

# INTELLIGENT THERMOSTAT

SIMPLE SYSTEMS

## Design Document

Kenneth Fisher  
Christian McArthur  
John Rawls  
Patrick Read

# Table of Contents

1	Introduction.....	3
2	Base System.....	3
2.1	Server Software.....	3
2.2	Physical Switches.....	5
2.3	Device Display.....	6
2.4	HVAC Machine.....	6
2.5	Thermocouple.....	8
3	Web-based User Interface.....	8
3.1	HTML Interface.....	8
3.2	Security & Authentication.....	9
3.3	Scheduler.....	10
3.3.1	TimeTemp Storage Structure.....	11
3.4	History Log.....	11
4	Traceability Matrix.....	12
5	Diagrams.....	14
5.1	UML Component Diagram.....	14
5.2	HVAC Machine State Diagram.....	15

# 1 Introduction

This document explains the design for Simple Systems' Intelligent Thermostat. Most of the information provided in this document comes from information stored in a Rational Rose UML design. The overall design includes the following components:

- Base System: Includes the server software, physical switches, device display, HVAC machine, and thermocouple. The base system is described in Section 2.
- Web-based User Interface: The thermostat has a web-based HTML user interface that includes security & authentication, a heating/cooling schedule, and history logging features. The user interface design is described in Section 3.
- Test Harness: A simple test harness is needed to test the overall system. Since it is not part of the final implementation it is not included as part of the UML design or this document.

The relationship between the various components of the thermostat design can be traced to requirements and the user statement of need on the traceability matrix in Section 4. Additional diagrams, such as the actual UML diagram, can be found in Section 5.

## 2 Base System

This section describes the Base System and the components within it. The base system includes the server software, physical switches, device display, HVAC machine and the thermocouple.

### 2.1 Server Software

#### Documentation

This is the BaseSystem. It is responsible for interacting with the various physical modules of the thermostat system including the physical display, physical switches, the HVAC machine and the thermocouple. It also provides the method for interprocess communication with the user interface and the test harness.

#### Operations

Name	Signature	Class
PollTemperatureState	Integer PollTemperatureState ()	BaseSystem
<b>Documentation</b> This function will poll the Thermocouple for the current temperature. <b>Preconditions</b> The function within the thermocouple class that is expected to be called will be getCurrentTemperature(). <b>Postconditions</b> Temperature returned will be the current temperature in Fahrenheit. The temperature will be an integer value.		
getScheduledTemp	bool getScheduledTemp (int* highTemp, int* lowTemp)	BaseSystem
<b>Documentation</b> This function will check the schedule and see what current temperature range should be for that moment in time. <b>Postconditions</b> The function will receive a pair of temperatures (a high and a low) from the schedule class. The function will also return a boolean value: true if the schedule is enabled, false if there has been no schedule enabled.		
setHVACStatus	bool setHVACStatus (enum status)	BaseSystem
<b>Documentation</b> Sets the status of the HVAC. This function should call the related function within the HVACMachine class to		

activate/deactivate the appropriate mechanism.

**Preconditions**

The operation type argument will be one of several values: off, idle, heat, or cool.

**Postconditions**

The function will return true if the HVACMachine status was set correctly. It will return false if there is there were any problems.

ListenForClient	void* ListenForClient ()	BaseSystem
-----------------	--------------------------	------------

**Documentation**

This function will create a port to communicate with external programs such as the GUI interface and test programs.

HandleArguments	void HandleArguments (Integer argc, char* args)	BaseSystem
-----------------	---	------------

**Documentation**

This function will handle any command line arguments given to the server. This will occur only during testing.

switchChange	void switchChange (int sig)	BaseSystem
--------------	-----------------------------	------------

**Documentation**

This function will be called when a signal is received from the physical switches announcing the switch has changed positions. The function will query the PhysicalSwitches module for the current switch state and will update the HVAC\_Machine as necessary.

**Preconditions**

The signal that should be recieved to call this function should be SIGUSR1.

HandleClientComm	void HandleClientComm (int clntSocketd)	BaseSystem
------------------	---	------------

**Documentation**

This function will parse the client program's request and perform the specified action. Any return values from the requested functions will be passed back to the client.

