（付録：プログラム主要部）

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Microsoft.VisualBasic.FileIO;
using System.Collections;
using System.IO;
using Excel = Microsoft.Office.Interop.Excel;
using System.Runtime.InteropServices;
namespace JungCsvChecker
{
  public partial class Form1 : Form
  {
    /// <summary>
    /// CSVデータを格納するArrayList（多次元配列）
    /// [0]日時
    /// [1]センサー番号（2桁）
    /// [2]センサーデータ[ArrayList][60分X24時間=144列][10進数]
    /// [3]1日の合計値（int型）
    /// [4]1時間毎の合計値[ArrayList]
    /// [5]30分毎
    /// [6]10分毎
    /// [7]5分毎
    /// [8]1日の回数
    /// [9]日中の回数
    /// [10]夜間の回数
    /// [11]起床時間
    /// [12]就寝時間
    /// [13]外出回数
    /// [14]外出合計時間
    /// </summary>
    public ArrayList aListCsvData;
    public int[,] aAveDweek; // 日中の曜日毎平均を格納する配列 [センサー番号,曜日番号]
    public int[,] aAveNweek; // 夜間の曜日毎平均を格納する配列 [センサー番号,曜日番号]
    public int[,] aAveAweek; // 1日の曜日毎平均を格納する配列 [センサー番号,曜日番号]
    public int[,] aAveWakeupWeek; //起床
    public int[,] aAveSleepWeek; //就寝
    public int[,] aAveOutdoorCntWeek; //外出回数
    public int[,] aAveOutdoorTimeWeek; //外出時間
    public string[,] arraySencer;
    public int intSencerCount=0;
    public ArrayList alSencer;  //センサー
    public int intSencerThreshold = 3; // 回数をカウントする閾値
    public int flgOutdoorTime = 30;  // この時間すべてのセンサーに反応がなければ外出とする
    public int cntFilter = 2; //連続データのﾌｨﾙﾀｰ 少ない連続は無いものとする
    public System.IO.Stream stream;
    public string configfilename = "";
    public string readCsvFileName = "";
    public string readCsvFilePath = "";
    public bool reanalysis = false;
    public ArrayList originalSencerData;
    public Form1()
    {
      InitializeComponent();
      this.clearFormData();
      this.initFormData();
    }
    /// <summary>
    /// フォームデータのイニシャライズ
    /// </summary>
    ///
    private void initFormData()
    {

      //閾値のコンボボックスをセット（データが16進数で1-16まであるため1-16をセット）
      for (int i = 0; i < 16; i++)
      {
        this.comboBox7.Items.Add(i);
      }
      this.comboBox7.SelectedIndex = 3;
      //時間のコンボボックスをセット
      for (int i = 0; i < 60; i++)
      {
        if (i < 24)
        {
          this.comboBox8.Items.Add(i);
          this.comboBox10.Items.Add(i);
          this.comboBox16.Items.Add(i);
          this.comboBox18.Items.Add(i);
          this.comboBox20.Items.Add(i);
          this.comboBox22.Items.Add(i);
        }
        if (i < 3)
        {
          this.comboBox15.Items.Add(i+1);
        }
        this.comboBox9.Items.Add(i);
        this.comboBox11.Items.Add(i);
        this.comboBox14.Items.Add(i+1);
        this.comboBox17.Items.Add(i);
        this.comboBox19.Items.Add(i);
        this.comboBox21.Items.Add(i);
        this.comboBox23.Items.Add(i);
      }
      // 日中・夜間の時間をセット（06:00-20:59）
      this.comboBox8.SelectedIndex   = 6;
      this.comboBox9.SelectedIndex   = 0;
      this.comboBox10.SelectedIndex = 20;
      this.comboBox11.SelectedIndex = 59;
      this.comboBox14.SelectedIndex = 29;
      this.comboBox15.SelectedIndex = 1;
      this.comboBox16.SelectedIndex = 3;
      this.comboBox17.SelectedIndex = 0;
      this.comboBox18.SelectedIndex = 11;
      this.comboBox19.SelectedIndex = 59;
      this.comboBox20.SelectedIndex = 18;
      this.comboBox21.SelectedIndex = 0;
      this.comboBox22.SelectedIndex = 3;
      this.comboBox23.SelectedIndex = 0;
    }
    /// <summary>
    /// フォームデータのクリア
    /// 新しいデータファイル読み込み時に現在の設定をクリアする
    /// </summary>
    ///
    private void clearFormData()
    {

