平成 23 年度　特別研究報告書

複数人家族向けの
ホームネットワークシステムの検討

龍谷大学　理工学部　情報メディア学科

学籍番号 T070442 中嶋　直史

指導教員　三好　力　教授

内容梗概

近年，家電製品の高性能・高機能化に伴い，それらをネットワーク化してより便利に使えるよう考えられているものが多い．これら家電製品の管理システムはホームネットワークシステムと呼ばれ，携帯電話で自宅の機器を操作できるものも多く，様々な企業から販売されている．

しかし，これら既存技術のホームネットワークシステムは二つの問題点を抱えていると考えられる．一つは対応した家電製品を所有していない場合は買い替える必要があり，安価に導入することができない費用面の問題である．もう一つは複数人家族の利用に際して起こりえる問題である．例えば外出している者が帰宅に合わせて携帯電話からエアコンを操作したとする．しかし，在宅者はエアコンがなぜ動作しているのか分からない．もし，消し忘れと判断され停止させられた場合は帰宅してくる者の欲求が満たされなかったことになる．これを防ぐには家族に連絡をすればよいが，それではホームネットワークシステムの利便性を否定することになってしまう．

本研究は導入費用を抑える方法の一つとして学習リモコンを利用した方法を提示し，履歴機能と人感センサーの導入による複数人家族での利便性の向上を検証した．

目次

第 1 章 はじめに

私たちの身の回りにはテレビや冷蔵庫，エアコンやパソコンといった具合に様々な家電製品が存在している．今や家電製品なくしては生活が成り立たないといえるだろう．毎年様々な新製品が発売され，以前の物より性能は良くなり，新しい機能を備えている物もある．より便利に使えるようにとメーカーが独自の家電製品規格を掲げて販売している物もある．同じメーカーのテレビとレコーダーを接続した場合にテレビのリモコン一つで両方の操作が可能といったものである．

インターネットが広く普及した現在においてはネットワーク対応型の家電製品も多くなってきている．テレビでインターネットを閲覧することも可能であり，ネットワークを通じて録画予約が可能な物もある．また，家電製品のホームネットワークサービスを販売する企業も増えている．これらのサービスの共通項目としてインターネットを利用した家電製品の制御があげられる．外出先から自宅の機器を操作することで帰宅時の環境をよりよいものにするためや，防犯を目的としたものがある．ホームネットワークサービスは非常に便利なものであるが導入には問題となる点もある．

問題点 1

対応家電製品でなければ利用できない点である．場合によっては機器そのものを取り換える必要があり，初期費用が高額となりやすい．

問題点 2

インターネットを通じて家電製品の状態が確認でき操作可能であったとしても，家電製品の周りの状況がわからないと操作ができない場合がある．在宅者が部屋が暖まったため直前まで使用していた暖房を，再度動作させてしまう可能性がある．在宅者は不快に感じるかもしれない．また，一人暮らしならば行えた消し忘れを防ぐ操作が複数人の家族では行いにくい．

解決方法として問題点 1 に対しては既存の製品をそのまま利用できる機器，あるいはシステムを構築するということが考えられる．これならば家電製品を買い替える必要もなく，費用も抑えることができる．問題点 2 に対しては操作履歴機能を搭載してなぜ動作しているのか判断可能にする，室内に人感センサーを設置して人がいるかどうか確認できるようにするなどいくつか方法が考えられる．

この二つの問題点の改善策として，学習リモコンと履歴機能，人感センサーを用いるシステムを提案する．既存家電製品の多くが赤外線リモコンによって操作されている点に注目した．学習リモコン一台で部屋すべての赤外線リモコン機器の制御を考えた．サーバーにはネットワーク操作用のページを設置しておき，アクセスがあった時のみ人感センサーに人の有無を要求し，操作履歴とともに表示する．これならば既存の家電製品の多くをそのまま利用することができる．また，人感センサーを利用し，操作履歴の閲覧が可能になることで複数人家族でも問題なく使用できると考えた．

第 2 章 ホームネットワーク

2.1 ホームネットワークとは

ホームネットワークとは暮らしをより快適なものにするための家電製品の自動実行，およびその管理システムのことをいう．家電ネットワークやホームオートメーションとも呼ばれる．身近にある物の例としてガス警報器や風呂の自動湯沸かしなどがある．家電製品単体での利用のほか，家電製品同士を直接ケーブルで接続して連係動作させるものやインターネットを通して利用できるものもある．特に AV 機器はメーカー独自の接続機能を提供している物が多く，対応機器同士を接続するだけで簡単に利用できるものも多い．例えばテレビとレコーダーを接続することでテレビのリモコンだけでレコーダーの基本的な操作を行うことができる．図 1 にホームネットワークの概要図を示す．

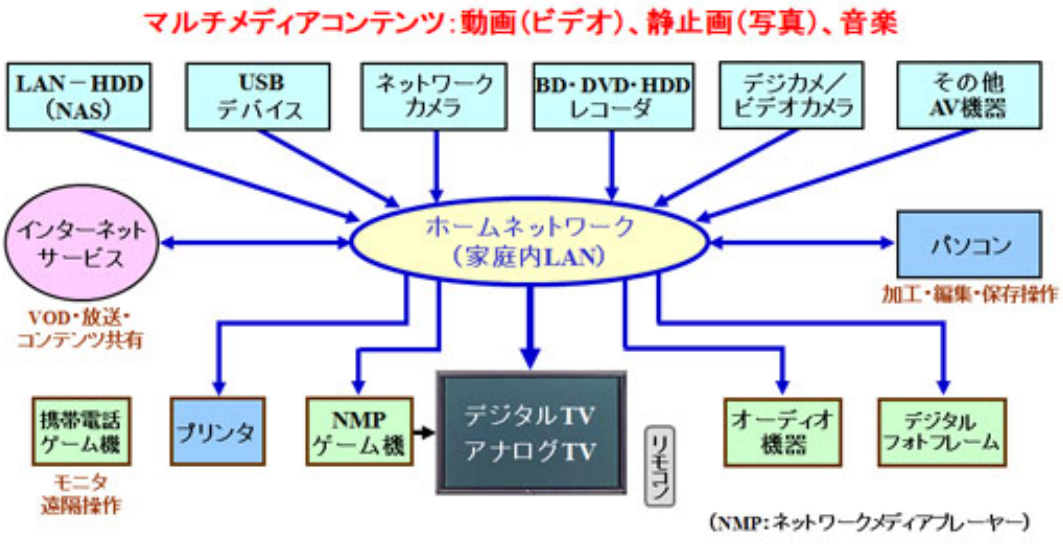

図 1 ホームネットワークの概要図 [1)]

2.2 ホームネットワークサービス

インターネットが広く普及した現在において，ホームネットワークとインターネットは密接につながっている．外出先から携帯電話を用いて自宅のホームネットワークシステムに接続し，家電製品を制御できるサービスが販売されている．販売されているサービスの例を以下にあげる．

・鍵の施錠および施錠確認
携帯電話を利用して玄関の鍵等の施錠確認および施錠を行うことができる．インターネットを経由しているので基本的に解錠操作はできない．

・インターホン
最近はカメラが搭載されている物もあり，室内から来訪者が誰なのか確認することが可能である．また，不在時に来訪者があった際には，登録されているメールアドレスに来訪者の写真を添付した通知メールを送り知らせてくれる．

・防犯カメラおよびセンサー
防犯カメラに動くものが映ればそれを検出し，携帯電話に動いたものの画像をメール送信する．また，警備会社に通報するものもある．不審な動きをする者を見分ける，置き去りにされた不審物や持ち去られた物を検出できる機能を持つものも存在する．

・遠隔操作サービス
　携帯電話から自宅の機器を遠隔操作することができる.
エアコンや床暖房を帰宅に合わせて動作させることで，快適な環境にしておくことができる．また，消し忘れを防ぐことができる.
照明の消し忘れを防止でき，留守時に点灯させることで防犯対策にもなる.

・光熱費計測
　計測機器を取り付けて光熱費の使用状況を視覚化して表示し省エネ意識を高める.

・ホームシアター
　設置したスクリーンやプロジェクター等をリモコンのボタン一つで連携動作させることで，映画の視聴に適した空間を作り出してくれる.

　また赤外線リモコンで操作する家電製品向けとして iRemocon が発売されている.

・iRemocon
　ネットワーク接続型の万能リモコンで操作は iPhone から行う．あらかじめ家電製品の信号を学習させておき，自宅に設置した本体に iPhone から命令を送り家電制御を行う．赤外線リモコン機器に限られるが比較的安価に導入ができる．図 2 に iRemocon の外観を示す.

図 2 iRemocon の外観 [2)]

2.3 ホームネットワーク関連技術

　ホームネットワークシステムを構築するに当たって利用される技術からいくつかを示す.

・HA 端子
「HA 端子とは日本電機工業会規格（JEM）で定められた下記の製品やその他の
機器のオンオフの制御とその状態モニタを行う端子である．オンオフ以外の制御およびモニタはこの端子ではできない．」とある [3)]．拡張アダプタを用いて無線化される場合もある.

・ECHONET
　家庭内の電力線を利用した通信規格である．電力線を利用するので配線工事は一切必要なくネットワークが構築できる．ただ，通信速度が遅いため，家電制御用のネットワークと考えられている．またゲートウェイを通して外部ネットワークからの操作も可能である．KNX の一部と競合している.

・Wi-Fi
　IEEE802.11a/b を利用する無線通信の相互接続性等を認められたことを表すブランド名であり，Wi-Fi Alliance が認定を行っている．Wi-Fi 自体が通信の規格というわけでない．近年ゲーム機に採用されたり，スマートフォンの普及などで耳にする機会が増えている.

・DLNA

異なるメーカー同士の機器接続を容易にするために作られた団体であり，一種の規格である．各製品が共通に対応すべきフォーマットや通信プロトコルを定めおり，家電寄りの部分がある．ゲーム機やテレビで利用可能な製品があり，パソコンを用いて家庭内 DLNA 環境を整えれば，パソコンに保存された動画をテレビで見ることができる．ちなみに DLNA は Digital Living Network Alliance の略称である．

2.4 ホームネットワークシステムの問題点

ホームネットワークについて述べてきたが，少なからず問題点も考えられる．

一つ目に対応機器でなければ利用することができず，場合によってはメーカーも統一する必要があるという点である．また，販売されている機器も高価であり，サービス利用料もわずかだがかかる．既存の家電製品をそのまま利用できるホームネットワークシステムが理想的である．二つ目に複数人の家族では問題が発生する可能性がある．例えば自宅にてエアコンを使用している者がいるとする．遠隔操作者からは消し忘れなのか判断がつかない可能性があり，エアコンを停止させた場合に在宅者は快適に過ごすことができない．

この二つの問題を改善するために「学習リモコンと LED 付送受信機」，「操作履歴機能」，「人感センサー」を導入したホームネットワークを考えた．既存家電製品の多くが赤外線リモコンによって操作されている点に注目し，様々な機器で使用されている別々のリモコンを学習リモコン一台でまかない，部屋すべての赤外線リモコン機器の制御を行うことを考えた．学習リモコンをサーバーに接続しネットワーク越しの操作も可能にする．また，対象機器に取り付けた LED 内臓受信機からサーバーへ信号情報を送り，履歴情報を作成する．ネットワーク越しの操作の場合は LED を点灯させるなどして見ただけで判断がつくようにする．合わせて人感センサーを設置しておくことで，履歴情報だけではわからなかった人の有無がわかり，複数人の家族でも問題なくホームネットワークシステムの運用ができると考えた．

第 3 章 提案手法

2.4 節で述べた問題点を改善する案として以下の三つの方法を考えた.

提案手法 1

学習リモコンと LED 付送受信機の利用

既存の家電製品の多くは赤外線リモコンを用いた操作になっている物が多い. この点を利用してホームネットワークシステムの家電制御に学習リモコンを用いる. あらかじめ部屋の全ての赤外線リモコン機器の赤外線コードを全て学習しておく. ネットワークを通した家電操作は学習リモコンから赤外線を送信して行う. また, 操作が確実に行われたことを確かめるために家電製品の赤外線受信部に赤外線受信機を取り付ける. この受信機は受信した内容をサーバーに送信し, 学習リモコンからの操作が正しく行われたかどうかの判断に用いる. さらに通常のリモコンでの操作も受信し, その内容を反映させる役目を持つ. 操作元がネットワークならば LED を光らせるなどして, 判断できるようにする.

提案手法 2

操作履歴機能の導入

既存技術では現在の機器の状態しか知ることができなかった. 一人暮らしならば使用するのは自分一人のため, 何も問題は無かった. 消し忘れた場合にも携帯電話から停止させることができた. しかし複数人家族ではそうはいかない. 機器を動作させたのは誰なのか, 消し忘れではないのかは動作させた者にしか分からない. このため帰宅に合わせて暖房を操作しても停止させられてしまう可能性があった. これを防ぐには動作開始時間や誰が動作させたのか分かればよい. 操作履歴機能を導入することで何時どういった操作を誰が行ったのか分かる.

提案手法 3

人感センサーの導入

既存技術では人がいるかどうかにかかわらずに機器を動作させていた. 在宅者から見れば突然機器が動き出すのであるから不快に思うかもしれない. 人感センサーを用いることで人がいる部屋の機器を動作させるべきか判断する材料になる. 人感センサーはエアコン搭載のものなどで利用可能であればそれを利用して費用を抑えることもできる.

提案手法 1 を導入することでホームネットワークシステムの導入は比較的安価になると考えられる. 提案手法 2, 3 の導入で複数人家族に対応したホームネットワークシステムが構築できると考えた. 図 3 に提案するホームネットワークシステムの全体図を示す.

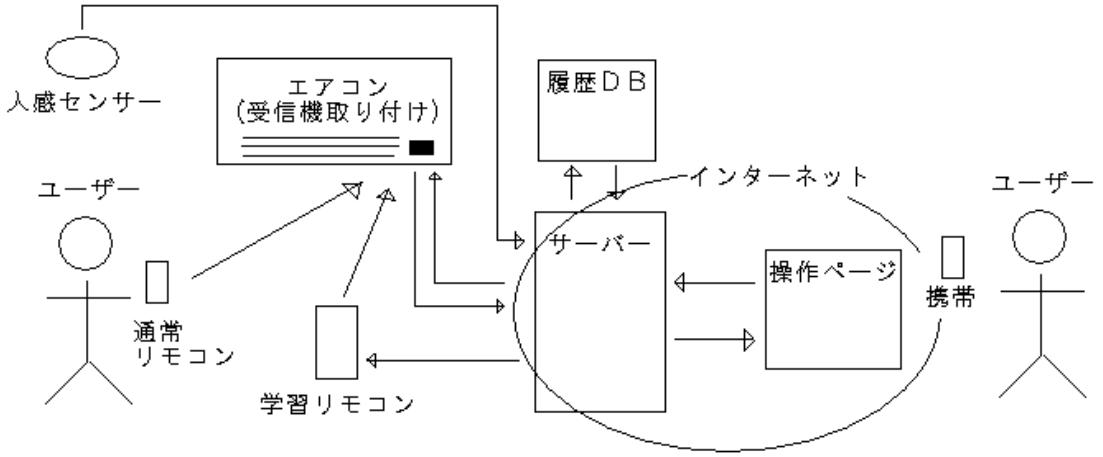

図 3 提案するホームネットワークシステムの全体図

第 4 章　実験と結果

4.1 実験目的

提案手法 1 で問題の一つである対応機器でなければ利用できない点はある程度解決できると考えた. そのため, 残る問題である複数人家族向けのホームネットワークシステムを実験で検証することとした. 既存のホームネットワークシステムに履歴機能, センサー機能を組み込むことで複数人家族に適したホームネットワークシステムを構築し, 既存システムと比較し, その効果を確かめる.

4.2 手順・実験方法

実際のホームネットワークシステムを構築することは費用の面など解決できない問題があるため, 仮想的なホームネットワークシステムのシミュレーターを構築し, その動作を検証した.
開発は Eclipse 3.7 Indigo と MySQL を用いて行った.