**Preconditions**

The client should already have established a connection with the server and a valid socket descriptor for the client should exist.

atoe	EventType atoe (string arg)	BaseSystem
------	-----------------------------	------------

**Documentation**

This function coverts strings into a HistoryLog EventType.

**Preconditions**

The string recieved should match to a valid EventType.

**Attributes**

Name	Class	Type	Initial Value
current_temp	BaseSystem	Integer	70
<b>Documentation</b>			
This is the current temperature as reported by the last polling of the thermocouple module.			
scheduled_temp_high	BaseSystem	Integer	0
<b>Documentation</b>			
This is the high temperature as reported by the scheduler.			
scheduled_temp_low	BaseSystem	Integer	0
<b>Documentation</b>			
This is the low temperature as reported by the scheduler.			

## Associations

MyRole	MyClass	OtherRole	OtherElement
InterProcess Communication	BaseSystem	InterProcess Communication	HTMLGUI
Instantiation	BaseSystem		HVACMachine
Instantiation	BaseSystem		HistoryLog
Instantiation	BaseSystem		Thermocouple
Instantiation	BaseSystem		PhysicalSwitches
Instantiation	BaseSystem		DeviceDisplay
Instantiation	BaseSystem		SecurityModule
Instantiation	BaseSystem		Scheduler

## 2.2 Physical Switches

### Documentation

This class is a replaceable module. It will interact with the physical switches on the thermostat. When the switches change position it will send a signal to the BaseSystem.

### Operations

Name	Signature	Class
on	on()	PhysicalSwitches
off	off()	PhysicalSwitches
getSwitchState	boolgetSwitchState()	PhysicalSwitches

**Documentation**  
This function returns the current state of the physical switch.  
True = On  
False = Off

### Attributes

Name	Class	Type	InitialValue
switchState	PhysicalSwitches	bool	

**Documentation**  
Represents the current state of the physical switch.  
True = on  
False = off

### Associations

MyRole	MyClass	OtherRole	OtherElement
--NotNamed--	PhysicalSwitches	Instantiation	BaseSystem

## 2.3 Device Display

### Documentation

This is a test module representing the thermostat's physical device display. It currently only shows the current environment temperature.

### Operations

Name	Signature	Class
setTemp	voidsetTemp(intx)	DeviceDisplay
<b>Documentation</b> This function will receive the current temperature from the BaseSystem. <b>Preconditions</b> The temperature received from the BaseSystem will be an integer value representing the temperature in Fahrenheit.		
showTemp	intshowTemp()	DeviceDisplay
<b>Documentation</b> This function returns the current temperature being displayed.		

### Attributes

Name	Class	Type	InitialValue
temp	DeviceDisplay	Integer	

### Associations

MyRole	MyClass	OtherRole	OtherElement
--NotNamed--	DeviceDisplay	Instantiation	BaseSystem

## 2.4 HVAC Machine

### Documentation

This is a replaceable module. It is currently a test module representing the HVAC machine. It will receive instructions from the base system about whether to activate or deactivate the heater or AC.

### Operations

Name	Signature	Class
heaterActivate	BooleanheaterActivate()	HVACMachine
<b>Documentation</b> When this function is called the HVAC Machine should turn on the heater. It returns true if the state change was successful. <b>Postconditions</b> The function will return true if the change of state was accepted without problem. It will return false if there was any error.		
heaterDeactivate	BooleanheaterDeactivate()	HVACMachine
<b>Documentation</b> When this function is called the HVAC Machine should turn off the heater. It returns true if the state change was successful. <b>Postconditions</b> The function will return true if the change of state was accepted without problem. It will return		

false if there was any error.