      // コンボボックスをクリア
      this.comboBox1.Items.Clear();
      this.comboBox2.Items.Clear();
      this.comboBox3.Items.Clear();
```

```
        this.comboBox4.Items.Clear();
        this.comboBox5.Items.Clear();
        this.comboBox6.Items.Clear();
        this.comboBox12.Items.Clear();
        this.comboBox13.Items.Clear();
        this.comboBox1.Items.Add("--");
        this.comboBox2.Items.Add("--");
        this.comboBox3.Items.Add("--");
        this.comboBox4.Items.Add("--");
        this.comboBox5.Items.Add("--");
        this.comboBox6.Items.Add("--");
        this.comboBox12.Items.Add("--");
        this.comboBox13.Items.Add("--");
        this.comboBox1.SelectedIndex = 0;
        this.comboBox2.SelectedIndex = 0;
        this.comboBox3.SelectedIndex = 0;
        this.comboBox4.SelectedIndex = 0;
        this.comboBox5.SelectedIndex = 0;
        this.comboBox6.SelectedIndex = 0;
        this.comboBox12.SelectedIndex = 0;
        this.comboBox13.SelectedIndex = 0;
        this.intSencerCount =0;
    }
    /// <summary>
    /// センサー名の登録
    /// </summary>
    /// <param name="filpath">設定ファイルのパス</param>
    private void setSencerName(string filepath)
    {

        if (System.IO.File.Exists(filepath))
        {
            // センサーの2次元配列
            this.arraySencer = new string[8, 2]{
                {"",""},
                {"",""},
                {"",""},
                {"",""},
                {"",""},
                {"",""},
                {"",""},
                {"",""}
            };
            TextFieldParser parser = new TextFieldParser(filepath,
System.Text.Encoding.GetEncoding("Shift_JIS"));
            using (parser)
            {
                parser.TextFieldType = FieldType.Delimited;
                parser.SetDelimiters(","); // 区切り文字はコンマ
                while (!parser.EndOfData)
                {
                    string[] row = parser.ReadFields(); // 1行読み込み
                    this.alSencer = new ArrayList();
                    int i = 0;
                    foreach (string field in row)
                    {
                        string f = field;
                        this.arraySencer[i, 0] = f;
                        this.arraySencer[i, 1] = "0"+(i+1).ToString();
                        this.alSencer.Add("0" + (i + 1).ToString());
                        this.intSencerCount++;
                        i++;
                    }
                    //Console.Write(this.intSencerCount);
                }

            }

            //ラベルをセット
            this.label1.Text = this.arraySencer[0, 0];
            this.label2.Text = this.arraySencer[1, 0];
            this.label3.Text = this.arraySencer[2, 0];
            this.label4.Text = this.arraySencer[3, 0];
            this.label5.Text = this.arraySencer[4, 0];
            this.label6.Text = this.arraySencer[5, 0];
            this.label13.Text = this.arraySencer[6, 0];
            this.label14.Text = this.arraySencer[7, 0];
            // 各ファイルの設定ファイルの場合はコンボボックスをセット
            if (this.configfilename != "")
            {
                this.comboBox1.Items.Add(this.arraySencer[0, 1]);
                this.comboBox1.SelectedIndex = 1;
                this.comboBox2.Items.Add(this.arraySencer[1, 1]);
                this.comboBox2.SelectedIndex = 1;
                this.comboBox3.Items.Add(this.arraySencer[2, 1]);
                this.comboBox3.SelectedIndex = 1;
                this.comboBox4.Items.Add(this.arraySencer[3, 1]);
                this.comboBox4.SelectedIndex = 1;
                this.comboBox5.Items.Add(this.arraySencer[4, 1]);
                this.comboBox5.SelectedIndex = 1;
                this.comboBox6.Items.Add(this.arraySencer[5, 1]);
                this.comboBox6.SelectedIndex = 1;
                this.comboBox12.Items.Add(this.arraySencer[6, 1]);
                this.comboBox12.SelectedIndex = 1;
                this.comboBox13.Items.Add(this.arraySencer[7, 1]);
                this.comboBox13.SelectedIndex = 1;
            }
        }
    }
    /// <summary>
    /// 「開く」メニュークリック時の処理(ファイル選択ダイアログを開
く)
    /// <param name="sender">IDEによる自動育成</param>
    /// <param name="e">IDEによる自動育成</param>
    /// </summary>
    ///
    private void OpenCsvToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        OpenFileDialog ofd = new OpenFileDialog();