シミュレーターの概要

提案手法 1 をベースにサーバー, モニター, 操作対象(エアコン数台と玄関錠), ユーザー, 操作履歴データベースの各ユニットからなる. 図 4 に動作時の様子を示す. また, 各ユニットについて以下に説明する.

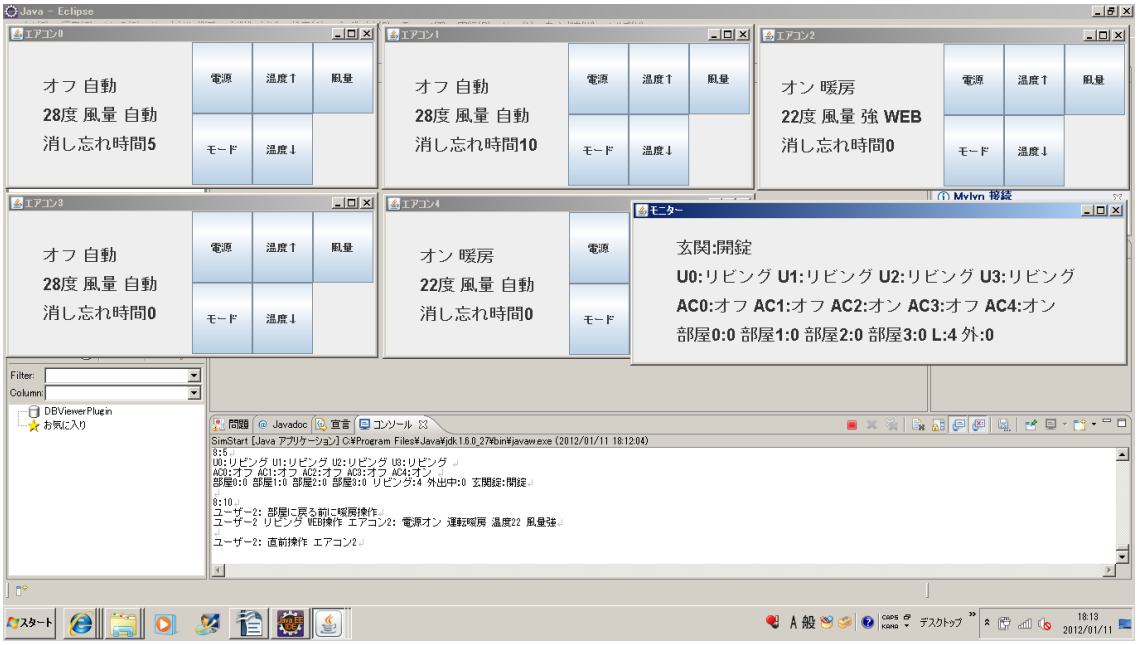

図 4 仮想ホームネットワークシステムの動作時の様子

サーバーユニット(学習リモコン，玄関錠及びシミュレーターシステムを内包)
・サーバーは WEB からの要求を受け，対象機器の状態，操作履歴，センサーの状況を伝える．
・サーバーは WEB からの操作を受け，学習リモコンに指示を送る．
・学習リモコンはそれにしたがって赤外線信号 A を送信する．
・その信号 A を受信してエアコンは動作する．同時にエアコンの赤外線信号受光部に取り付けた LED 内臓受信機でも信号 A を受信する．
・LED 内臓受信機は受信した信号 A をサーバーに返し，正しく操作が行われたことを伝える．同時に学習リモコンによる操作（ネットワーク操作）であることが判明するのでその印に LED を点灯させる．
・サーバーは信号 A と操作の種類(通常リモコンあるいはネットワーク操作)から操作履歴 DB に操作履歴を登録する．
・玄関錠は施錠および開錠の状態を持つ．
・ネットワーク操作では防犯のため施錠のみ可能である．

シミュレーターシステム
・図 5 に動作の流れを示す．
・実験者がユーザーの一人となって行動し，機器を操作することも可能である．行動はユーザーと同等に行うことができるが，行動順は一番遅い．
・実験者が参加する場合，履歴は MySQL を操作して取得する．センサー情報はモニターユニットに表示される．
・節電行動時間が設定されており，この時間間隔でユーザーは消し忘れのチェックを行う．
・ネットワーク操作考慮時間が設定されており，これをもとに消し忘れや操作の上書きを判断する．
・時間は家族全員の行動が終了すると一定時間進み，再度家族の行動終了を待つ．
・各時間の行動開始前に各ユーザーのいる部屋のエアコンの状況を取得する．
・家族全員の行動終了後，ネットワーク越しに各ユーザーのいる部屋のエアコンが操作されていないかチェックする．されていた場合はそのユーザーの不快回数を増加させる．
・ネットワーク操作の際に操作した人が操作した機器のある部屋に行く前に操作した機器の動作が停止させられた際は不快回数を増加させる．ただし，操作からネットワーク操作考慮時間以上経過していた場合はこの限りではない．
・シミュレーション終了時に各ユーザーの不快回数やエアコンのネットワーク操作失敗回数などを表示する．

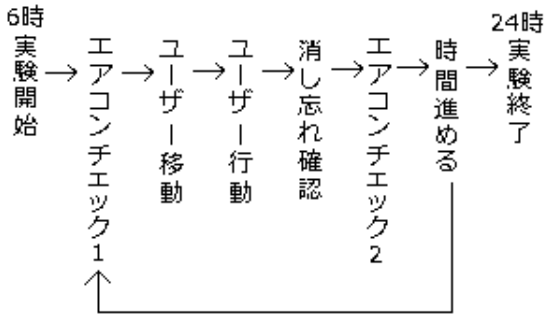

図 5 仮想ホームネットワークシステムの動作

モニターユニット
・図 6 に外観を示す.
・現在の時間や自宅状況を表示する
・Eclipse のコンソールにも出力されている.
・図 7 に実験者がユーザーの一人として参加したときの外観を示す.
・表示されたボタンで移動や玄関錠の操作ができる. 操作切り替えボタンで通常リモコンと携帯電話によるネットワーク操作を切り替える.

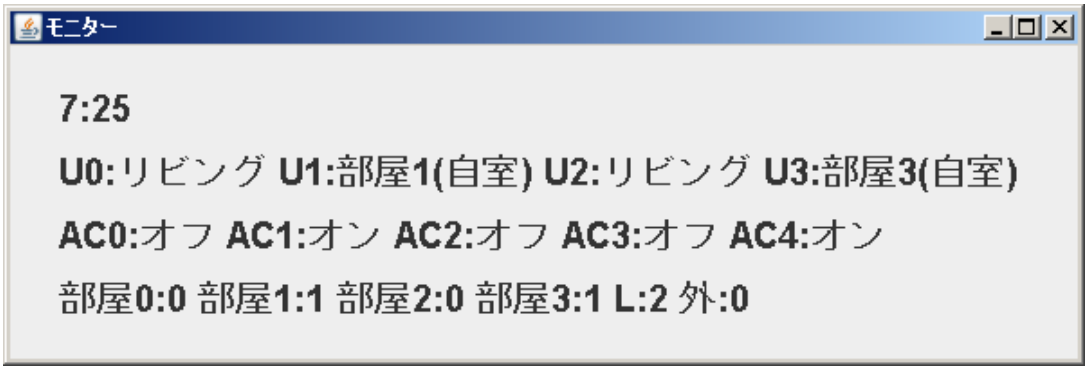

図 6 モニター外観

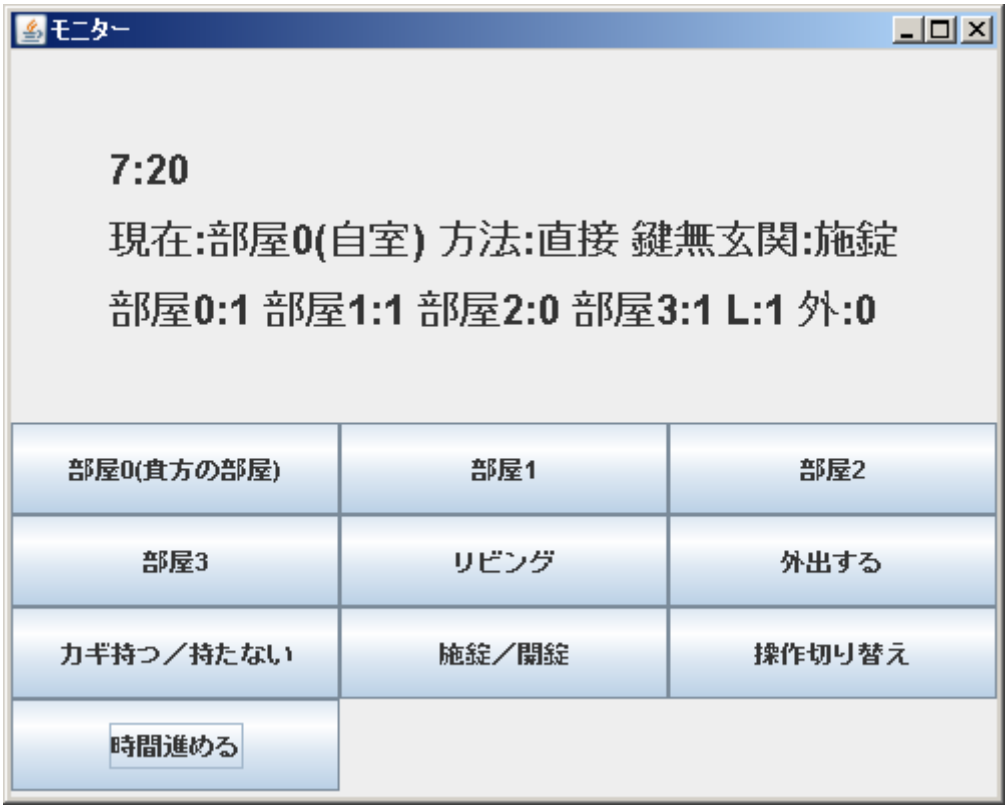

図 7 実験者参加時のモニター外観

エアコンユニット
・図 8 に外観を示す.
・基本的なエアコンの機能(電源, 運転モード, 温度上下, 風量)を持ち, 表示する.
・受光部近くに LED 内臓送受信機を持ち, エアコンが受信した信号をそのままサーバーに送信する.
・ネットワーク操作の場合は「WEB」と表示する.
・実験者が参加している場合に表示されたボタンで操作できる. ただし, 通常リモコンによる直接操作を選択しているのに別の部屋のエアコンを操作するといったことはできない.

図 8 実験ホームネットワークシステムのエアコン外観

ユーザーユニット
・シミュレーション開始時に 0 から 1 の範囲で注意力が与えられる.
・作成した行動表を読み込み, それに基づいて行動する.
・部屋を移動する際に誰もいなくなる場合はエアコンを停止させる.
・直接操作は対象機器のリモコンを用いて同じ部屋で行う. WEB 操作(ネットワーク操作)はユーザーのいる場所に関係なく行う.
・注意力が低いと行動を忘れることもある.
・約 1/3 の確率で外出を取りやめ, 外出しなかったときの行動をとる.
・一定感覚で誰か一人が消し忘れを考え, 直接あるいはネットワーク越しに消し忘れ(閉め忘れ)の可能性のあるものを止める(閉める).

ユーザーの行動表について
図 9 に例を示す. csv ファイル形式で作成されている. ユーザーユニットはこれを読み込んで行動する. 1, 2 列目は時間を示している. 3 列目は行動名を示している. 4 列目は操作する機器を示している. ”ac”はエアコンを, ”lock”は玄関錠を表している. また, ”0”以外は行動名の表示を行う. 5 列目は機器番号を示している. エアコンは”0”が自室を, ”100”はリビングを表す. 玄関錠は”0”のみである. 6 列目から 9 列目は機器により内容が異なる. エアコンの場合は電源, 運転モード, 温度, 風量を表す. 玄関錠は施錠状態と鍵を持つかどうか表している. ”0”は鍵を持たない, ”1”は鍵を持つ, ”2”は自宅に誰もいなくなるなら鍵を持つ. 10 列目は後に拡張できるよう空欄としている. 11 列目は操作方法を示している. ”直接”は通常のリモコンを使用した操作, ”WEB”はネットワーク操作を表している. 12 列目は場所を示している. ”0”は自室を, ”100”はリビングを, ”101”は外出中を表している. 13 列目は行動の種類を表している. ”0”は通常の行動, ”1”は忘れる可能性のある行動, ”2”は外出するかどうか判定する行動, ”3”は外出しなかった場合の行動を表している. ”4”は睡眠起床スイッチを表し, 睡眠起床状態を入れ替える. 睡眠中は消し忘れを心配することはない. 外出関連は自宅内の場所で”0”の行動の際に通常の行動に戻る.

14	0	起床		1	0	0	0	0	0	0	0	4
14	5	0	0	0	0	0	0	0	0	0	0	0
14	10	暖房操作	ac	0	オフ	0	0	0	0	直接	0	1
14	15	外出	lock	0	開錠	1	0	0	0	直接	101	2
14	20	外出施錠	lock	0	施錠	0	0	0	0	直接	101	0
14	25	0	0	0	0	0	0	0	0	0	100	3
14	30	暖房操作	ac	100	オン	暖房	21	自動	0	直接	100	3
14	35	0	0	0	0	0	0	0	0	0	100	3
14	40	0	0	0	0	0	0	0	0	0	100	3
14	45	0	0	0	0	0	0	0	0	0	100	3
14	50	0	0	0	0	0	0	0	0	0	100	3
14	55	0	0	0	0	0	0	0	0	0	100	3
15	0	0	0	0	0	0	0	0	0	0	100	3
15	5	移動先暖房操作	ac	100	オン	暖房	21	自動	0	WEB	101	0
15	10	0	0	0	0	0	0	0	0	0	100	3
15	15	帰宅		1	0	0	0	0	0	0	101	0
15	20	0	0	0	0	0	0	0	0	0	100	0
15	25	暖房操作	ac	100	オン	暖房	21	強	0	直接	100	0
15	30	0	0	0	0	0	0	0	0	0	100	0

図 9 ユーザーの行動表の例

操作履歴データベース

・MySQL を利用している.

・操作機器ごとに記録する.

・記録内容は操作時間, 操作内容, 操作方法である. 操作方法は通常リモコンによる場合は直接操作と記録する. ネットワーク操作の場合は端末情報などから誰が操作したのか分かるように記録する.