ACActivate	BooleanACActivate()	HVACMachine
<b>Documentation</b> When this function is called the HVAC Machine should turn on the air conditioner. It returns true if the state change was successful. <b>Postconditions</b> The function will return true if the change of state was accepted without problem. It will return false if there was any error.		
ACDeactivate	BooleanACDeactivate()	HVACMachine
<b>Documentation</b> When this function is called the HVAC Machine should turn off the air conditioner. It returns true if the state change was successful. <b>Postconditions</b> The function will return true if the change of state was accepted without problem. It will return false if there was any error.		
HVAC_OFF	boolHVAC_OFF()	HVACMachine
<b>Documentation</b> This function turns the HVAC_Machine off. It returns true if the state change was successful.		
HVAC_IDLE	boolHVAC_IDLE()	HVACMachine
<b>Documentation</b> This function puts the HVAC_Machine in an idle state. It returns true if the state change was successful.		
getStatus	StatusgetStatus()	HVACMachine
<b>Documentation</b> This function returns the current state of the HVAC machine.		

#### Attributes

Name	Class	Type	InitialValue
currStatus	HVACMachine	enum	off
<b>Documentation</b> This variable represents the status of the HVAC machine. It can have the values of: heat, cool, idle, and off.			
Status	HVACMachine	enum	
<b>Documentation</b> This enum variable provides a list of possible states for the HVAC machine. The available options are: AC HEAT IDLE OFF			

#### Associations

MyRole	MyClass	OtherRole	OtherElement
--NotNamed--	HVACMachine	Instantiation	BaseSystem

## 2.5 Thermocouple

### Documentation

This is a test module representing the thermocouple.

### Operations

Name	Signature	Class
checkTemp	IntegercheckTemp()	Thermocouple
<b>Documentation</b> This function returns the current environment temperature.		
updateTemp	voidupdateTemp(HVAC_Machine*HVAC)	Thermocouple
<b>Documentation</b> This function will update the environment temperature. If the AC is current on, the temp will go down. If the heater is on, the temp goes up. Otherwise, the temp changes randomly (40% up, 40% down, 20% stays the same).		
setTemp	voidsetTemp(intnewTemp)	Thermocouple
<b>Documentation</b> Allows the environment temperature to be changed to the specified value.		

### Attributes

Name	Class	Type	InitialValue
temp	Thermocouple	Integer	
<b>Documentation</b> This variable contains the current environment temperature.			

### Associations

MyRole	MyClass	OtherRole	OtherElement
--NotNamed--	Thermocouple	Instantiation	BaseSystem

## 3 Web-based User Interface

This section describes the web-based user interface. The user will have an HTML-based interface that is accessible via a web browser from a computer on the same home network as the thermostat or via the Internet should the thermostat be connected to the Internet. The user interface will include an authentication system, a method of scheduling the temperature ranges, and a history log feature.

### 3.1 HTML Interface

#### Documentation

The GUI is a series of HTML and PHP pages that provide a web-based mechanism for configuring the thermostat, viewing the current environment temperature, and checking the history log.

### Operations

Name	Signature	Class
session_start	session_start()	HTMLGUI
session_destroy	session_destroy()	HTMLGUI

### Attributes

Name	Class	Type	InitialValue
\$_SESSION['authenticated']	HTMLGUI	Integer	

### Associations

MyRole	MyClass	OtherRole	OtherElement
Client, IPC	HTMLGUI	Server, IPC	BaseSystem



## 3.2 Security & Authentication

### Documentation

Provides mechanism for authenticating proper user(s) of ThermoSystem

### Operations

Name	Signature	Class
validateLogin	BooleanvalidateLogin(char*password)	SecurityModule
<b>Preconditions</b>		

The password must be received as a SHA1 encoded string.

**password**

This is a pointer to a character array of size 40.

changePassword	boolchangePassword(char*oldpw,char*newpw)	SecurityModule
----------------	---	----------------

**Documentation**

This function allows the system password to be changed.

**Preconditions**

Both the old password and the new password must be recieved as a SHA1 encoded string.

**oldpw**

The old password.

**newpw**

The new password.

**Attributes**

Name	Class	Type	InitialValue
user_password	SecurityModule	char	

**Documentation**

This is a SHA1 formatted string containing the system password.

**Associations**

MyRole	MyClass	OtherRole	OtherElement
--NotNamed--	SecurityModule	Instantiation	BaseSystem

### 3.3 Scheduler

**Documentation**

This module contains functionality to allow for temperature schedules. There are 5 available schedules per day.