        //[ファイルの種類]に表示される選択肢を指定する
        ofd.Filter = "csvファイル(*.csv;*.txt)|*.csv;*.txt|すべてのファイル
(*.*)|*.*";
        //ダイアログを表示する
        if (ofd.ShowDialog() == DialogResult.OK)
        {
            //OKボタンがクリックされたとき、選択されたファイルを読み取
り専用で開く
            this.stream = ofd.OpenFile();
            if (this.stream != null)
            {
                this.readCsvFilePath = ofd.FileName;
                this.readCsvFileName = Path.GetFileName(ofd.FileName);
                this.clearFormData();
                //  configフォルダに、開くファイルに該当する設定ファイル
があるかチェック
                string filename = Path.GetFileName(ofd.FileName);
                filename = filename.Replace(".csv", ".config");
                string stCurrentDir =
System.IO.Directory.GetCurrentDirectory();
                string initConfigFile = stCurrentDir + "¥¥config¥¥" + filename;
                if (System.IO.File.Exists(initConfigFile))
                {
                    this.configfilename = filename;
```

```
                this.setSencerName(initConfigFile);
                this.ReadCsvFile(this.stream);
            }
            else
            {
                //メッセージボックスを表示する
                MessageBox.Show("設定ファイルが見つかりません。",
                    "エラー",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
        }
        this.stream.Close();
    }
}
/// <summary>
/// CSVファイルの読み込み処理("日時","センサー番号","センサーデータ"の3列になっている)
/// <param name="stream">読み込むCSVファイルのSystem.IO.Stream</param>
/// </summary>
///
private void ReadCsvFile(System.IO.Stream stream2)
{
    if (this.aListCsvData != null)
    {
        this.aListCsvData.Clear();
    }
    this.aListCsvData = new ArrayList();  // 読み込んだCSVデータを格納するArrayList のインスタンス
    TextFieldParser parser = new TextFieldParser(stream2, Encoding.GetEncoding("Shift_JIS"));
    using (parser)
    {
        parser.TextFieldType = FieldType.Delimited;
        parser.SetDelimiters(",");// 区切り文字はコンマ
        try
        {
            while (!parser.EndOfData)
            {
                string[] row = parser.ReadFields();//1行読み込み
                // 読み込んだ行の処理
                // ArrayListに列の値を格納して、行のArrayListに格納する(ArrayListの二次元配列)
                ArrayList tmpAL = new ArrayList();
                foreach (string field in row)
                {
                    string f = field;
                    tmpAL.Add(f);
                }
                //列のArrayList を行のArrayListに格納
                this.aListCsvData.Add(tmpAL);
            }
            //センサーデータを10進数にして配列に格納
            this.AnalysisSencerData();
        }
        catch (Exception ex)
        {
            Console.Write(ex.ToString());
        }
    }
}
/// <summary>
/// センサーデータを10進数にして配列に格納
/// <param name="aListCsv">読み込んだCSVデータを格納しているArrayListの2次元配列</param>
/// </summary>
///
private void AnalysisSencerData()
{