各機能の効果を比較確認するため，実験は以下の四つの条件で行った．なお実験 2 から実験 4 は学習リモコンとLED 付送受信機を利用しているものとする．

実験1：既存技術と同等(履歴機能・人感センサー無し)
・操作対象のある部屋の状況を全く考慮せずにユーザーが操作を実行する．
・在宅者は操作対象の部屋に人がいなければ消し忘れとして停止させる．
・外出者は消し忘れの判断ができないため，動作していても放置する．
・外出者は防犯のため，自分が鍵を持っている時に限り施錠する．

実験2：履歴機能有・センサー機能無
・在宅者は履歴を参照して消し忘れか判断する．
・在宅者は部屋に誰もいなくなる場合でもネットワーク操作で動作している機器は停止させない．ただし，最後の操作からネットワーク操作考慮時間以上経過している場合はその限りではない．
・外出者は操作対象のある部屋の状況を全く考慮せずに操作を実行する．
・外出者は履歴の間隔から消し忘れの可能性を考える．履歴二つの時間間隔の 2 倍と節電行動時間を足し合わせた時間以上最後の操作から経過していた場合は，もう移動してしまった可能性を考慮して停止させる．
・外出者は防犯のため，履歴を参照し自分が鍵を持っている時に限り施錠する．

実験3：履歴機能無・センサー機能有
・在宅者はLED 送受信機の LED から消し忘れか判断する．
・在宅者は部屋に誰もいなくなる場合でもネットワーク操作で動作している機器は停止させない．
・ネットワーク操作はセンサーの情報を利用し人がいない場合に操作を実行する．
・外出者はセンサーの情報を利用し消し忘れを防ぐ．
・外出者は防犯のため，センサーの情報から誰もいないと分かり，自分が鍵を持っている時に限り施錠する．

実験4：履歴機能有・センサー機能有
・在宅者は履歴を参照して消し忘れか判断する．
・在宅者は部屋に誰もいなくなる場合でもネットワーク操作で動作している機器は停止させない．ただし，最後の操作からネットワーク操作考慮時間以上経過している場合はその限りではない．
・ネットワーク操作はセンサーの情報を利用し人がいない場合に操作を実行する．
・外出者は履歴とセンサーの情報を利用し消し忘れか判断する．
・外出者は防犯のため，開錠から一定時間以降にセンサーの情報から誰もいないと分かり，自分が鍵を持っている時に限り施錠する．

共通条件は以下である．
・家族構成は四人，実験者は参加しない．
・注意力は固定する．
・部屋の数はユーザーの各部屋とリビング．
・行動は決められているが四人ともそれぞれ異なる．
・6 時から 24 時までの生活シミュレーションを行う．行動間隔は五分毎である．
・節電行動時間は 15 分とする，この時間ごとに誰かが消し忘れや施錠状態を心配することがあり，消し忘れと疑わしきものを条件に従って停止させたり，鍵を施錠する．
・ネットワーク操作考慮時間は 30 分とする．ネットワーク操作から 30 分以上経過したと判断できる場合は操作を上書きする．

これらの条件のもと，家族が快適に過ごせるのはどの提案手法を採用したものなのか不快回数とネットワーク操作失敗回数を比較することで得られると考え，実験を行った．

4.3 結果

実験1:既存技術と同等(履歴機能・人感センサー無し)

表 1 と表 2 に結果を示す. ユーザーに与えた不快回数が最も多い結果となった. また, ネットワーク操作が上書きされてしまった回数も最も多かった. これは状況を考えずに操作を行ったり, 機器を停止させているためである. 一人暮らしには問題ないが複数人の家族には適さないことがわかる.

表 1 実験 1 のユーザー不快回数

ユーザー	不快回数
0	3
1	4
2	5
3	3

表 2 実験 1 のネットワーク操作失敗回数

エアコン	失敗回数
0	2
1	0
2	3
3	2
4	1

実験2:履歴機能有・センサー機能無

表 3 と表 4 に結果を示す. 不快回数は実験 1 と比べて減少した. これはネットワーク操作で動作している機器が停止させられることがなくなったためであると考えられる. この不快感は在宅者が感じているもので, ネットワーク上から機器を操作されたためである. また, 外出者も履歴から消し忘れを考えて機器を操作しているためでもある. 在宅者は消し忘れの判断などに利用でき, 外出者はネットワーク操作がある程度保障される. しかし, 履歴機能単体では快適な生活を送るのに不十分である.

表 3 実験 2 のユーザー不快回数

ユーザー	不快回数
0	1
1	5
2	2
3	2

表 4 実験 2 のネットワーク操作失敗回数

エアコン	失敗回数
0	0
1	0
2	0
3	0
4	0

実験3:履歴機能無・センサー機能有

表 5 と表 6 に結果を示す. 不快回数は実験 2 よりさらに減少した. 注目すべきはその回数で, ネットワーク操作が停止させられた回数と等しいことである. 外出者は不快感を感じているが, 在宅者に対して不快感を与えずに機器を操作できている. 人がいるのかどうか判断できるだけで在宅者は外出者から影響を受けずに快適に過ごすことができる. しかし, 外出者の操作が別の外出者の操作によって停止させられてしまっている. これはセンサーに人の反応がないため, 消し忘れと判断されているためである. 外出者も動作の種類を判断できる必要がある.

表 5 実験 3 のユーザー不快回数

ユーザー	不快回数
0	2
1	1
2	1
3	1

表 6 実験 3 のネットワーク操作失敗回数

エアコン	失敗回数
0	1
1	0
2	1
3	1
4	2

実験4:履歴機能有・センサー機能有

表 7 と表 8 に結果を示す. これまでの実験と比べてユーザーに全く不快感を与えずに一日を終えている. これは実験 2 の操作履歴機能があることによる減少と, 実験 3 の人感センサーがあることによる減少が合わさった効果だといえる. 結果として外出者の操作を考え, 人感センサーがあることで在宅者の状況を考えて行動を行うことができている.

表 7 実験 4 のユーザー不快回数

ユーザー	不快回数
0	0
1	0
2	0
3	0

表 8 実験 4 のネットワーク操作失敗回数

エアコン	失敗回数
0	0
1	0
2	0
3	0
4	0

第 4 章 考察

実験 1 では履歴機能及びセンサー機能を持たない既存技術のため，操作対象のある部屋の状況を考慮せずに操作を行っている．このため部屋にいる者に多大な迷惑をかけてしまっている．また，なぜ動作しているのかも不明なため，ネットワーク上からは使用しているのか消し忘れているのかの判断はつけられない．同様にネットワーク操作で動作している場合でも在宅者が消し忘れと判断した場合は停止させてしまう．在宅者と外出者双方の快適な利用に向かないことが分かる．このことから操作対象のある部屋の状況を少しでも知る方法があれば便利だと分かる．

実験 2 では操作履歴機能を利用してどういった操作がいつ行われたか知ることができるようになっている．在宅者は操作方法によって対象の動作を止めるべきか判断することができる．ネットワーク操作においては直前の操作とその一つ前の操作の間隔を元に誰かいたとしても移動してしまっているのではないかと判断して動作を停止させている．勿論使用されている機器ををを止めてしまうこともある．ただネットワーク操作による動作も履歴から判別できるため，外出者が操作した機器を停止させてしまうことはなくなっている．消し忘れを防止することよりもなぜついているかの理由を知るための手段として役に立つと考えられる．ただし人がいるかどうかは実験 1 と同様に分からないため，考慮せずに操作を行っている．このためネットワーク操作による不快感を与えてしまった回数も少し減少した程度で，履歴機能だけではあまり快適ではないことが分かる．

実験 3 ではセンサー機能を利用して操作対象のある部屋に人がいるのかどうか判断できるようになっている．人がいるかどうか分かるため，在宅者に不快感を与えずに操作ができている．消し忘れの防止については在宅者は操作対象の LED 内臓送受信機の LED の状態で直前の動作がネットワーク操作か判断できるため問題ないが，ネットワーク上からでは判断できない．このため，誰かが行ったネットワーク操作で動作している機器を外出者が消し忘れと判断してしまい，ネットワーク操作で停止させてしまっている．在宅者は快適に過ごすことができるが，外出者が快適に利用できない可能性がある．

実験 4 では履歴機能とセンサー機能を併用して使用している．実験 2 の結果ではなぜ動作しているのか在宅者も外出者も分かるため，ある程度操作が約束されていた．ただし，人の有無が分からずに不快感を与えてしまっていた．この問題は人感センサーの導入によって解決された．実験 3 の結果ではネットワーク上の操作において，在宅者に不快感を与えずに操作ができていた．ただし，操作の種類が分からないため，外出者のネットワーク操作が別の外出者のネットワーク操作で人がいないことから消し忘れと判断されて停止させられていた．この問題は履歴機能の導入で解決された．実験 2 と実験 3 の弱点を補い合う形となっている．

各実験の結果から操作履歴機能とセンサー機能を併用して使用することが最も効果が高い事が分かる．

また実験の結果としては考慮していないが玄関錠については考える必要がある．実験全体を通して何度か閉めだされることが起こっていた．家族がいた場合は自宅に入ることができているが，電気錠は現在の仕組みのままでは複数人家族にあまり適さないことがわかる．ただ実験において玄関錠の扱いが極端なことは否めない．しかし鍵を持たずに外出していたものが閉めだされてしまう可能性は否定できない．仮にセンサーで自宅内に人がいないことが分かり，ネットワーク操作で施錠したとする．その時に誰かが一般的に鍵を持つ必要のないような用事で家の外に，例えば庭いじりをするために庭にいた場合には閉めだされてしまう．携帯電話を持っていたとしても開錠は行えないため，鍵を持つ家族に連絡し帰りを待つこととなる．このような状況が起こらないとは限らない．これを防ぐために玄関錠を施錠する際には家族全員の操作を必要とするなど方法を工夫する必要がある．

第 5 章 結論

本研究では学習リモコンとLED付送受信機，操作履歴機能，人感センサーの利用といった三つの手法から既存のホームネットワークシステムを複数人家族に対応させることを試みた．各実験の結果から操作履歴機能と人感センサーを併用することがもっとも複数人家族に適していることがわかった．この併用型のシステムならば在宅者と外出者双方が相手のことを考えながら操作を行うことができる．

ただし，消し忘れを防ぐ行動は現実とはかけ離れている部分があったことは否めない．消し忘れと判断され停止させられた機器の中にはネットワーク操作による動作のものもあった．これによる不快回数の増加もあったと考えられる．

今後の課題としてプログラムを改良し，提案手法以外の方法についても検討したい．また，操作履歴機能や人感センサーを搭載しているのでこれらを連携して省エネ効果を発揮するシステムも検討したいと思う．

謝辞

本研究を行うにあたり，ご指導を頂いた卒業論文指導教員の三好力教授に感謝いたします．また，様々な助言を下さった三好研究室の皆様に感謝します．

参考文献

1) Lanhome（ランホーム）:ホームネットワーク（家庭内 LAN）によるデジタルホーム構築
http://www.lanhome.co.jp/

2) iRemoconトップページ
http://i-remocon.com/

3) HA 端子　JEM-A 端子　JEMA 端子　JEM1427 端子
http://www.mtrx.jp/INFORMATION/JEMA.htm

4）ホーム IT システム FEMINITY | 東芝ライテック(株)
http://feminity.toshiba.co.jp/feminity/

5）東京ガスホームオートメーション:Remote＋（リモートプラス）
http://home.tokyo-gas.co.jp/tes/remote/

付録 ソースコード

SimStart.java

```
public class SimStart {
	public static void main(String[] args) {
		HsStatus.TH = 6;//開始時刻 時
		HsStatus.TM = 0;//開始時刻 分
		HsStatus.TimeH = HsStatus.TH;
		HsStatus.TimeM = HsStatus.TM;
		HsStatus.ACnum = 5;//部屋数(エアコン台数) 3以上
共有一部屋のためユーザー数は-1;
		HsStatus.Living = 100;//リビング
		HsStatus.Out = 101;//外
		//節電行動時間
		HsStatus.Setuden = 15;
		//ネットワーク操作考慮時間
		HsStatus.WebSetuden = 30;
		//操作履歴機能と人感センサーの有無 true 有
		HsStatus.history = false;
		HsStatus.Psensor = false;
		//スレッドスリープ時間
		HsStatus.sleep = 1500;
		//モニターなどを表示する true
		HsStatus.vis = true;
		HsStatus.monitor = true;
		//CPUユーザーのみ true
		HsStatus.CPUonly = false;

		if(HsStatus.CPUonly == true){
			HsStatus.CPUNoStart = 0;
			}
		else{
			HsStatus.CPUNoStart = 1;
		}
		HsStatus.Details = new String[10];//sql項目数
		Sensor.RoomNum = new int[HsStatus.ACnum + 1];//
部屋の数と外出先を含めた人数格納用
		//ホームサーバー起動
		HomeServer HS = new HomeServer();
		Thread THS = new Thread(HS);
		THS.start();
		}
	}
```

HomeServer.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

//ホームサーバー情報
class HsStatus {
	//操作データ
	static String WebData;
	static String Details[];
	static String Sensor[];
	static int ReturnCheck = 1;
	//時間関係
	static int TH;
	static int TM;
	static int TimeH;
	static int TimeM;
	// エアコンの台数
	static int ACnum;
	//CPU ユーザーの開始番号
	static int CPUNoStart;
	//操作履歴 DB(MySQL)接続用
	static String Driver = "com.mysql.jdbc.Driver";
	static String DB = "jdbc:mysql://localhost/home";
	static String Master = "root";
	static String Pass = "root";
	//リビングと外出先の番号
	static int Living;
	static int Out;
	//玄関錠について
	static String Lock = "施錠";
	static boolean LockB = true;//true 施錠
	static int EveryTime = 5;//時間の区切り 5 分ごとに行動
	static boolean history;// true 履歴有
	static boolean Psensor;// true センサー有
	//節電行動時間
	static int Setuden = 0;
	//ネットワーク操作考慮時間
	static int WebSetuden = 0;
	//スレッド休止時間
	static int sleep;
	//消し忘れフラグ
	static boolean Keshi = false;
	//コンピュータのみかどうか
	static boolean CPUonly = false;
	//消し忘れ考慮フラグ
	static boolean FtE = false;
	//ユーザー現在位置エアコンチェックフラグ
	static boolean UPC = false;
	//状態表示フラグ
	static boolean comp = false;
	//モニターなどのユニットを表示するかどうか
	static boolean monitor = false;
	static boolean vis = false;
}
//ユーザー現在位置記憶部分
class UserEnv {
	static String ACD[];
	static String ACU[];
	static String ACO[];
	static String ACF[];
}
//センサー部分
class Sensor {
	static int[] RoomNum;

	synchronized static void plus(int x) {
		RoomNum[x]++;
	}
	synchronized static void minus(int x) {
		RoomNum[x]--;
	}
}
public class HomeServer extends Thread {
	static AirCon[] AirCon = new AirCon[HsStatus.ACnum];
	Thread[] TAE = new Thread[HsStatus.ACnum];
	static User[] User = new User[HsStatus.ACnum];
	Thread[] TU = new Thread[HsStatus.ACnum];

	//節電行動チェック
	synchronized static void FtE(boolean x) {
		HsStatus.FtE = x;
	}
	synchronized static boolean FtECheck() {
		return HsStatus.FtE;
	}
	public void run() {
		UserEnv.ACD = new String[HsStatus.ACnum - 1];
		UserEnv.ACU = new String[HsStatus.ACnum - 1];
		UserEnv.ACO = new String[HsStatus.ACnum - 1];
		UserEnv.ACF = new String[HsStatus.ACnum - 1];
		HsStatus.comp = false;
		double brain;
		HsStatus.Details = new String[10];
		HsStatus.Sensor = new String[HsStatus.ACnum + 1];
		System.out.println("履歴機能: " + HsStatus.history + "
センサー: " + HsStatus.Psensor);
		System.out.println("メンバー");
		for (int x = 0; x < HsStatus.ACnum; x++) {
			AirCon[x] = new AirCon();
			AirCon[x].id = x;
			if (HsStatus.vis == true) {
				TAE[x] = new
Thread(AirCon[x]);
				TAE[x].start();
			}
		}
		if(HsStatus.CPUonly == false){
			System.out.println("実験者:ユーザー 0");
			User[0] = new User();
			User[0].name = "実験者";
			User[0].place = 0;
			User[0].placename = "部屋 0(自室)";
			Sensor.plus(0);
		}
		for(int x = HsStatus.CPUNoStart;x < HsStatus.ACnum
-1;x++){
			User[x] = new User();
			User[x].SetInfo(x);
			TU[x] = new Thread(User[x]);
			Sensor.plus(x);
			for (brain = 0; brain < 0.5 || brain > 0.9;) {
				brain = Math.random();
			}
			//注意力固定のため
			if (x == 0) {
				brain = 0.6505995828321334;
			}
			if (x == 1) {
				brain = 0.8609175542414864;
			}
			if (x == 2) {
				brain = 0.7426643080105362;
			}
			if (x == 3) {
				brain = 0.7776786239145319;
			}
			User[x].SetMemory(brain);
			System.out.println("CPU:ユーザー" + x + "
注意度:" + brain);
			TU[x].start();
		}
		if (HsStatus.monitor == true) {
			Monitor Moni = new Monitor();
			Thread TMoni = new Thread(Moni);
			TMoni.start();
		}
		int k = activeCount();
		System.out.println("スレッド数:" + k);
		System.out.println("節電喚起時間: " +
HsStatus.Setuden + "分");
```

