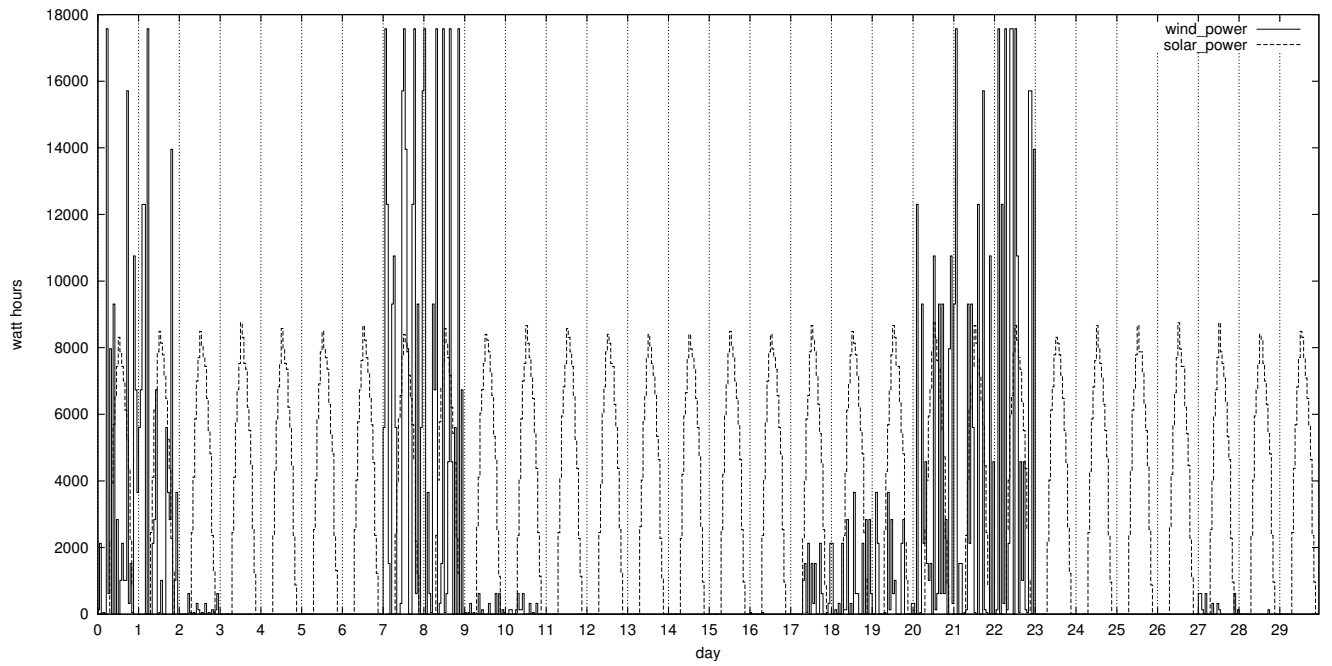


Figure 4: Daily Wind and Solar Power Production

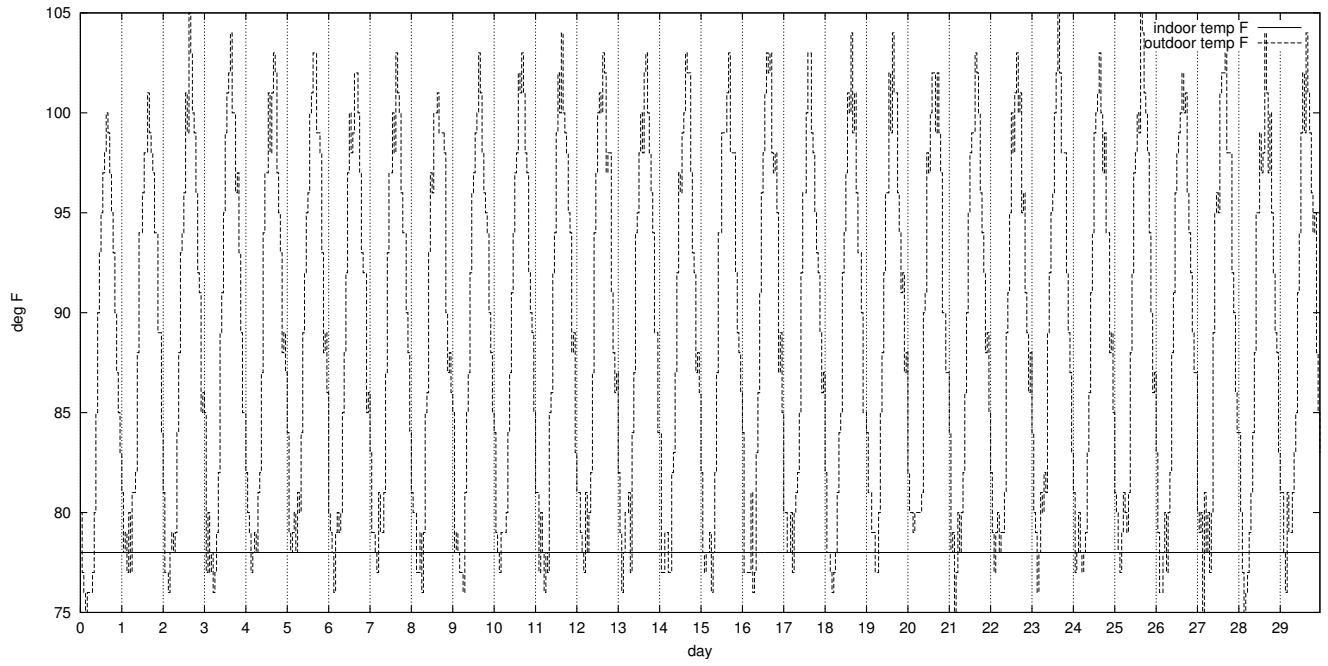