    // aListCsv[0][3]に、センサーデータをArrayListにして格納
    for (int i = 0; i < this.aListCsvData.Count; i++)
    {
        //センサーデータを格納するArrayList
        ArrayList aListSencerData = new ArrayList();
        //5分毎のセンサーデータの合計を格納するArrayList
        ArrayList aList5minDataPerHour = new ArrayList();
        //10分毎のセンサーデータの合計を格納するArrayList
        ArrayList aList10minDataPerHour = new ArrayList();
        //30分毎のセンサーデータの合計を格納するArrayList
        ArrayList aList30minDataPerHour = new ArrayList();
        //1時間ごとのセンサーデータの合計を格納するArrayList
        ArrayList aListHourDataPerHour = new ArrayList();
        //1日の合計値
        int SumDay = 0;
        int cnt1min  = 0; //1分
        int cnt5min  = 0; //5分
        int cnt10min = 0; //10分
        int cnt30min = 0; //30分
        int cnt1hour = 0; //1時間
        // CSV3列目にセンサーのデータが格納されている
        string strData = ((ArrayList)this.aListCsvData[i])[2].ToString();
        // センサーデータを1文字ずつ配列に格納
        for (int j = 0; j < strData.Length; j++)
        {
            // センサーデータを1文字ずつ取り出し
            string strNum16 = strData.Substring(j, 1);
            //16進数を10進数に変換する
            int num = Convert.ToInt32(strNum16, 16);
            // 配列に格納
            aListSencerData.Add(num);
            //1日の合計値を計算
            SumDay += num;
            cnt1min  += num;
            cnt5min  += num;
            cnt10min += num;
            cnt30min += num;
            cnt1hour += num;
            // 1時間ごとの合計値を計算
            if ((j + 1) % 60 == 0)
            {
                aListHourDataPerHour.Add(cnt1hour);
                cnt1hour = 0;
            }
            //30分ごとの合計値を計算
            if ((j + 1) % 30 == 0)
            {
                aList30minDataPerHour.Add(cnt30min);
                cnt30min = 0;
            }
            //10分ごとの合計値を計算
            if ((j + 1) % 10 == 0)
            {
                aList10minDataPerHour.Add(cnt10min);
                cnt10min = 0;
            }
            //5分ごとの合計値を計算
            if ((j + 1) % 5 == 0)
            {
                aList5minDataPerHour.Add(cnt5min);
                cnt5min = 0;
            }
        }
```