```
                        System.out.println("インターネット操作考慮時間: " +
HsStatus.WebSetuden + "分");
                        System.out.println("サーバー起動 シュミレーション
開始");
                        System.out.println();
                        System.out.println(HsStatus.TimeH + ":" +
HsStatus.TimeM);
                        //ユーザー環境チェック終了人数
                        int uacc = 0;
                        //ユーザー行動完了人数
                        int uacomp = 0;

                        while (true) {
                                if (HsStatus.CPUonly == true) {
                                        try {

        Thread.sleep(HsStatus.sleep);
                                                Master.FA = true;
                                        }
                                        catch (Exception e) {
                                                System.err.println("
エラー" + e.getMessage());
                                        }
                                }
                                if (HsStatus.Keshi == false) {
                                        for (int x = 0; x <
HsStatus.ACnum; x++) {
                                                if (AirCon[x].direct
== true && AirCon[x].DengenPresent == 1 && Sensor.RoomNum[x] < 1) {

        AirCon[x].loss = AirCon[x].loss + HsStatus.EveryTime;

        System.out.println("消し忘れ発生中:エアコン" + AirCon[x].id);
                                                }
                                                else if
(AirCon[x].direct == false && AirCon[x].DengenPresent == 1 &&
AirCon[x].LastUse == 100 && Sensor.RoomNum[x] < 1) {

        AirCon[x].loss = AirCon[x].loss + HsStatus.EveryTime;

        System.out.println("消し忘れ発生中:エアコン" + AirCon[x].id);
                                                }
                                        }
                                        HsStatus.Keshi = true;
                                }
                                if (HsStatus.UPC == false) {
                                        for (int x = 0; x <
HsStatus.ACnum - 1; x++) {
                                                if (User[x].place !=
HsStatus.ACnum) {

        UserEnv.ACD[x] = HomeServer.AirCon[User[x].place].Dengen;

        UserEnv.ACU[x] = HomeServer.AirCon[User[x].place].Unten;

        UserEnv.ACO[x] = HomeServer.AirCon[User[x].place].Ondo;

        UserEnv.ACF[x] = HomeServer.AirCon[User[x].place].Furyo;
                                                }
                                        }
                                }
                                HsStatus.UPC = true;
                                for (int x = 0; x < HsStatus.ACnum - 1; x+
+) {
                                        if (HsStatus.CPUonly == false
&& x == 0) {
                                                continue;
                                        }
                                        if (User[x].BR == true) {
                                                uacc++;
                                        }
                                }
                                if (((uacc == HsStatus.ACnum - 2 &&
HsStatus.CPUonly == false) || (uacc == HsStatus.ACnum - 1 &&
HsStatus.CPUonly == true)) && Master.FA == true) {
                                        for (int x = 0; x <
HsStatus.ACnum - 1; x++) {
                                                if (User[x].move ==
false && User[x].place != HsStatus.ACnum) {
                                                        if
(UserEnv.ACD[x].equals(HomeServer.AirCon[User[x].place].Dengen)) {
                                                        }
                                                        else {

        User[x].dis = true;
                                                        }
                                                        if
(UserEnv.ACD[x].equals("オン")) {

        if
(UserEnv.ACU[x].equals(HomeServer.AirCon[User[x].place].Unten)) {


        }

        else {

                User[x].dis = true;
        }

        if
(UserEnv.ACO[x].equals(HomeServer.AirCon[User[x].place].Ondo)) {


        }

        else {

                User[x].dis = true;

        }

        if
(UserEnv.ACF[x].equals(HomeServer.AirCon[User[x].place].Furyo)) {


        }

        else {

                User[x].dis = true;

        }                                                       }
                                                                for (int y
= 0; y < HsStatus.ACnum - 1; y++) {

        if (HsStatus.CPUonly == false && y == 0) {

                continue;

        }

        if (y != x && User[y].place == User[x].place &&
User[y].UA[3].equals("ac")                                      &&
User[y].UA[4].equals(Integer.toString(User[x].place))) {

                User[x].dis = false;

        }                                                       }
                                                                for (int y
= 0; y < HsStatus.ACnum - 1; y++) {

        if (HsStatus.CPUonly == false && y == 0) {

                continue;

        }

        if (y != x && User[y].place != User[x].place &&
User[y].UA[3].equals("ac")                                      &&
User[y].UA[4].equals(Integer.toString(User[x].place)) && HsStatus.Psensor
== false) {

                User[x].dis = true;

        }                                                       }
                                                                if
(HsStatus.CPUonly == false && User[0].place == User[x].place &&
Master.ACFA == true) {

        User[x].dis = false;                                    }
                                                                if
(HomeServer.AirCon[User[x].place].direct == true) {

        User[x].dis = false;                                    }
                                                                if
(User[x].dis == true) {

        User[x].displeasure();

        User[x].dis = false;

        System.out.println(User[x].name + ": 今 い る " +
User[x].placename + "のエアコンが WEB から操作された。");

        System.out.println(User[x].name + ": 不 快 回 数 " +
User[x].displeasure);

        System.out.println("前:" + UserEnv.ACD[x] + UserEnv.ACU[x] +
UserEnv.ACO[x] + UserEnv.ACF[x]);

        System.out.println("後 :" +
HomeServer.AirCon[User[x].place].Dengen                          +
HomeServer.AirCon[User[x].place].Unten                           +
HomeServer.AirCon[User[x].place].Ondo                            +
HomeServer.AirCon[User[x].place].Furyo);

        System.out.println();
                                                                }
                                                        }
                                                        uacomp++;
```

```
                        }
                    }
                    else {
                        uacc = 0;
                    }
                    if ((uacomp == HsStatus.ACnum - 1 &&
HsStatus.CPUonly == false) || (uacomp == HsStatus.ACnum - 1 &&
HsStatus.CPUonly == true)) {
                        if (HsStatus.comp == false) {
                            for (int x = 0; x <
HsStatus.ACnum - 1; x++) {
                                if
(HsStatus.CPUonly == false && x == 0) {
        System.out.print("実験者:" + User[x].placename + " ");
                                }
                                else{
        System.out.print("U" + x + ":" + User[x].placename + " ");
                                }
                            }

        System.out.println();
                            for (int x = 0; x <
HsStatus.ACnum; x++) {
        System.out.print("AC" + x + ":" + AirCon[x].Dengen + " ");
                            }

        System.out.println();
                            for (int x = 0; x <
HsStatus.ACnum + 1; x++) {
                                if (x ==
HsStatus.ACnum - 1) {
        System.out.print("リビング:" + Sensor.RoomNum[x] + " ");
                                }
                                else if
(x == HsStatus.ACnum) {
        System.out.println("外出中:" + Sensor.RoomNum[x] + " 玄関錠:"
+ HsStatus.Lock);
                                }
                                else {
        System.out.print("部屋" + x + ":" + Sensor.RoomNum[x] + " ");
                                }
                            }

        System.out.println();
                            HsStatus.comp    =
true;
                        }
                        HsStatus.UPC = false;
                        HsStatus.TimeM                =
HsStatus.TimeM + HsStatus.EveryTime;
                        if (HsStatus.TimeM == 60) {
                            HsStatus.TimeM  =
0;
                            HsStatus.TimeH  =
HsStatus.TimeH + 1;
                        }

        System.out.println(HsStatus.TimeH + ":" + HsStatus.TimeM);
                        if (HsStatus.TimeH == 24) {

        System.out.println("1 日の終わり。シュミレーションを終了しま
す。");
                            System.out.println("
結果");
                            System.out.println("
ユーザー名: ユーザー不快回数 | 閉めだされた時間");
                            for (int x = 0; x <
HsStatus.ACnum - 1; x++) {
                                if
(HsStatus.CPUonly == false && x == 0) {
        System.out.println(" 実 験 者 : " + User[x].displeasure + " | " +
User[x].shimedashi);
                                }
                                else{

        System.out.println(" ユ ー ザ ー " + User[x].id + ": " +
User[x].displeasure + " | " + User[x].shimedashi);
                                }
                            }
                            System.out.println("
結果:エアコンの消し忘れ時間|ネットワーク操作失敗回数");
                            for (int x = 0; x <
HsStatus.ACnum; x++) {
        System.out.println("エアコン" + AirCon[x].id + ": " + AirCon[x].loss
+ " | " + AirCon[x].webcancel);
                            }
                            System.exit(0);
                        }
                        Master.ACFA = false;
                        Master.FA = false;
                        HsStatus.Keshi = false;
                        HsStatus.FtE = false;
                        HsStatus.comp = false;
                        uacomp = 0;
                    }
                }
            }

            synchronized static void Registration(String A, String B, String C,
String D, String E, String F, String G, String H, String I, int id) {

                Connection con = null;
                String HH = null;
                String MM = null;
                String sql = "";

                // Details
                // ユーザー名,機器,機器番号,内容 1,内容 2,内容 3,内容
4,内容 5
                // 0 1 2 3 4 5 6 7
                //          HomeServer.Registration("          直
接","ac",Integer.toString(x),"オフ","0","0","0",name,placename,id);

                HsStatus.Details[0] = A;
                HsStatus.Details[1] = B;
                HsStatus.Details[2] = C;
                HsStatus.Details[3] = D;
                HsStatus.Details[4] = E;
                HsStatus.Details[5] = F;
                HsStatus.Details[6] = G;
                HsStatus.Details[7] = H;
                HsStatus.Details[8] = I;

                if (HsStatus.Details[1].equals("ac")) {
                    if (HsStatus.Details[4].equals("0")) {
                        HsStatus.Details[4] = "自動";
                        HsStatus.Details[5] = "28";
                        HsStatus.Details[6] = "自動";
                    }


AirCon[Integer.parseInt(HsStatus.Details[2])].SetStatus(HsStatus.Details[3],
HsStatus.Details[4],          Integer.parseInt(HsStatus.Details[5]),
HsStatus.Details[6]);
                    if (HsStatus.Details[0].equals("直接")) {

        AirCon[Integer.parseInt(HsStatus.Details[2])].direct = true;

        AirCon[Integer.parseInt(HsStatus.Details[2])].LastUse = 100;
                    }
                    else {

        AirCon[Integer.parseInt(HsStatus.Details[2])].direct = false;
                        if (HsStatus.Details[3].equals("
オン")) {

        AirCon[Integer.parseInt(HsStatus.Details[2])].LastUse = id;
                        }
                        else {

        AirCon[Integer.parseInt(HsStatus.Details[2])].LastUse = 100;
                        }
                    }
                }
                if (HsStatus.Details[1].equals("lock")) {
                    if (HsStatus.Details[0].equals("直接")) {
                        HsStatus.Lock                =
HsStatus.Details[3];
                        if (HsStatus.Details[3].equals("
施錠")) {
                            HsStatus.LockB   =
true;
                            HsStatus.Lock  = "
施錠";
                        }
                        else {
                            HsStatus.LockB   =
false;
                            HsStatus.Lock  = "
開錠";
                        }
                    }
                    else if (HsStatus.Details[3].equals("施錠"))
{
                        HsStatus.LockB = true;
                        HsStatus.Lock = "施錠";
                    }
                }
                try {
                    Class.forName(HsStatus.Driver);
                    String url = HsStatus.DB;
                    String user = HsStatus.Master;
                    String pass = HsStatus.Pass;
                    con   =   DriverManager.getConnection(url,
user, pass);
                    Statement stmt = con.createStatement();
                    HH = Integer.toString(HsStatus.TimeH);
                    MM = Integer.toString(HsStatus.TimeM);
                    if (HsStatus.Details[1].equals("ac")) {
```

```
                                        if
(HsStatus.Details[2].equals(Integer.toString(HsStatus.ACnum - 1))) {

                System.out.println(HsStatus.Details[7] + " " + HsStatus.Details[8]
+ " " + HsStatus.Details[0] + "操作 リビングエアコン: 電源" +
HsStatus.Details[3] + " 運転" + HsStatus.Details[4] + " 温度" +
HsStatus.Details[5] + "風量" + HsStatus.Details[6]);
                                        }
                                        else {

                System.out.println(HsStatus.Details[7] + " " + HsStatus.Details[8]
+ " " + HsStatus.Details[0] + "操作 エアコン" + HsStatus.Details[2] + ": 電源"
+ HsStatus.Details[3] + " 運転" + HsStatus.Details[4] + " 温度" +
HsStatus.Details[5] + "風量"

                + HsStatus.Details[6]);
                                        }
                                        if
(HsStatus.Details[0].equals("WEB")) {
                                                sql = "insert into " +
HsStatus.Details[1] + HsStatus.Details[2] + "(時間,操作者,電源,運転,温度,風
量,実時間) values('" + HH + ":" + MM + "','" + HsStatus.Details[7] + "','" +
HsStatus.Details[3] + "','" + HsStatus.Details[4] + "','"

                + HsStatus.Details[5] + "','" + HsStatus.Details[6] + "',NULL)";
                                        }
                                        else {
                                                sql = "insert into " +
HsStatus.Details[1] + HsStatus.Details[2] + "(時間,操作者,電源,運転,温度,風
量,実時間) values('" + HH + ":" + MM + "','" + HsStatus.Details[0] + "','" +
HsStatus.Details[3] + "','" + HsStatus.Details[4] + "','"

                + HsStatus.Details[5] + "','" + HsStatus.Details[6] + "',NULL)";
                                        }
                                }
                                if (HsStatus.Details[1].equals("lock")) {

                System.out.println(HsStatus.Details[7] + " " + HsStatus.Details[8]
+ " " + HsStatus.Details[0] + "操作 玄関錠:" + HsStatus.Details[3]);
                                        sql = "insert into " +
HsStatus.Details[1] + HsStatus.Details[2] + "(時間,操作者,状態,実時間)
values('" + HH + ":" + MM + "','" + HsStatus.Details[0] + "','" +
HsStatus.Details[3] + "',NULL)";
                                }
                                System.out.println();
                                stmt.executeUpdate(sql);
                                stmt.close();
                                con.close();
                        }
                        catch (Exception e) {
                                e.printStackTrace();
                        }
                        return;
                }
}
```

Master.java