Figure 5: Indoor vs. Outdoor Temperature

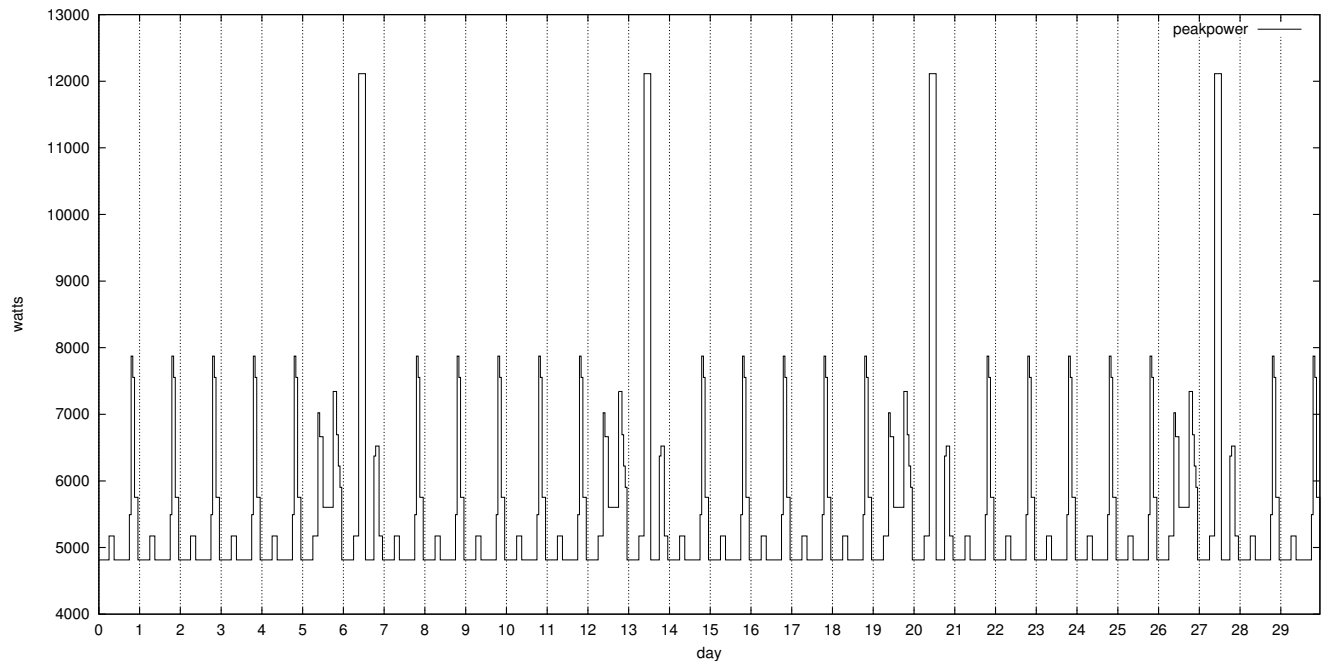


Figure 6: Peak Power Consumption