```
            //3列目のセンサーデータの列を削除
            ((ArrayList)this.aListCsvData[i]).RemoveAt(2);
            //3列目にセンサーデータのArrayListを追加する
            ((ArrayList)this.aListCsvData[i]).Add(aListSencerData);
            //4列目に1日の合計値
            ((ArrayList)this.aListCsvData[i]).Add(SumDay);
            //5列目に1時間毎の合計値のArrayList
            ((ArrayList)this.aListCsvData[i]).Add(aListHourDataPerHour);
            //6列目に30分毎の合計値のArrayList
            ((ArrayList)this.aListCsvData[i]).Add(aList30minDataPerHour);
            //7列目に10分毎の合計値のArrayList
            ((ArrayList)this.aListCsvData[i]).Add(aList10minDataPerHour);
            //8列目に5分毎の合計値のArrayList
            ((ArrayList)this.aListCsvData[i]).Add(aList5minDataPerHour);
            //9列目に1日の回数をいれる 0 を入れる
            ((ArrayList)this.aListCsvData[i]).Add(0);
            //10列目に日中の回数
            ((ArrayList)this.aListCsvData[i]).Add(0);
            //11列目に夜間の回数
            ((ArrayList)this.aListCsvData[i]).Add(0);
            // [11]起床時間
            ((ArrayList)this.aListCsvData[i]).Add("00:00");
            // [12]就寝時間
            ((ArrayList)this.aListCsvData[i]).Add("00:00");
            // [13]外出回数
            ((ArrayList)this.aListCsvData[i]).Add(0);
            // [14]外出合計時間
            ((ArrayList)this.aListCsvData[i]).Add("00:00");
        }
    }
    /// <summary>
    /// 「解析スタート」ボタンを落下
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void button1_Click(object sender, EventArgs e)
    {
        if (this.readCsvFilePath != "")
        {
            this.stream = File.OpenRead(this.readCsvFilePath);
            this.ReadCsvFile(this.stream);
        }
        if (this.aListCsvData != null)
        {
            this.analysisCsvData();
        }
        else
        {
            //メッセージボックスを表示する
            MessageBox.Show("CSVファイルを読み込んでください",
                "エラー",
                MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
    }
    /// <summary>
    /// CSVデータを解析する
    /// </summary>
    ///
    private void analysisCsvData()
    {
        try
        {
            // 回数をカウントする(Aに反応があったら、A以外に反応が出
たときにAに1をカウント)
            // ループの順番
            // -日付
            //   -センサーデータ ((ArrayList)this.aListCsvData[i]).Count
            //      -センサー番号
            //
            string tmpDay = ((ArrayList)this.aListCsvData[0])[0].ToString();
            string tmpSencernum =
((ArrayList)this.aListCsvData[0])[1].ToString();
            // センサーデータ
            ArrayList alcsv = (ArrayList)((ArrayList)this.aListCsvData[0])[2];
            bool[] flg = new bool[this.intSencerCount];
            int[] tmpj = new int[this.intSencerCount];
            for (int i = 0; i < this.intSencerCount; i++)
            {
                flg[i] = false;
                tmpj[i] = 0;
            }
            //日中データのセンサー番号範囲
            int intDaytimeS =
int.Parse(this.comboBox8.SelectedItem.ToString()) * 60 +
int.Parse(this.comboBox9.SelectedItem.ToString());
            int intDaytimeE =
int.Parse(this.comboBox10.SelectedItem.ToString()) * 60 +
int.Parse(this.comboBox11.SelectedItem.ToString());
            //就寝時間・起床時間のセンサー番号範囲
            int intWakeUptimeS =
int.Parse(this.comboBox16.SelectedItem.ToString()) * 60 +
int.Parse(this.comboBox17.SelectedItem.ToString());
            int intWakeUptimeE =
int.Parse(this.comboBox18.SelectedItem.ToString()) * 60 +
int.Parse(this.comboBox19.SelectedItem.ToString());
            int intSleeptimeS =
int.Parse(this.comboBox20.SelectedItem.ToString()) * 60 +
int.Parse(this.comboBox21.SelectedItem.ToString());
            int intSleeptimeE =
int.Parse(this.comboBox22.SelectedItem.ToString()) * 60 +
int.Parse(this.comboBox23.SelectedItem.ToString());
            // 合計値を格納しているカラムを0にする
            for (int i = 0; i < this.aListCsvData.Count; i++)
            {
                ((ArrayList)this.aListCsvData[i])[8] = 0;
                ((ArrayList)this.aListCsvData[i])[9] = 0;
                ((ArrayList)this.aListCsvData[i])[10] = 0;
            }
            // ★★センサーデータをフィルタリングする★★
            for (int i = 0; i < this.aListCsvData.Count; i++) // センサー番号
            {
                ArrayList tmpcsv =
(ArrayList)((ArrayList)this.aListCsvData[i])[2];
                int cntData = 0;
                // 01000,03000 とか数値が連続しない場合は0にする
                //int cntFilter = 2;
                for (int j = 0; j < tmpcsv.Count; j++) // センサー毎にカウント
                {
                    if (j > cntFilter)
                    {
                        if ((int)tmpcsv[j] > 0)
                        {
                            cntData += 1;
                        }
                        else
                        {
                            if (cntData <= cntFilter)
                            {
                                for (int f = 0; f < cntFilter; f++)
                                {
                                    tmpcsv[j - (f+1)] = 0;
                                }
```

```
                }
                cntData = 0;
              }
            }
          }
          ((ArrayList)this.aListCsvData[i])[2] = tmpcsv;
        }
        /*   ★★★ここが最も大切なデータ解析箇所★★★
         *
         *  「あるセンサーが反応後、別のセンサーが反応したらインクリメント」のために、少々複雑
         *
         */
        int cntOutdoor = 0;       // 外出回数を格納
        int cntSencer = 0;
        int cntSencerSum = 0;       // 反応しない時間の合計
        int  cntSencerFirst  =  0;    // 最初に反応したセンサーの位置（起床時間）
        int  cntSencerLast  =  0;     // 最後に反応したセンサーの位置（就寝時間）
        int  cntSencerFirst2  =  0;    // 最初に反応したセンサーの位置（起床時間）：前日に使うバッファ
        int  cntSencerLast2  =  0;     // 最後に反応したセンサーの位置（就寝時間）：前日に使うバッファ