```
public class Master {
        //操作の種類
        static int Method = 0;
        static String Methodname = "直接";
        //鍵の有無表示用
        static String Keyname ="鍵無";
        //行動終了
        static boolean FA = false;
        //エアコン操作行動終了
        static boolean ACFA = false;
        //再描画フラグ
        static boolean Redraw = true;
}
```

Monitor.java

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

//モニター
public class Monitor extends JFrame implements Runnable {
        JLabel label;
        JButton button[];

        public void run() {
                setTitle("モニター");

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                if (HsStatus.CPUonly == false) {
                        getContentPane().setLayout(new
GridLayout(2, 2));
                        setSize(500, 400);
                }
                else {
                        getContentPane().setLayout(new
GridLayout(1, 2));
                        setSize(600, 200);
                }
                label = new JLabel("", JLabel.CENTER);
                label.setFont(new Font("Dialog", Font.BOLD, 20));
                getContentPane().add(label);
                if (HsStatus.CPUonly == false) {
                        button = new JButton[15];
                        JPanel panel = new JPanel();
                        panel.setLayout(new GridLayout(4, 3));
                        button[0] = new JButton("部屋 0(貴方の部
屋)");
                        button[0].addActionListener(new
ActionListenerRoom0());
                        button[1] = new JButton("部屋 1");
                        button[1].addActionListener(new
ActionListenerRoom1());
                        button[2] = new JButton("部屋 2");
                        button[2].addActionListener(new
ActionListenerRoom2());
                        button[3] = new JButton("部屋 3");
                        button[3].addActionListener(new
ActionListenerRoom3());
                        button[4] = new JButton("部屋 4");
                        button[4].addActionListener(new
ActionListenerRoom4());
                        button[5] = new JButton("部屋 5");
                        button[5].addActionListener(new
ActionListenerRoom5());
                        button[6] = new JButton("リビング");
                        button[6].addActionListener(new
ActionListenerRoomL());
                        button[7] = new JButton("外出する");
                        button[7].addActionListener(new
ActionListenerRoomOut());
                        button[8] = new JButton("カギ持つ／持た
ない");
                        button[8].addActionListener(new
ActionListenerKey());
                        button[9] = new JButton("施錠／開錠");
                        button[9].addActionListener(new
ActionListenerLock());
                        button[10] = new JButton("操作切り替え");
                        button[10].addActionListener(new
ActionListenerChange());
                        button[11] = new JButton("時間進める");
                        button[11].addActionListener(new
ActionListenerNext());
                        for (int x = 0; x < HsStatus.ACnum - 1; x+
+) {
                                panel.add(button[x]);
                        }
                        panel.add(button[6]);
                        panel.add(button[7]);
                        panel.add(button[8]);
                        panel.add(button[9]);
                        panel.add(button[10]);
                        panel.add(button[11]);
                        getContentPane().add(panel);
                }
                setVisible(true);
                Redraw();

                while (true) {
                        //Redraw();//正しく動かないときはこれを
コメントアウト
                        if (Master.Redraw == true) {
                                Redraw();
                                Master.Redraw = false;
                        }
                        else if (Master.ACFA == true) {
                                Redraw();
                        }
                }
        }
        // エアコンウィンドウの再描画
        void Redraw() {
                String s = "";
                if (HsStatus.CPUonly == true) {
                        s = s + HsStatus.TimeH + ":" +
HsStatus.TimeM + "<br>";
                        for (int x = 0; x < HsStatus.ACnum - 1; x+
+) {
                                s = s + "U" + x + ":" +
HomeServer.User[x].placename + " ";
                        }
                        s = s + "<br>";
                        for (int x = 0; x < HsStatus.ACnum; x++) {
                                s = s + "AC" + x + ":" +
HomeServer.AirCon[x].Dengen + " ";
                        }
                        s = s + "<br>";
                        for (int x = 0; x < HsStatus.ACnum - 1; x+
+) {
                                s = s + "部屋" + x + ":" +
Sensor.RoomNum[x] + " ";
                        }
                        s = s + "L:" +
Sensor.RoomNum[HsStatus.ACnum - 1] + "  外 :" +
Sensor.RoomNum[HsStatus.ACnum];
                        label.setText("<html>" + s + "</html>");
                }
                if (HsStatus.CPUonly == false) {
```

```
                                for (int x = 0; x < HsStatus.ACnum - 1; x++) {
                                        s = s + "部屋" + x + ":" + Sensor.RoomNum[x] + " ";
                                }
                                s = s + "L:" + Sensor.RoomNum[HsStatus.ACnum - 1] + "  外 :" + Sensor.RoomNum[HsStatus.ACnum];
                                if ((HsStatus.Psensor == true && Master.Method == 1) || HomeServer.User[0].place != HsStatus.ACnum) {
                                        label.setText("<html>" + HsStatus.TimeH + ":" + HsStatus.TimeM + "<br>" + "現在:" + HomeServer.User[0].placename + "方法:" + Master.Methodname + " " + Master.Keyname + "玄関:" + HsStatus.Lock + "<br>" + s + "</html>");
                                }
                                else {
                                        label.setText("<html>" + HsStatus.TimeH + ":" + HsStatus.TimeM + "<br>" + "現在:" + HomeServer.User[0].placename + "方法:" + Master.Methodname + " " + Master.Keyname + "玄関:" + HsStatus.Lock + "</html>");
                                }
                        }
                }
                // リスナー
                class ActionListenerRoom0 implements ActionListener {
                        public void actionPerformed(ActionEvent ae) {
                                if (HomeServer.User[0].place == HsStatus.ACnum && HsStatus.LockB == true) {
                                        System.out.println("施錠されている");
                                        System.out.println();

                        HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi + HsStatus.EveryTime;
                                        for (int x = 0; x < HsStatus.ACnum; x++) {
                                                if (Sensor.RoomNum[x] > 0) {

                        System.out.println("家族がいたので開けてくれた");

                        HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi - HsStatus.EveryTime;

                        HomeServer.Registration("直接", "lock", "0", "開錠", "0", "0", "0", "実験者", HomeServer.User[0].placename, 0);

                        Master.Redraw = true;
                                                        break;
                                                }
                                        }
                                }
                                else {
                                        if (HomeServer.User[0].move == false) {

                        Sensor.minus(HomeServer.User[0].place);

                        HomeServer.User[0].oldplace = HomeServer.User[0].place;

                        HomeServer.User[0].oldplacename = HomeServer.User[0].placename;

                        HomeServer.User[0].place = 0;

                        Sensor.plus(HomeServer.User[0].place);

                        HomeServer.User[0].placename = "部屋 0(自室)";
                                                System.out.println("実験者: 移動 " + HomeServer.User[0].oldplacename + " → " + HomeServer.User[0].placename);

                        System.out.println();

                        HomeServer.User[0].move = true;
                                                if (HomeServer.User[0].place != HsStatus.ACnum && 0 == HomeServer.AirCon[HomeServer.User[0].place].LastUse) {

                        System.out.println("実験者: ネットワーク操作した エアコン" + HomeServer.User[0].place + "の部屋です。ネットワーク操作完遂。");

                        HomeServer.AirCon[HomeServer.User[0].place].LastUse = 100;
                                                }
                                                Master.Redraw = true;
                                        }
                                }
                        }
                }
                class ActionListenerRoom1 implements ActionListener {
                        public void actionPerformed(ActionEvent ae) {
                                if (HomeServer.User[0].place == HsStatus.ACnum && HsStatus.LockB == true) {
                                        System.out.println("施錠されている");
                                        System.out.println();

                        HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi + HsStatus.EveryTime;
                                        for (int x = 0; x < HsStatus.ACnum; x++) {
                                                if (Sensor.RoomNum[x] > 0) {

                        System.out.println("家族がいたので開けてくれた");

                        HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi - HsStatus.EveryTime;

                        HomeServer.Registration("直接", "lock", "0", "開錠", "0", "0", "0", "実験者", HomeServer.User[0].placename, 0);

                        Master.Redraw = true;
                                                        break;
                                                }
                                        }
                                }
                                else {
                                        if (HomeServer.User[0].move == false) {

                        Sensor.minus(HomeServer.User[0].place);

                        HomeServer.User[0].oldplace = HomeServer.User[0].place;

                        HomeServer.User[0].oldplacename = HomeServer.User[0].placename;

                        HomeServer.User[0].place = 1;

                        Sensor.plus(HomeServer.User[0].place);

                        HomeServer.User[0].placename = "部屋 1";
                                                System.out.println("実験者: 移動 " + HomeServer.User[0].oldplacename + " → " + HomeServer.User[0].placename);

                        System.out.println();

                        HomeServer.User[0].move = true;
                                                if (HomeServer.User[0].place != HsStatus.ACnum && 0 == HomeServer.AirCon[HomeServer.User[0].place].LastUse) {

                        System.out.println("実験者: ネットワーク操作した エアコン" + HomeServer.User[0].place + "の部屋です。ネットワーク操作完遂。");

                        HomeServer.AirCon[HomeServer.User[0].place].LastUse = 100;
                                                }
                                                Master.Redraw = true;
                                        }
                                }
                        }
                }
                class ActionListenerRoom2 implements ActionListener {
                        public void actionPerformed(ActionEvent ae) {
                                if (HomeServer.User[0].place == HsStatus.ACnum && HsStatus.LockB == true) {
                                        System.out.println("施錠されている");
                                        System.out.println();

                        HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi + HsStatus.EveryTime;
                                        for (int x = 0; x < HsStatus.ACnum; x++) {
                                                if (Sensor.RoomNum[x] > 0) {

                        System.out.println("家族がいたので開けてくれた");

                        HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi - HsStatus.EveryTime;

                        HomeServer.Registration("直接", "lock", "0", "開錠", "0", "0", "0", "実験者", HomeServer.User[0].placename, 0);

                        Master.Redraw = true;
                                                        break;
                                                }
                                        }
                                }
                                else {
                                        if (HomeServer.User[0].move == false) {

                        Sensor.minus(HomeServer.User[0].place);

                        HomeServer.User[0].oldplace = HomeServer.User[0].place;

                        HomeServer.User[0].oldplacename = HomeServer.User[0].placename;

                        HomeServer.User[0].place = 2;
```

```
		Sensor.plus(HomeServer.User[0].place);

		HomeServer.User[0].placename = "部屋2";
		System.out.println("実験者： 移動 " + HomeServer.User[0].oldplacename + " → " + HomeServer.User[0].placename);

		System.out.println();

		HomeServer.User[0].move = true;
		if (HomeServer.User[0].place != HsStatus.ACnum && 0 == HomeServer.AirCon[HomeServer.User[0].place].LastUse) {

		System.out.println("実験者: ネットワーク操作した エアコン" + HomeServer.User[0].place + "の部屋です。ネットワーク操作完遂。");

		HomeServer.AirCon[HomeServer.User[0].place].LastUse = 100;
		}
		Master.Redraw = true;
		}
		}
		}
	}
	class ActionListenerRoom3 implements ActionListener {
		public void actionPerformed(ActionEvent ae) {
			if (HomeServer.User[0].place == HsStatus.ACnum && HsStatus.LockB == true) {
		System.out.println("施錠されている");
		System.out.println();

		HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi + HsStatus.EveryTime;
		for (int x = 0; x < HsStatus.ACnum; x++) {
		if (Sensor.RoomNum[x] > 0) {

		System.out.println("家族がいたので開けてくれた");

		HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi - HsStatus.EveryTime;

		HomeServer.Registration("直接", "lock", "0", "開錠", "0", "0", "0", "実験者", HomeServer.User[0].placename, 0);

		Master.Redraw = true;
		break;
		}
		}
		}
		else {
			if (HomeServer.User[0].move == false) {

		Sensor.minus(HomeServer.User[0].place);

		HomeServer.User[0].oldplace = HomeServer.User[0].place;

		HomeServer.User[0].oldplacename = HomeServer.User[0].placename;

		HomeServer.User[0].place = 3;

		Sensor.plus(HomeServer.User[0].place);

		HomeServer.User[0].placename = "部屋3";
		System.out.println("実験者： 移動 " + HomeServer.User[0].oldplacename + " → " + HomeServer.User[0].placename);

		System.out.println();

		HomeServer.User[0].move = true;
		if (HomeServer.User[0].place != HsStatus.ACnum && 0 == HomeServer.AirCon[HomeServer.User[0].place].LastUse) {

		System.out.println("実験者: ネットワーク操作した エアコン" + HomeServer.User[0].place + "の部屋です。ネットワーク操作完遂。");

		HomeServer.AirCon[HomeServer.User[0].place].LastUse = 100;
		}
		Master.Redraw = true;
		}
		}
		}
	}

	class ActionListenerRoom4 implements ActionListener {
		public void actionPerformed(ActionEvent ae) {
			if (HomeServer.User[0].place == HsStatus.ACnum && HsStatus.LockB == true) {
		System.out.println("施錠されている");
		System.out.println();

		HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi + HsStatus.EveryTime;
		for (int x = 0; x < HsStatus.ACnum; x++) {
		if (Sensor.RoomNum[x] > 0) {

		System.out.println("家族がいたので開けてくれた");

		HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi - HsStatus.EveryTime;

		HomeServer.Registration("直接", "lock", "0", "開錠", "0", "0", "0", "実験者", HomeServer.User[0].placename, 0);

		Master.Redraw = true;
		break;
		}
		}
		}
		else {
			if (HomeServer.User[0].move == false) {

		Sensor.minus(HomeServer.User[0].place);

		HomeServer.User[0].oldplace = HomeServer.User[0].place;

		HomeServer.User[0].oldplacename = HomeServer.User[0].placename;

		HomeServer.User[0].place = 4;

		Sensor.plus(HomeServer.User[0].place);

		HomeServer.User[0].placename = "部屋4";
		System.out.println("実験者： 移動 " + HomeServer.User[0].oldplacename + " → " + HomeServer.User[0].placename);

		System.out.println();

		HomeServer.User[0].move = true;
		if (HomeServer.User[0].place != HsStatus.ACnum && 0 == HomeServer.AirCon[HomeServer.User[0].place].LastUse) {

		System.out.println("実験者: ネットワーク操作した エアコン" + HomeServer.User[0].place + "の部屋です。ネットワーク操作完遂。");

		HomeServer.AirCon[HomeServer.User[0].place].LastUse = 100;
		}
		Master.Redraw = true;
		}
		}
		}
	}
	class ActionListenerRoom5 implements ActionListener {
		public void actionPerformed(ActionEvent ae) {
			if (HomeServer.User[0].place == HsStatus.ACnum && HsStatus.LockB == true) {
		System.out.println("施錠されている");
		System.out.println();

		HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi + HsStatus.EveryTime;
		for (int x = 0; x < HsStatus.ACnum; x++) {
		if (Sensor.RoomNum[x] > 0) {

		System.out.println("家族がいたので開けてくれた");

		HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi - HsStatus.EveryTime;

		HomeServer.Registration("直接", "lock", "0", "開錠", "0", "0", "0", "実験者", HomeServer.User[0].placename, 0);

		Master.Redraw = true;
		break;
		}
		}
		}
		else {
			if (HomeServer.User[0].move == false) {

		Sensor.minus(HomeServer.User[0].place);

		HomeServer.User[0].oldplace = HomeServer.User[0].place;

		HomeServer.User[0].oldplacename = HomeServer.User[0].placename;
```

```
                HomeServer.User[0].place = 5;

                Sensor.plus(HomeServer.User[0].place);

                HomeServer.User[0].placename = "部屋 5";
                                                System.out.println("実験者: 移動 " + HomeServer.User[0].oldplacename + " → " + HomeServer.User[0].placename);

                System.out.println();

                HomeServer.User[0].move = true;
                                                if (HomeServer.User[0].place != HsStatus.ACnum && 0 == HomeServer.AirCon[HomeServer.User[0].place].LastUse) {

                System.out.println("実験者: ネットワーク操作した エアコン" + HomeServer.User[0].place + "の部屋です。ネットワーク操作完遂。");

                HomeServer.AirCon[HomeServer.User[0].place].LastUse = 100;
                                                }
                                                Master.Redraw = true;
                                        }
                                }
                        }
                }
                class ActionListenerRoomL implements ActionListener {
                        public void actionPerformed(ActionEvent ae) {
                                if (HomeServer.User[0].place == HsStatus.ACnum && HsStatus.LockB == true) {
                                        System.out.println("施錠されている");
                                        System.out.println();

                HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi + HsStatus.EveryTime;
                                        for (int x = 0; x < HsStatus.ACnum; x++) {
                                                if (Sensor.RoomNum[x] > 0) {

                System.out.println("家族がいたので開けてくれた");

                HomeServer.User[0].shimedashi = HomeServer.User[0].shimedashi - HsStatus.EveryTime;

                HomeServer.Registration("直接", "lock", "0", "開錠", "0", "0", "0", "実験者", HomeServer.User[0].placename, 0);

                Master.Redraw = true;
                                                        break;
                                                }
                                        }
                                }
                                else {
                                        if (HomeServer.User[0].move == false) {

                Sensor.minus(HomeServer.User[0].place);

                HomeServer.User[0].oldplace = HomeServer.User[0].place;

                HomeServer.User[0].oldplacename = HomeServer.User[0].placename;

                HomeServer.User[0].place = HsStatus.ACnum - 1;

                Sensor.plus(HomeServer.User[0].place);

                HomeServer.User[0].placename = "リビング";
                                                System.out.println("実験者: 移動 " + HomeServer.User[0].oldplacename + " → " + HomeServer.User[0].placename);

                System.out.println();

                HomeServer.User[0].move = true;
                                                if (HomeServer.User[0].place != HsStatus.ACnum && 0 == HomeServer.AirCon[HomeServer.User[0].place].LastUse) {

                System.out.println("実験者: ネットワーク操作した エアコン" + HomeServer.User[0].place + "の部屋です。ネットワーク操作完遂。");

                HomeServer.AirCon[HomeServer.User[0].place].LastUse = 100;
                                                }
                                                Master.Redraw = true;
                                        }
                                }
                        }
                }
                class ActionListenerRoomOut implements ActionListener {
                        public void actionPerformed(ActionEvent ae) {
                                if (HomeServer.User[0].place != HsStatus.ACnum && HsStatus.LockB == true) {
                                        System.out.println("施錠されている");
                                        System.out.println();
                                }
                                else {
                                        if (HomeServer.User[0].move == false) {
                                        }

                Sensor.minus(HomeServer.User[0].place);
                                        HomeServer.User[0].oldplace = HomeServer.User[0].place;

                HomeServer.User[0].oldplacename = HomeServer.User[0].placename;
                                        HomeServer.User[0].place = HsStatus.ACnum;

                Sensor.plus(HomeServer.User[0].place);

                HomeServer.User[0].placename = "外出中";
                                        System.out.println("実験者: 移動 " + HomeServer.User[0].oldplacename + " → " + HomeServer.User[0].placename);
                                        System.out.println();
                                        HomeServer.User[0].move = true;
                                        if (HomeServer.User[0].place != HsStatus.ACnum && 0 == HomeServer.AirCon[HomeServer.User[0].place].LastUse) {
                                                System.out.println("実験者: ネットワーク操作した エアコン" + HomeServer.User[0].place + "の部屋です。ネットワーク操作完遂。");

                HomeServer.AirCon[HomeServer.User[0].place].LastUse = 100;
                                        }
                                        Master.Redraw = true;
                                }
                        }
                }
                class ActionListenerKey implements ActionListener {
                        public void actionPerformed(ActionEvent ae) {
                                if (HomeServer.User[0].place != HsStatus.ACnum) {
                                        if (HomeServer.User[0].key == false) {

                HomeServer.User[0].key = true;
                                                Master.Keyname = "鍵有";
                                                Master.Redraw = true;
                                        }
                                        else {

                HomeServer.User[0].key = false;
                                                Master.Keyname = "鍵無";
                                                Master.Redraw = true;
                                        }
                                }
                        }
                }
                class ActionListenerLock implements ActionListener {
                        public void actionPerformed(ActionEvent ae) {
                                if (HomeServer.User[0].place != HsStatus.ACnum) {
                                        if (HsStatus.LockB == false) {

                HomeServer.Registration("直接", "lock", "0", "施錠", "", "", "", "実験者", HomeServer.User[0].placename, 0);
                                        }
                                        else {

                HomeServer.Registration("直接", "lock", "0", "開錠", "", "", "", "実験者", HomeServer.User[0].placename, 0);
                                        }
                                        Master.FA = true;
                                }
                                else if (HomeServer.User[0].place == HsStatus.ACnum) {
                                        if (HomeServer.User[0].key == true && Master.Method == 0) {
                                                if (HsStatus.LockB == false) {

                HomeServer.Registration("直接", "lock", "0", "施錠", "", "", "", "実験者", HomeServer.User[0].placename, 0);
                                                }
                                                else {

                HomeServer.Registration("直接", "lock", "0", "開錠", "", "", "", "実験者", HomeServer.User[0].placename, 0);
                                                }
                                                Master.FA = true;
                                        }
                                        else if (HomeServer.User[0].key == true && Master.Method == 1) {
```

```
		HomeServer.Registration("WEB", "lock", "0", "施錠", "", "", "", "実験者", HomeServer.User[0].placename, 0);
							Master.FA = true;
						}
					}
				}
			}
			class ActionListenerChange implements ActionListener {
				public void actionPerformed(ActionEvent ae) {
					if (Master.Method == 0) {
						Master.Method = 1;
						Master.Methodname = "WEB";
						Master.Redraw = true;
					}
					else {
						Master.Method = 0;
						Master.Methodname = "直接";
						Master.Redraw = true;
					}
				}
			}
			class ActionListenerNext implements ActionListener {
				public void actionPerformed(ActionEvent ae) {
					HomeServer.User[0].move = false;
					Master.Redraw = true;
					Master.FA = true;
				}
			}
}


AirCon.java

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

//エアコンクラス
public class AirCon extends JFrame implements Runnable {
			int id;// エアコン ID
			// 表示用ラベル等
			JLabel label;
			JButton buttonP;
			JButton buttonU;
			JButton buttonD;
			JButton buttonM;
			JButton buttonW;
			String Dengen = "オフ";
			String Unten = "暖房";
			String Ondo = "20";
			String Furyo = "自動";
			// 動作状態数値化
			int DengenPresent = 0;
			int DengenPast = 0;
			int OndoInt = 28;
			// 最後のネットワーク操作者の ID 記録
			int LastUse = 100;
			// 消し忘れ時間
			int loss = 0;
			// ネットワーク操作キャンセル回数
			int webcancel = 0;
			// 直接操作かどうか
			boolean direct = true;
			// 実験者操作フラグ
			boolean Dcheck = false;
			// 表示部の更新フラグ
			boolean Koushin = false;

			public void run() {
				setTitle("エアコン" + id);

			setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
				getContentPane().setLayout(new GridLayout(1, 2));
				label = new JLabel("<html>" + Dengen + " " + Unten + "<br>" + Ondo + "度 風量" + Furyo + "<br>消し忘れ時間" + loss + "</html>", JLabel.CENTER);
				label.setFont(new Font("Dialog", Font.BOLD, 20));
				getContentPane().add(label);
				JPanel panel = new JPanel();
				panel.setLayout(new GridLayout(2, 3));

				buttonP = new JButton("電源");
				buttonP.addActionListener(new ActionListenerDengen());
				buttonU = new JButton("温度↑");
				buttonU.addActionListener(new ActionListenerOndoUP());
				buttonW = new JButton("風量");
				buttonW.addActionListener(new ActionListenerFuryo());
				buttonM = new JButton("モード");
				buttonM.addActionListener(new ActionListenerUnten());
				buttonD = new JButton("温度↓");
				buttonD.addActionListener(new ActionListenerOndoDOWN());
				panel.add(buttonP);
				panel.add(buttonU);
				panel.add(buttonW);
				panel.add(buttonM);
				panel.add(buttonD);
				getContentPane().add(panel);
				setSize(450, 200);
				setVisible(true);

				while (true) {
					if (HsStatus.CPUonly == false && (HomeServer.User[0].place == id || Master.Method == 1)) {
						if (Koushin == true) {
							SetValue();
							Koushin = false;
							if (Dcheck == true) {
								if (DengenPresent == 1 || DengenPast == 1) {

			DengenPast = DengenPresent;

			if (HomeServer.AirCon[id].direct == false && HomeServer.AirCon[id].LastUse != 100) {

				System.out.println("ユーザー" + HomeServer.AirCon[id].LastUse + ": ネットワーク操作失敗だ エアコン" + HomeServer.AirCon[id].id);


HomeServer.User[HomeServer.AirCon[id].LastUse].displeasure();

				HomeServer.AirCon[id].webcancel++;

				HomeServer.AirCon[id].LastUse = 100;

			}

			HomeServer.Registration(Master.Methodname, "ac", String.valueOf(id), Dengen, Unten, Ondo, Furyo, "実験者", HomeServer.User[0].placename, 0);

			Master.ACFA = true;

			Master.FA = true;

			Dcheck = false;
								}
							}
						}
					}
					if (HsStatus.CPUonly == true && Koushin == true) {
						SetValue();
						Koushin = false;
					}
				}
			}
			// エアコンウィンドウの再描画
			void SetValue() {
				if (direct == false) {
					label.setText("<html>" + Dengen + " " + Unten + "<br>" + Ondo + "度 風量 " + Furyo + " WEB<br>消し忘れ時間" + loss + "</html>");
				}
				else {
					label.setText("<html>" + Dengen + " " + Unten + "<br>" + Ondo + "度 風量 " + Furyo + "<br>消し忘れ時間" + loss + "</html>");
				}
			}

			int SetStatus(String a, String b, int c, String d) {
				Dengen = a;
				if (Dengen.equals("オフ")) {
					DengenPast = DengenPresent;
					DengenPresent = 0;
				}
				else {
					DengenPast = DengenPresent;
					DengenPresent = 1;
				}
				Unten = b;
				OndoInt = c;
				Ondo = String.valueOf(OndoInt);
				Furyo = d;
				Koushin = true;
				return id;
			}
			class ActionListenerDengen implements ActionListener {
				public void actionPerformed(ActionEvent ae) {
					if (HsStatus.CPUonly == false && (HomeServer.User[0].place == id || Master.Method == 1)) {
						if (Dengen.equals("オフ")) {
							Dengen = "オン";
							DengenPast = DengenPresent;
							DengenPresent = 1;
						}
```

```
                                        else {
                                                Dengen = "オフ";
                                                DengenPast      =
DengenPresent;
                                                DengenPresent   =
0;
                                        }
                                        Koushin = true;
                                        Dcheck = true;
                                }
                        }
                }
                class ActionListenerUnten implements ActionListener {
                        public void actionPerformed(ActionEvent ae) {
                                if    (HsStatus.CPUonly    ==    false    &&
(HomeServer.User[0].place == id || Master.Method == 1)) {
                                        if (Unten.equals("自動")) {
                                                Unten = "暖房";
                                        }
                                        else if (Unten.equals("暖房")) {
                                                Unten = "冷房";
                                        }
                                        else if (Unten.equals("冷房")) {
                                                Unten = "ドライ";
                                        }
                                        else {
                                                Unten = "自動";
                                        }
                                        DengenPast                      =
DengenPresent;
                                        Koushin = true;
                                        Dcheck = true;
                                }
                        }
                }
                class ActionListenerOndoUP implements ActionListener {
                        public void actionPerformed(ActionEvent ae) {
                                if    (HsStatus.CPUonly    ==    false    &&
(HomeServer.User[0].place == id || Master.Method == 1)) {
                                        if (OndoInt < 30) {
                                                OndoInt++;
                                        }
                                        Ondo                            =
String.valueOf(OndoInt);
                                        DengenPast                      =
DengenPresent;
                                        Koushin = true;
                                        Dcheck = true;
                                }
                        }
                }
                class ActionListenerOndoDOWN implements ActionListener {
                        public void actionPerformed(ActionEvent ae) {
                                if    (HsStatus.CPUonly    ==    false    &&
(HomeServer.User[0].place == id || Master.Method == 1)) {
                                        if (OndoInt > 18) {
                                                OndoInt--;
                                        }
                                        Ondo                            =
String.valueOf(OndoInt);
                                        DengenPast                      =
DengenPresent;
                                        Koushin = true;
                                        Dcheck = true;
                                }
                        }
                }
                class ActionListenerFuryo implements ActionListener {
                        public void actionPerformed(ActionEvent ae) {
                                if    (HsStatus.CPUonly    ==    false    &&
(HomeServer.User[0].place == id || Master.Method == 1)) {
                                        if (Furyo.equals("自動")) {
                                                Furyo = "弱";
                                        }
                                        else if (Furyo.equals("弱")) {
                                                Furyo = "中";
                                        }
                                        else if (Furyo.equals("中")) {
                                                Furyo = "強";
                                        }
                                        else {
                                                Furyo = "自動";
                                        }
                                        DengenPast                      =
DengenPresent;
                                        Koushin = true;
                                        Dcheck = true;
                                }
                        }
                }
}
```