**Operations**

Name	Signature	Class
scheduleTemp	BooleanscheduleTemp(weekday,start_time,temp_min,temp_max)	Scheduler
getScheduledTemp	boolgetScheduledTemp(short*low,short*high)	Scheduler

**Documentation**

This function will provide the current temperature range.

**Preconditions**

This function needs to receive pointers to short integers where the two temperature ranges need to be stored at.

setSchedule	boolsetSchedule(char*schedule_string)	Scheduler
-------------	---------------------------------------	-----------

**Documentation**

This function will recieve a string and will create a schedule of temperature ranges from it.

**Preconditions**

The string received must be formatted correctly.

## Attributes

Name	Class	Type	InitialValue
schedule[7][5]	Scheduler	TimeTemp	

## Associations

MyRole	MyClass	OtherRole	OtherElement
--NotNamed--	Scheduler	Instantiation	BaseSystem
Parent Class	Scheduler	Member Struct	TimeTemp

## 3.3.1 TimeTemp Storage Structure

### Documentation

This is the data structure used to keep track of the various scheduler settings.

### Attributes

Name	Class	Type	InitialValue
start_hr	TimeTemp	short	0
start_min	TimeTemp	short	0
low	TimeTemp	short	68
high	TimeTemp	short	72
enable	TimeTemp	short	0

### Associations

MyRole	MyClass	OtherRole	OtherElement
Member Struct	TimeTemp	Parent Class	Scheduler

## 3.4 History Log

### Documentation

This module contains methods for logging information about system events and for querying the recorded information.

### Operations

Name	Signature	Class
logQuery	StringlogQuery(Integeroffset=0,Integercount=20)	HistoryLog
<b>Documentation</b> This function provides a mechanism for querying the contents of the log file.		
<b>Postconditions</b> The string returned from the function will be in the following format: 1208460166 1 72 70 77 0 Optional string message(s) here 10bytes 5b 5b 5b 5b 5b up to 40 chars		
createLogEntry	BooleancreateLogEntry(EventTypeentry_type,Integercurr_temp,Integermin_temp=0,Integermax_temp=0,Integeruser_id=0,Stringdesc=(empt	HistoryLog

	y))	
<b>Preconditions</b>		
The EventType and the current temperature must be specified.		
readLogFile	voidreadLogFile()	HistoryLog
writeLogFile	voidwriteLogFile()	HistoryLog

#### Attributes

Name	Class	Type	Initial Value
EventType	HistoryLog	enum	
<b>Documentation</b>			
This is enum contains a list of all valid EventTypes that can be logged. The values are: UNSPECIFIED STATUS_REPORT POWER_ON POWER_OFF HEATER_ON HEATER_OFF COOLING_ON COOLING_OFF FAN_ON FAN_OFF HEATER_ON_OVERRIDE HEATER_OFF_OVERRIDE COOLING_ON_OVERRIDE COOLING_OFF_OVERRIDE TEMP_RANGE_CHANGE MIN_TEMP_CHANGE MAX_TEMP_CHANGE USER_LOGIN USER_CREATE USER_DELETE LOG_ROTATE LOG_ERASE LOG_RELOAD LOG_WRITE			

#### Associations

My Role	My Class	Other Role	Other Element
--Not Named--	HistoryLog	Instantiation	BaseSystem