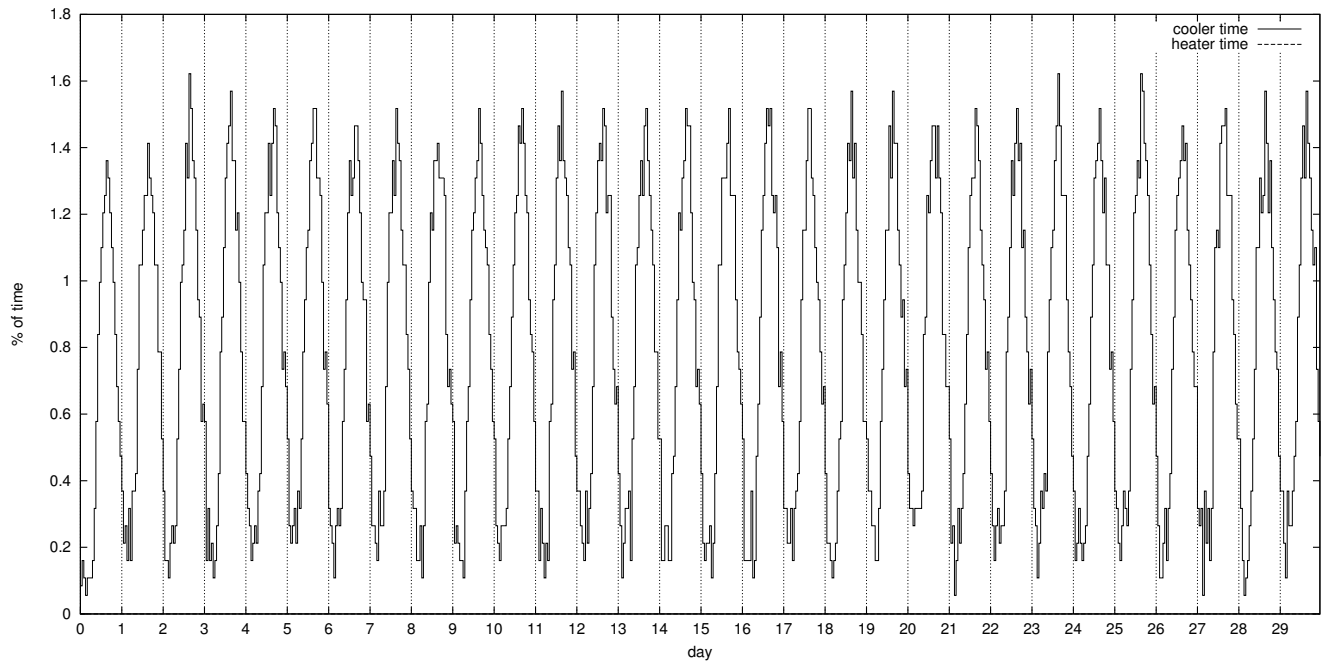


Figure 7: HVAC Usage

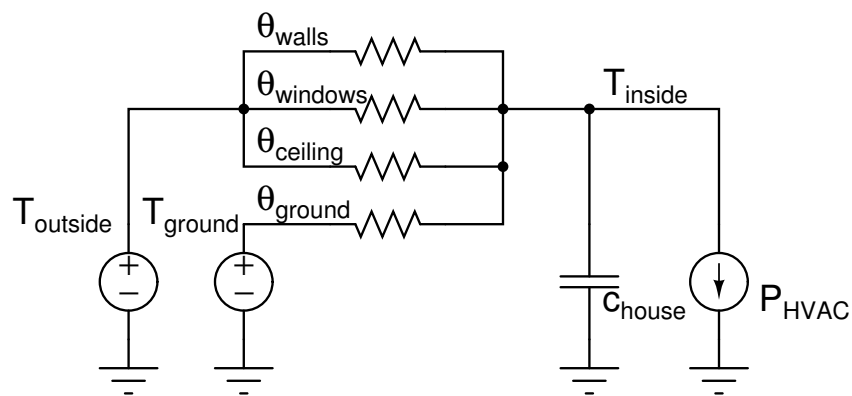


Figure 8: Temperature Model