User.java

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.sql.*;

public class User extends Thread {

        // ユーザーの情報など
        String name = "";
        int id = 0;
        double memory = 0;
        int place = id;
        String placename = "";
        int oldplace = id;
        String oldplacename = "";
        // 閉め出しフラグ
        boolean shutout = false;
        // 鍵を持つか true 持つ
        boolean key = false;
        // 行動終了フラグ
        boolean BR = false;
        // 移動済みフラグ
        boolean move = false;
        // 睡眠フラグ
        boolean sleep = true;
        // 不快感フラグ
        boolean dis = false;
        // 不快回数
        int displeasure = 0;
        // 閉め出され時間
        int shimedashi = 0;
        // 外出しなかったフラグ
        boolean Notout = false;
        // 行動リスト内容格納用
        String UA[];

        synchronized void displeasure() {
                displeasure++;
        }

        public void run() {
                // 時間
                int hh = HsStatus.TH;
                int mm = HsStatus.TM;
                // 乱数用
                double ran = 0;
                // リスト読み取り用
                String line = "";
                UA = new String[13];
                String UAPname = "";
                int UAP = 0;
                // SQL
                Connection con = null;
                String sql = "";
                int TimeNow = 0;
                String[]  SQLTime[]  =  { { "",  "" }, { "",  "" } };// new
String[2];
                int[] SQLTimeC = { 0, 0 };

                Statement stmt;
                String url = HsStatus.DB;
                String user = HsStatus.Master;
                String pass = HsStatus.Pass;
                ResultSet rs;
                String[] jikan = { "", "" };
                String[] dengen = { "", "" };
                String[] sousa = { "", "" };
                String Method = "";
                String LockTime = "";
                String[] LockSQLTime = { "", "" };
                int LockSQLTimeC = 0;
                String Lock = "";
                int zaitaku = 0;
                boolean Anxious = true;
                int ACNo = 99;
                String TargetR = "";
                int count = 0;
                String LACNo = Integer.toString(HsStatus.Living);
                String MyRNo = Integer.toString(id);
                String filename = "u" + id + "ac.csv";
                name = "ユーザー" + id;

                /*
                 * line の内容 時刻,状態,操作機器,機器番号,内容
1,2,3,4,5,方法,場所 忘れる可能性等
                 *          0 1   2      3      4      5   6 7 8 9  10    11
12
                 */

                try {
                        FileReader          csv          =          new
FileReader(filename); // CSV データファイル
                        BufferedReader          br          =          new
BufferedReader(csv);

                        while ((line = br.readLine()) != null) {
                                while (true) {
                                        if          (hh          ==
HsStatus.TimeH && mm == HsStatus.TimeM && HsStatus.UPC == true) {
                                                UA      =
line.split(",");
```

```
                                                                        if
(UA[12].equals("4")) {

                sleep = !sleep;

                                                                        }
                                                                        if (sleep
== false) {
                if (UA[3].equals("ac")) {

                                if (UA[4].equals(LACNo)) {

                                                UA[4]  =  Integer.toString(HsStatus.ACnum
- 1);

                                }

                                else if (UA[4].equals("0")) {

                                                UA[4] = MyRNo;

                                }

                                else if (UA[4].equals(MyRNo)) {

                                                UA[4] = "0";

                                }

                }

                move = false;

                BR = false;


                // 移動

                UAP = Integer.parseInt(UA[11]);

                UAPname = "部屋" + UAP;

                if (UAP == 0) {

                                UAP = id;

                                UAPname = "部屋" + UAP + "(自室)";

                }

                else if (UAP == id) {

                                UAP = 0;

                                UAPname = "部屋" + UAP;

                }

                if (UAP == HsStatus.Living) {

                                UAP = HsStatus.ACnum - 1;

                                UAPname = "リビング";

                }

                else if (UAP == HsStatus.Out) {

                                UAP = HsStatus.ACnum;

                                UAPname = "外出";

                }

                ran = Math.random();

                if (ran <= 0.33 && UA[12].equals("2") && Notout == false && UAP
== HsStatus.ACnum) {

                                Notout = true;

                                System.out.println("ユーザー" + id + ": 外出を取りや
めた。");

                }

                if (Notout == true && UAP != HsStatus.ACnum &&
UA[12].equals("0")) {

                                Notout = false;

                }

                if (Notout == false && place == HsStatus.ACnum && key == false
&& UAP != HsStatus.ACnum && HsStatus.LockB == true) {