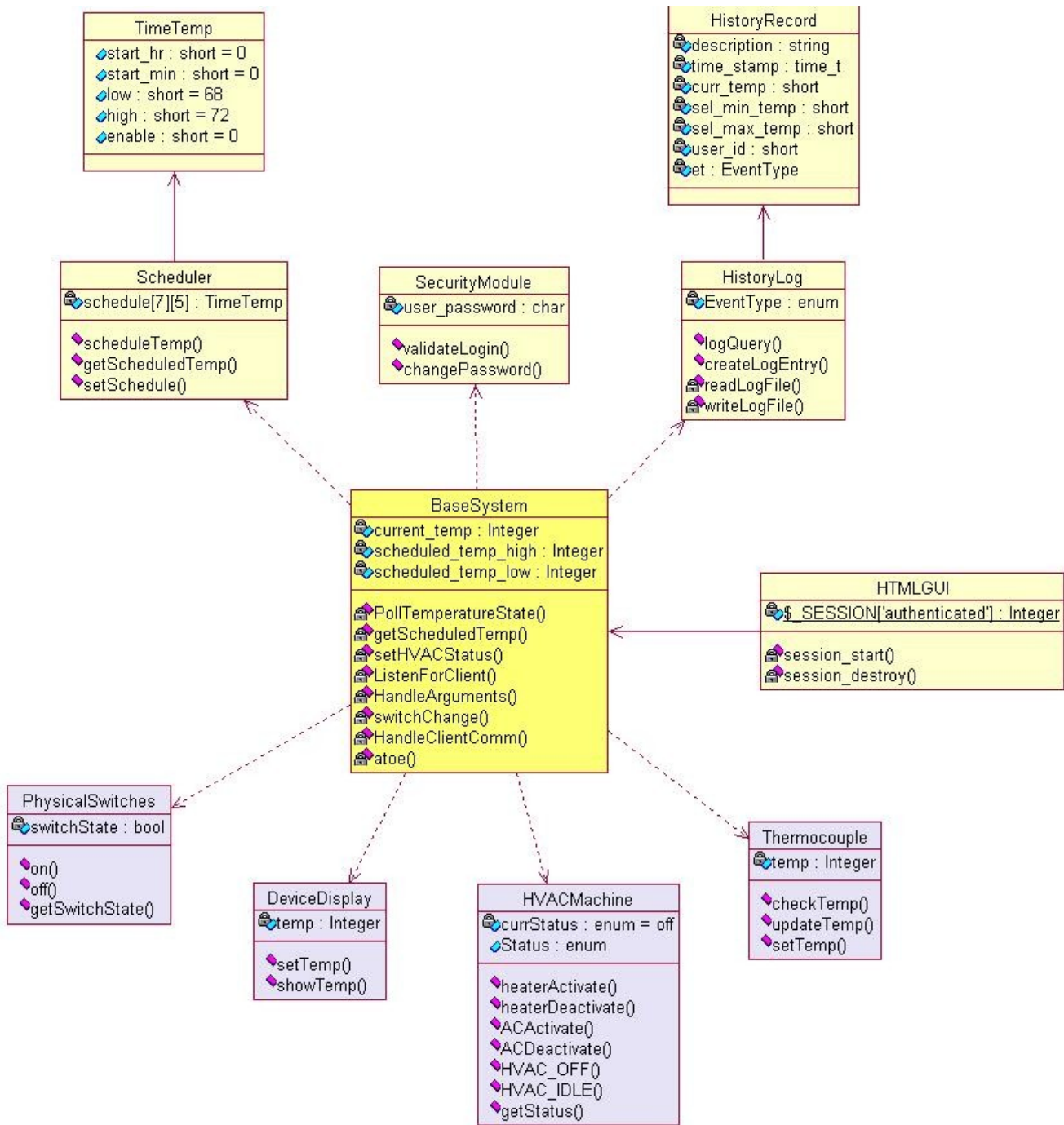
## 4 Traceability Matrix

The traceability matrix shows the relation of the various components in this design to stakeholder requests (user statements of need) and other features in the requirements document.

<b>Module</b>	<b>Traced-to</b>	<b>Traced-To Description</b>
BaseSystem		
DeviceDisplay	STRQ3	Stakeholder Request: Device Display
HVACMachine	FEAT3.6	Requirement: HVAC 'machine' Operation
PhysicalSwitches	STRQ3	Stakeholder Request: Device Switches
Thermocouple	FEAT3.1	Requirement: Thermocouple/Temperature Sensor Interaction
HistoryLog	STRQ1.1	Stakeholder Request: History
SecurityAuth	STRQ1.2	Stakeholder Request: Security
Scheduler	FEAT2.5	Requirement: Scheduler
TimeTemp		
HTMLGUI	STRQ1	Stakeholder Request: HTML Interface

# 5 Diagrams

## 5.1 UML Component Diagram



## 5.2 HVAC Machine State Diagram