                                System.out.println("ユーザー" + id + ": 施錠されてい
て入れない。");

                                System.out.println();

                                shimedashi = shimedashi + HsStatus.EveryTime;

                                shutout = true;

                                for (int x = 0; x < HsStatus.ACnum; x++) {

                                                if (Sensor.RoomNum[x] > 0) {

                                                                System.out.println("家族がいた
ので開けてくれた");

                                                                shimedashi  =  shimedashi  -
HsStatus.EveryTime;

                                                                HomeServer.Registration("  直
接", "lock", "0", "開錠", "0", "0", "0", name, placename, id);

                                                                shutout = false;

                                                                break;

                                                }

                                }

                }

                if ((shutout == false && Notout == false && (UA[12].equals("0")))
|| (Notout == true && (UA[12].equals("1") || UA[12].equals("3")))) {

                                oldplace = place;

                                oldplacename = placename;

                                Sensor.minus(place);

                                Sensor.plus(UAP);

                                place = UAP;

                                placename = UAPname;

                                if (oldplace != place) {

                                                if (place != HsStatus.ACnum && id ==
HomeServer.AirCon[place].LastUse) {

                                                                System.out.println("ユーザー"
+ id + ": ネットワーク操作した エアコン" + oldplace + "の部屋です。ネット
ワーク操作完遂。");


                HomeServer.AirCon[place].LastUse = 100;

                                                }

                                                ran = Math.random();

                                                if (ran < memory) {

                                                                if (HsStatus.history == false
&& HsStatus.Psensor == false && oldplace != HsStatus.ACnum &&
HomeServer.AirCon[oldplace].DengenPresent == 1 &&
Sensor.RoomNum[oldplace] == 0) {

                                                                                System.out.println("
ユーザー" + id + ": 誰もいなくなるため エアコン" + oldplace + "を停止させ
ます。");

                                                                                if
(HomeServer.AirCon[oldplace].direct == false &&
HomeServer.AirCon[oldplace].LastUse != 100) {


                System.out.println("  ユ  ー  ザ  ー  " +
HomeServer.AirCon[oldplace].LastUse + ": ネットワーク操作失敗 エアコン"
+ HomeServer.AirCon[oldplace].id);


HomeServer.User[HomeServer.AirCon[oldplace].LastUse].displeasure();


                HomeServer.AirCon[oldplace].webcancel++;


                HomeServer.AirCon[oldplace].LastUse = 100;

                                                                                }


                HomeServer.Registration("直接", "ac", Integer.toString(oldplace),
"オフ", "0", "0", "0", name, oldplacename, id);

                                                                }
```

```
                    if ((HsStatus.history == true || HsStatus.Psensor == true) && oldplace != HsStatus.ACnum && HomeServer.AirCon[oldplace].DengenPresent == 1 && Sensor.RoomNum[oldplace] == 0) {

                        if (HomeServer.AirCon[oldplace].direct == true) {

            System.out.println("ユーザー" + id + ": 誰もいなくなるため エアコン" + oldplace + "を停止させます。");

            HomeServer.Registration("直接", "ac", Integer.toString(oldplace), "オフ", "0", "0", "0", name, oldplacename, id);

                        }

                        if (HomeServer.AirCon[oldplace].direct == false) {

                            if (HsStatus.history == true) {

        try {

                con = DriverManager.getConnection(url, user, pass);

                stmt = con.createStatement();

                TimeNow = HsStatus.TimeH * 60 + HsStatus.TimeM;

                sql = "select * from ac" + oldplace + " order by 実時間 desc limit 1";

                rs = stmt.executeQuery(sql);

                rs.next();

                jikan[0] = rs.getString("時間");

                sousa[0] = rs.getString("操作者");

                SQLTime[0] = jikan[0].split(":");

                SQLTimeC[0] = Integer.parseInt(SQLTime[0][0]) * 60 + Integer.parseInt(SQLTime[0][1]);

                if (TimeNow - SQLTimeC[0] > HsStatus.WebSetuden) {

                    System.out.println("ユーザー" + id + ": " + sousa[0] + "のネットワーク操作から" + (TimeNow - SQLTimeC[0]) + "分経過。節電のため エアコン" + oldplace + "を停止させます。");

                    HomeServer.Registration("直接", "ac", Integer.toString(oldplace), "オフ", "0", "0", "0", name, oldplacename, id);

                }

                else {

                    System.out.println("ユーザー" + id + ": ネットワーク操作で動作している。" + HsStatus.WebSetuden + "分経過するまではエアコン" + oldplace + "はそのままにしておきます。");

                }

        }

        catch (Exception e) {

        }

                            }

                        else if (HsStatus.Psensor == true) {

            System.out.println("ユーザー" + id + ": ネットワーク操作で動作しているので エアコン" + oldplace + "は放置します。");

                        }

                    }

                }

                System.out.println("ユーザー" + id + ": 移動 " + oldplacename + " → " + placename);

                if (oldplace == HsStatus.ACnum && key == true) {

                    if (HsStatus.LockB == true) {

            HomeServer.Registration("直接", "lock", "0", "開錠", "0", "0", "0", name, placename, id);

                        System.out.println("玄関錠を開けた。");

                    }

                    System.out.println("鍵を置いた。");

                    key = false;

                }

                System.out.println();

                move = true;

            }

        }

        // 行動

        ran = Math.random();

        if (UA[3].equals("0")) {

            // 何もしない

        }

        else if (shutout == true && UA[10].equals("直接")) {

            // 閉めだされている

        }

        else if (ran >= memory && UA[12].equals("1")) {

            System.out.println("ユーザー" + id + ": 何かし忘れた。");

            System.out.println();

        }

        else if (UA[12].equals("4")) {

            System.out.println("ユーザー" + id + ": " + UA[2]);

        }

        else if ((Notout == false && (UA[12].equals("0") || UA[12].equals("1") || UA[12].equals("2"))) || (Notout == true && (UA[12].equals("1") || UA[12].equals("3")))) {

            if (UA[2].equals("0")) {

            }

            else {

                if (Notout == false && (UA[12].equals("0") || UA[12].equals("1") || UA[12].equals("2"))) {

                    System.out.println("ユーザー" + id + ": " + UA[2]);

                }

                if (Notout == true && UA[12].equals("3")) {
```

```
                                        System.out.println("ユーザー"
+ id + ": " + UA[2]);

                                }
                        }
                        // ネットワーク操作センサー有り部屋に誰かいる時
                        if (UA[10].equals("WEB") && HsStatus.Psensor ==
true && Sensor.RoomNum[Integer.parseInt(UA[4])] > 0) {
                                if (UA[4].equals("100")) {
                                        System.out.println("ユーザー"
+ id + ": リビングに誰かいます。WEB からの操作を止めます。");
                                        System.out.println();
                                }
                                else {
                                        System.out.println("ユーザー"
+ id + ": 部屋 " + UA[4] + " に誰かいます。WEB からの操作を止めます。");
                                        System.out.println();
                                }
                        }
                        else {
                                if (UA[3].equals("ac")) {
                                        if (HsStatus.history == true) {
                                                try {
                                                        con =
DriverManager.getConnection(url, user, pass);
                                                        stmt =
con.createStatement();
        TimeNow = HsStatus.TimeH * 60 + HsStatus.TimeM;
                                                        sql =
"select count(*) from ac" + UA[4] + "";
                                                        rs =
stmt.executeQuery(sql);
        rs.next();
                                                        if
(rs.getInt(1) == 0) {
        if (UA[5].equals("オン")) {
                ACNo = Integer.parseInt(UA[4]);
        }
        else {
                ACNo = 99;
        }
        HomeServer.Registration(UA[10], UA[3], UA[4], UA[5], UA[6],
UA[7], UA[8], name, placename, id);
                                                        }
                                                        if
(rs.getInt(1) > 0) {
        sql = "select * from ac" + UA[4] + " order by 実時間 desc limit 1";
        rs = stmt.executeQuery(sql);
        rs.next();
        jikan[0] = rs.getString("時間");
        sousa[0] = rs.getString("操作者");
        dengen[0] = rs.getString("電源");
        SQLTime[0] = jikan[0].split(":");
        SQLTimeC[0] = Integer.parseInt(SQLTime[0][0]) * 60 +
Integer.parseInt(SQLTime[0][1]);
        if (dengen[0].equals("オン")) {
                if (sousa[0].equals("直接")) {
                        if (UA[10].equals("直接")) {
                                if (UA[5].equals("オン")) {
                                        ACNo =
Integer.parseInt(UA[4]);
                                }
                                else {
                                        ACNo = 99;
                                }
        HomeServer.Registration(UA[10], UA[3], UA[4], UA[5], UA[6],
UA[7], UA[8], name, placename, id);
                        }
                        else {
                                if (TimeNow - SQLTimeC[0] >
HsStatus.Setuden) {
                                        if (UA[5].equals("オ
ン")) {
                                                ACNo =
Integer.parseInt(UA[4]);
                                        }
                                        else {
                                                ACNo =
99;
                                        }
                                        System.out.println("
ユーザー" + id + ": 最後の操作から" + (TimeNow - SQLTimeC[0]) + "分経過。
一定時間経過しているのでエアコン" + UA[4] + " 操作します。");
        HomeServer.Registration(UA[10], UA[3], UA[4], UA[5], UA[6],
UA[7], UA[8], name, placename, id);
                                }
                        }
                }
        }
```

```
                else {

                        if (TimeNow - SQLTimeC[0] > HsStatus.WebSetuden) {

                                if (UA[5].equals("オン")) {

                                        ACNo = Integer.parseInt(UA[4]);

                                }

                                else {

                                        ACNo = 99;

                                }

                                System.out.println("ユーザー" + id + ": " + sousa[0] + "のネットワーク操作から" + (TimeNow - SQLTimeC[0]) + "分経過。一定時間経過しているのでエアコン" + UA[4] + "を操作します。");


                HomeServer.Registration(UA[10], UA[3], UA[4], UA[5], UA[6], UA[7], UA[8], name, placename, id);

                        }

                        else {

                                System.out.println("ユーザー" + id + ": ネットワーク操作で動作している。" + HsStatus.WebSetuden + "分経過するまではエアコン" + UA[4] + "はそのままにしておきます。");

                        }

                }

        }

        else {

                if (UA[5].equals("オン")) {

                        ACNo = Integer.parseInt(UA[4]);

                }

                else {

                        ACNo = 99;

                }

                HomeServer.Registration(UA[10], UA[3], UA[4], UA[5], UA[6], UA[7], UA[8], name, placename, id);

        }
                                                        }

                                }

                                catch (Exception e) {

                                }

                        }

                        else if (HsStatus.Psensor == true) {

                                if (UA[10].equals("直接")) {

                                        if (HomeServer.AirCon[place].direct == true) {

                if (UA[5].equals("オン")) {

                        ACNo = Integer.parseInt(UA[4]);

                }

                else {

                        ACNo = 99;

                }

                HomeServer.Registration(UA[10], UA[3], UA[4], UA[5], UA[6], UA[7], UA[8], name, placename, id);

                                        }

                                        else if (HomeServer.AirCon[place].direct == false && HomeServer.AirCon[place].DengenPresent == 1) {

                System.out.println("ユーザー" + id + ": ネットワーク操作で動作しているのでエアコン" + UA[4] + "は放置します。");

                                        }

                                        else if (HomeServer.AirCon[place].direct == false && HomeServer.AirCon[place].DengenPresent == 0) {

                if (UA[5].equals("オン")) {

                        ACNo = Integer.parseInt(UA[4]);

                }

                else {

                        ACNo = 99;

                }

                HomeServer.Registration(UA[10], UA[3], UA[4], UA[5], UA[6], UA[7], UA[8], name, placename, id);

                                        }
                                }

                                else {

                                        if (HomeServer.AirCon[Integer.parseInt(UA[4])].direct == false && HomeServer.AirCon[Integer.parseInt(UA[4])].LastUse != 100) {

                System.out.println("ユーザー" + HomeServer.AirCon[Integer.parseInt(UA[4])].LastUse + ": ネットワーク操作失敗 エアコン" + HomeServer.AirCon[Integer.parseInt(UA[4])].id);


HomeServer.User[HomeServer.AirCon[Integer.parseInt(UA[4])].LastUse].displeasure();

                HomeServer.AirCon[Integer.parseInt(UA[4])].webcancel++;

                HomeServer.AirCon[Integer.parseInt(UA[4])].LastUse = 100;

                                        }

                                        if (UA[5].equals("オン")) {
```

```
ACNo = Integer.parseInt(UA[4]);
                                                }
                                                else {
ACNo = 99;
                                                }

HomeServer.Registration(UA[10], UA[3], UA[4], UA[5], UA[6], UA[7], UA[8], name, placename, id);
                                                }
                                        }
                                        else {
                                                if (HomeServer.AirCon[Integer.parseInt(UA[4])].direct == false && HomeServer.AirCon[Integer.parseInt(UA[4])].LastUse != 100) {

System.out.println("ユーザー" + HomeServer.AirCon[Integer.parseInt(UA[4])].LastUse + ": ネットワーク操作失敗 エアコン" + HomeServer.AirCon[Integer.parseInt(UA[4])].id);

HomeServer.User[HomeServer.AirCon[Integer.parseInt(UA[4])].LastUse].displeasure();

HomeServer.AirCon[Integer.parseInt(UA[4])].webcancel++;

HomeServer.AirCon[Integer.parseInt(UA[4])].LastUse = 100;
                                                }
                                                if (UA[5].equals("オン")) {
                                                        ACNo = Integer.parseInt(UA[4]);
                                                }
                                                else {
                                                        ACNo = 99;
                                                }

HomeServer.Registration(UA[10], UA[3], UA[4], UA[5], UA[6], UA[7], UA[8], name, placename, id);
                                        }
                                }
                                if (UA[3].equals("lock") && UA[5].equals("施錠") && HsStatus.LockB == false) {
                                        for (int x = 0; x < HsStatus.ACnum; x++) {
                                                if (Sensor.RoomNum[x] > 0) {

zaitaku++;
                                                }
                                        }
                                        if (zaitaku == 0 && key == true) {
                                                System.out.println("ユーザー" + id + ": 家に誰もいなくなるので施錠して行きます。");

HomeServer.Registration(UA[10], UA[3], UA[4], UA[5], UA[6], UA[7], UA[8], name, placename, id);
                                        }
                                        else if (zaitaku == 0) {
                                                System.out.println("ユーザー" + id + ": 家に誰もいなくなるので鍵を持って施錠して行きます。");

HomeServer.Registration(UA[10], UA[3], UA[4], UA[5], UA[6], UA[7], UA[8], name, placename, id);
                                                key = true;
                                        }
                                        else if (zaitaku >= 1) {
                                                System.out.println("ユーザー" + id + ": 家族が家にいるのでそのまま出ます。");
                                        }
                                        zaitaku = 0;
                                }
                                else if (UA[3].equals("lock") && UA[5].equals("開錠")) {
                                        if (HsStatus.LockB == true) {

HomeServer.Registration(UA[10], UA[3], UA[4], UA[5], UA[6], UA[7], UA[8], name, placename, id);
                                        }
                                        if (oldplace != HsStatus.ACnum && UA[6].equals("0")) {
                                                System.out.println("ユーザー" + id + ": 鍵を持たず出ます。");
                                                key = false;

System.out.println();
                                        }
                                        else if (oldplace != HsStatus.ACnum && UA[6].equals("1")) {
                                                System.out.println("ユーザー" + id + ": 鍵を持って出ます。");
                                                key = true;

System.out.println();
                                        }
                                        else if (oldplace != HsStatus.ACnum && UA[6].equals("2")) {
                                                for (int x = 0; x < HsStatus.ACnum; x++) {
                                                        if (Sensor.RoomNum[x] > 0) {

zaitaku++;
                                                        }
                                                }
                                                if (zaitaku < 2) {

System.out.println("ユーザー" + id + ": 家に誰もいなくなるので鍵を持って出ます。");
                                                        key = true;
                                                }
                                                else if (zaitaku >= 2) {

System.out.println("ユーザー" + id + ": 家族が家にいるので鍵を持たずに出ます。");
                                                        key = false;
                                                }
```

```
                System.out.println();
                                                zaitaku = 0;
                                        }
                                }
                        }
                }
                Anxious = HomeServer.FtECheck();
                HomeServer.FtE(true);
                ran = Math.random();
                if (ran < memory && Anxious == false && (HsStatus.TimeM == 0
|| HsStatus.TimeM % HsStatus.Setuden == 0)) {
                        try {
                                Class.forName(HsStatus.Driver);
                                con = DriverManager.getConnection(url,
user, pass);
                                stmt = con.createStatement();
                                TimeNow = HsStatus.TimeH * 60 +
HsStatus.TimeM;
                                for (int x = 0; x < HsStatus.ACnum; x++) {
                                        if (ACNo != x) {
                                                if (HsStatus.history
== true) {
                                                        sql =
"select count(*) from ac" + x;
                                                        rs =
stmt.executeQuery(sql);
                rs.next();
                                                        int
gyousu = rs.getInt(1);
                                                        if
(gyousu >= 1) {
                sql = "select * from ac" + x + " order by 実時間 desc limit 2";
                rs = stmt.executeQuery(sql);
                rs.next();
                for (int y = 0; y < 2; y++) {
                        jikan[y] = rs.getString("時間");
                        dengen[y] = rs.getString("電源");
                        sousa[y] = rs.getString("操作者");
                        SQLTime[y] = jikan[y].split(":");
                        SQLTimeC[y] = Integer.parseInt(SQLTime[y][0]) * 60 +
Integer.parseInt(SQLTime[y][1]);
                        if (gyousu == 1) {
                                break;
                        }
                        if (y == 0 && gyousu > 1) {
                                rs.next();
                        }
                }
        }
        if (sousa[0].equals("直接")) {
                Method = "直接";
        }
        else {
                Method = "WEB";
        }
                                        }
                                }
                                // 自宅にいて部屋
に人がいない
                                if (place !=
HsStatus.ACnum && Sensor.RoomNum[x] == 0) {
                                        TargetR
= "部屋" + x;
                                        if (x ==
HsStatus.ACnum - 1) {
        TargetR = "リビング";
                                        }
                                        // 既存
技術
                                        if
(HsStatus.history == false && HsStatus.Psensor == false &&
HomeServer.AirCon[x].DengenPresent == 1) {
        System.out.println("ユーザー" + id + ": 移動 " + placename + " →
" + TargetR);
        System.out.println();
        System.out.println("ユーザー" + id + ": 部屋に誰もいない。エア
コン" + x + " 止めます。");
        System.out.println();
        if (HomeServer.AirCon[x].direct == false &&
HomeServer.AirCon[x].LastUse != 100) {
                System.out.println("ユーザー" +
HomeServer.AirCon[x].LastUse + ": ネットワーク操作失敗 エアコン" +
HomeServer.AirCon[x].id);
        HomeServer.User[HomeServer.AirCon[x].LastUse].displeasure();
                HomeServer.AirCon[x].webcancel++;
                HomeServer.AirCon[x].LastUse = 100;
        }
        HomeServer.Registration("直接", "ac", Integer.toString(x), "オフ",
"0", "0", "0", name, placename, id);
        System.out.println("ユーザー" + id + ": 元の部屋へ " + TargetR + "
→ " + placename);
        System.out.println();
```

```
                }

                // 履歴有りセンサー無し

                else if (HsStatus.history == true && HsStatus.Psensor == false && TimeNow - SQLTimeC[0] > HsStatus.Setuden && dengen[0].equals("オン") && Method.equals("直接")) {

                    System.out.println("ユーザー" + id + ": 移動 " + placename + " → " + TargetR);

                    System.out.println();

                    System.out.println("ユーザー" + id + ": 最後の操作から" + (TimeNow - SQLTimeC[0]) + "分以上経過。部屋に誰もいない。エアコン" + x + " 止めます。");

                    System.out.println();

                    HomeServer.Registration("直接", "ac", Integer.toString(x), "オフ", "0", "0", "0", name, placename, id);

                    System.out.println("ユーザー" + id + ": 元の部屋へ " + TargetR + " → " + placename);

                    System.out.println();

                }

                else if (HsStatus.history == true && HsStatus.Psensor == false && TimeNow - SQLTimeC[0] > HsStatus.WebSetuden && dengen[0].equals("オン") && Method.equals("WEB")) {

                    System.out.println("ユーザー" + id + ": 移動 " + placename + " → " + TargetR);

                    System.out.println();

                    System.out.println("ユーザー" + id + ": " + sousa[0] + "のネットワーク操作から" + (TimeNow - SQLTimeC[0]) + "分以上経過。部屋に誰もいない。節電のためエアコン" + x + " 止めます。");

                    System.out.println();

                    HomeServer.Registration("直接", "ac", Integer.toString(x), "オフ", "0", "0", "0", name, placename, id);

                    System.out.println("ユーザー" + id + ": 元の部屋へ " + TargetR + " → " + placename);

                    System.out.println();

                }

                // 履歴無しセンサー有り

                else if (HsStatus.history == false && HsStatus.Psensor == true && HomeServer.AirCon[x].direct == true && HomeServer.AirCon[x].DengenPresent == 1) {

                    System.out.println("ユーザー" + id + ": 移動 " + placename + " → " + TargetR);

                    System.out.println();

                    System.out.println("ユーザー" + id + ": 通常リモコンで操作されているのに部屋に誰もいない。エアコン" + x + " 止めます。");

                    System.out.println();

                    HomeServer.Registration("直接", "ac", Integer.toString(x), "オフ", "0", "0", "0", name, TargetR, id);

                    System.out.println("ユーザー" + id + ": 元の部屋へ " + TargetR + " → " + placename);

                    System.out.println();

                }

                // 履歴有りセンサー有り

                else if (HsStatus.history == true && HsStatus.Psensor == true && TimeNow - SQLTimeC[0] > HsStatus.Setuden && dengen[0].equals("オン") && Method.equals("直接")) {

                    System.out.println("ユーザー" + id + ": 移動 " + placename + " → " + TargetR);

                    System.out.println();

                    System.out.println("ユーザー" + id + ": 最後の操作から" + (TimeNow - SQLTimeC[0]) + "分以上経過。部屋に誰もいない。エアコン" + x + " 止めます。");

                    System.out.println();

                    HomeServer.Registration("直接", "ac", Integer.toString(x), "オフ", "0", "0", "0", name, placename, id);

                    System.out.println("ユーザー" + id + ": 元の部屋へ " + TargetR + " → " + placename);

                    System.out.println();

                }

                else if (HsStatus.history == true && HsStatus.Psensor == true && TimeNow - SQLTimeC[0] > HsStatus.WebSetuden && dengen[0].equals("オン") && Method.equals("WEB")) {

                    System.out.println("ユーザー" + id + ": 移動 " + placename + " → " + TargetR);

                    System.out.println();

                    System.out.println("ユーザー" + id + ": " + sousa[0] + "の最後のネットワーク操作から" + (TimeNow - SQLTimeC[0]) + "分以上経過。部屋に誰もいない。節電のためエアコン" + x + " 止めます。");

                    System.out.println();

                    HomeServer.Registration("直接", "ac", Integer.toString(x), "オフ", "0", "0", "0", name, placename, id);

                    System.out.println("ユーザー" + id + ": 元の部屋へ " + TargetR + " → " + placename);

                    System.out.println();

                }
            }

            // 自宅にいて部屋に人がいない 終わり

            // 外出先からの操作

            if (place == HsStatus.ACnum) {

                // 既存技術

                if (HsStatus.history == false && HsStatus.Psensor == false && HomeServer.AirCon[x].DengenPresent == 1) {

                    System.out.println("ユーザー" + id + ": エアコン" + x + " 動作中。判断できないので放置します。");
```

```
                System.out.println();

                                                }

                                                // 履歴有りセンサー無し

                                                else if (HsStatus.history == true && HsStatus.Psensor == false && TimeNow - SQLTimeC[0] > HsStatus.Setuden && dengen[0].equals("オン") && Method.equals("直接")) {

                if (dengen[0].equals("オン") && dengen[1].equals("オフ")) {

                                System.out.println("ユーザー" + id + ": 電源オンから" + (TimeNow - SQLTimeC[0]) + "分以上経過。消し忘れかもしれない。エアコン" + x + " 止めます。");

                                System.out.println();

                                HomeServer.Registration("WEB", "ac", Integer.toString(x), "オフ", "0", "0", "0", name, placename, id);

                                System.out.println();

                }

                else if (dengen[0].equals("オン") && dengen[1].equals("オン")) {

                                if (TimeNow > SQLTimeC[0] + ((SQLTimeC[0] - SQLTimeC[1]) * 2 + HsStatus.Setuden)) {

                                                System.out.println("ユーザー" + id + ": 最後の操作から" + ((SQLTimeC[0] - SQLTimeC[1]) * 2 + HsStatus.Setuden) + "分以上経過。もう誰もいなくて、消し忘れかもしれない。エアコン" + x + " 止めます。");

                                                System.out.println();

                                                HomeServer.Registration("WEB", "ac", Integer.toString(x), "オフ", "0", "0", "0", name, placename, id);

                                                System.out.println();

                                }

                }

                                                }

                                                // 履歴無しセンサー有り

                                                else if (HsStatus.history == false && HsStatus.Psensor == true && Sensor.RoomNum[x] == 0 && HomeServer.AirCon[x].DengenPresent == 1) {

                System.out.println("ユーザー" + id + ": 部屋に誰もいないようだ。エアコン" + x + " 止めます。");

                System.out.println();

                if (HomeServer.AirCon[x].direct == false && HomeServer.AirCon[x].LastUse != 100) {

                                System.out.println("ユーザー" + HomeServer.AirCon[x].LastUse + ": ネットワーク操作失敗 エアコン" + HomeServer.AirCon[x].id);

                HomeServer.User[HomeServer.AirCon[x].LastUse].displeasure();

                                HomeServer.AirCon[x].webcancel++;

                                HomeServer.AirCon[x].LastUse = 100;

                }

                HomeServer.Registration("WEB", "ac", Integer.toString(x), "オフ", "0", "0", "0", name, placename, id);

                System.out.println();

                                                }

                                                // 履歴有りセンサー有り

                                                else if (HsStatus.history == true && HsStatus.Psensor == true && Sensor.RoomNum[x] == 0 && TimeNow - SQLTimeC[0] > HsStatus.Setuden && dengen[0].equals("オン") && Method.equals("直接")) {

                System.out.println("ユーザー" + id + ": 最後の直接操作から" + (TimeNow - SQLTimeC[0]) + "分以上経過。部屋に誰もいないようだ。エアコン" + x + " 止めます。");

                System.out.println();

                HomeServer.Registration("WEB", "ac", Integer.toString(x), "オフ", "0", "0", "0", name, placename, id);

                System.out.println();

                                                }

                                                else if (HsStatus.history == true && HsStatus.Psensor == true && Sensor.RoomNum[x] == 0 && TimeNow - SQLTimeC[0] > HsStatus.WebSetuden && dengen[0].equals("オン") && Method.equals("WEB")) {

                System.out.println("ユーザー" + id + ": 最後のネットワーク操作から" + (TimeNow - SQLTimeC[0]) + "分以上経過。部屋に誰もいないようだ。節電のためエアコン" + x + " 止めます。");

                System.out.println();

                HomeServer.Registration("WEB", "ac", Integer.toString(x), "オフ", "0", "0", "0", name, placename, id);

                System.out.println();

                                                }

                                }

                }

                // 玄関錠処理

                // 外出先からの操作

                if (place == HsStatus.ACnum) {

                                if (HsStatus.history == true) {

                                                sql = "select count(*) from lock0";

                                                rs = stmt.executeQuery(sql);

                                                rs.next();

                                                if (rs.getInt(1) > 0) {

                                                                sql = "select * from lock0 order by 実時間 desc limit 1";

                                                                rs = stmt.executeQuery(sql);

                rs.next();

                LockTime = rs.getString("時間");

                LockSQLTime = LockTime.split(":");
```

```
LockSQLTimeC = Integer.parseInt(LockSQLTime[0]) * 60 + Integer.parseInt(LockSQLTime[1]);
Lock = rs.getString("状態");
}
}
// 既存技術
if (HsStatus.history == false && HsStatus.Psensor == false && HsStatus.LockB == false && key == true) {
System.out.println("ユーザー" + id + ": 玄関錠が開いている。防犯のため施錠します。");
System.out.println();
HomeServer.Registration("WEB", "lock", "0", "施錠", "0", "0", "0", name, placename, id);
System.out.println();
}
// 履歴有りセンサー無し
else if (HsStatus.history == true && HsStatus.Psensor == false && TimeNow - LockSQLTimeC > HsStatus.Setuden && Lock.equals("開錠") && key == true) {
System.out.println("ユーザー" + id + ": 玄関錠が開いたまま" + (TimeNow - LockSQLTimeC) + "分以上経過。閉め忘れかもしれない。防犯のため施錠します。");
System.out.println();
HomeServer.Registration("WEB", "lock", "0", "施錠", "0", "0", "0", name, placename, id);
System.out.println();
}
for (int x = 0; x < HsStatus.ACnum; x++) {
// 履歴無しセンサー有り
if (HsStatus.history == false && HsStatus.Psensor == true && Sensor.RoomNum[x] == 0 && HsStatus.LockB == false && key == true) {
count++;
if (count == HsStatus.ACnum) {
System.out.println("ユーザー" + id + ": 自宅に誰もいないようだ。なのに玄関錠が開いている。防犯のため施錠します");
System.out.println();
HomeServer.Registration("WEB", "lock", "0", "施錠", "0", "0", "0", name, placename, id);
System.out.println();
}
}
// 履歴有りセンサー有り
else if (HsStatus.history == true && HsStatus.Psensor == true && TimeNow - LockSQLTimeC > HsStatus.Setuden && Sensor.RoomNum[x] == 0 && Lock.equals("開錠") && key == true) {
count++;
if (count == HsStatus.ACnum) {
System.out.println("ユーザー" + id + ": 玄関錠が開いたまま" + (TimeNow - LockSQLTimeC) + "分以上経過。自宅に誰もいないようだ。防犯のため施錠します。");
System.out.println();
HomeServer.Registration("WEB", "lock", "0", "施錠", "0", "0", "0", name, placename, id);
System.out.println();
}
}
}
// 切断
stmt.close();
con.close();
}
catch (Exception e) {
e.printStackTrace();
}
}
mm = mm + HsStatus.EveryTime;
if (mm == 60) {
mm = 0;
hh = hh + 1;
}
shutout = false;
BR = true;
count = 0;
break;
}
else {
}
}
br.close();}
catch (FileNotFoundException e) {
e.printStackTrace();}
catch (IOException e) {
e.printStackTrace(); }
}
void SetInfo(int i) {
id = i;
place = i;
placename = "部屋" + i + "(自室)";
}
void SetMemory(double d) {
memory = d;
}
}
```

MySQL テーブル構成及びデータ挿入形式

実時間はデータ抽出時に用いる。

エアコン(ac0 はエアコン番号 0 を表す)

```
create table ac0(
時間 text not null,
操作者 text not null,
電源 text not null,
運転 text,
温度 text,
風量 text,
実時間 timestamp NOT NULL default CURRENT_TIMESTAMP on update CURRENT_TIMESTAM);

insert into ac0
(時間,操作者,電源,運転,温度,風量,実時間)
values('0:00','直接','オフ','暖房','20','自動',NULL);
```

玄関錠

```
create table lock0(
時間 text not null,
操作者 text not null,
状態 text not null,
実時間 timestamp NOT NULL default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP
);

insert into lock0
(時間,操作者,状態,実時間)
values('0:00','直接','施錠',NULL);
```