        string strSencerFirst = "";
        string strSencerLast = "";
        // 一日毎に解析ループする
        for (int intDay = 0; intDay < this.aListCsvData.Count; intDay += this.intSencerCount) // 日毎
        {
          for (int i = 0; i < alcsv.Count - 1; i++) //センサーデータ
          {
            for (int j = intDay; j < intDay + this.intSencerCount; j++) // センサー番号
            {
              ArrayList tmpcsv = (ArrayList)((ArrayList)this.aListCsvData[j])[2];  //センサーデータを格納
              for (int z = 0; z < this.intSencerCount; z++) // センサー毎にカウント（センサーの数）
              {
                // 「あるセンサーが反応後に別のセンサーが反応したらカウント」のためにこの複雑な処理となる
                // ★分析するデータのセンサー番号のセンサー
                if ((String)((ArrayList)this.aListCsvData[j])[1] == (String)this.alSencer[z])
                {
                  // （既に反応がなく）センサーが反応したらフラグを立てる
                  if (flg[z] == false && (int)tmpcsv[i] > this.intSencerThreshold)
                  {
                    tmpj[z] = j;  // 実際のセンサー配列番号jをセンサー番号zに対応させる
                    flg[z] = true;  // センサーに反応あり
                  }
                  // [11]起床時間
                  // [12]就寝時間
                  // [13]外出回数
                  // [14]外出合計時間
                  // センサーが反応しない時間をカウントする
                  if ((int)tmpcsv[i] <= this.intSencerThreshold)
                  {
                    cntSencer++;
                    // 外出回数判定
                    if (cntSencer == flgOutdoorTime * this.intSencerCount)
                    {
                      cntOutdoor++;
                    }
                    // 外出時間判定
                    if (cntSencer >= flgOutdoorTime * this.intSencerCount)
                    {
                      cntSencerSum++;
                      //Console.Write(cntSencer + ":");
                    }
                  }
                  else // センサー反応!
                  {
                    //最初にセンサーが反応した時間（起床時間）
                    if (cntSencerFirst == 0)
                    {
                      // 起床時間範囲
                      if (i >= intWakeUptimeS && i <= intWakeUptimeE)
                      {
                        cntSencerFirst = i;
                        //strSencerFirst = this.arraySencer[z, 0]; //この処理は無視
                      }
                    }
                    // ここに残ったデータが最後に反応した時間（就寝時間）
                    //就寝時間範囲（当日夜）
                    if (intSleeptimeS <= i)
                    {
                      cntSencerLast = i;
                    }
                    //就寝時間範囲（前日の深夜の場合）
                    if (intSleeptimeE >= i)
                    {
                      cntSencerLast2 = i;
                    }
                    //strSencerLast = this.arraySencer[z, 0]; //この処理は無視
                    cntSencer = 0;
                  }
                }
                // ★分析するデータのセンサー番号以外のセンサー。
                //   他のセンサーに既に反応があり、自分に反応があった場合インクリメント！
                else
                {
                  // センサー反応後、別のセンサーが反応したらインクリメント
                  if (flg[z] == true && (int)tmpcsv[i] > this.intSencerThreshold)
                  {
                    // ★反応合計値をインクリメント
                    // 全体
                    int x = (int)((ArrayList)this.aListCsvData[tmpj[z]])[8] + 1;
                    ((ArrayList)this.aListCsvData[tmpj[z]])[8] = x;
                    //日中データの場合
                    if (i >= intDaytimeS && i <= intDaytimeE)
                    {
                      int y = (int)((ArrayList)this.aListCsvData[tmpj[z]])[9] + 1;
                      ((ArrayList)this.aListCsvData[tmpj[z]])[9] = y;
                    }
                    //夜間データの場合
                    else
                    {
```

```
                        int y =
(int)((ArrayList)this.aListCsvData[tmpj[z]])[10] + 1;
                        ((ArrayList)this.aListCsvData[tmpj[z]])[10] = y;
                      }
                      flg[z] = false;
                    }
                  }
                }
              }
            }
            // [11]起床時間
            ((ArrayList)this.aListCsvData[intDay])[11] = cntSencerFirst;
            // [12]就寝時間
            ((ArrayList)this.aListCsvData[intDay])[12] = cntSencerLast;
            if (cntSencerLast2 > 0 && intDay>this.intSencerCount)
            {
              ((ArrayList)this.aListCsvData[intDay -
this.intSencerCount])[12] = cntSencerLast2 + 1440;
            }
            // [13]外出回数
            cntOutdoor = cntOutdoor - 2; // 就寝回数を一回として(起床
までと修身からの)2回をマイナス
            if (cntOutdoor < 0) cntOutdoor = 0;
            ((ArrayList)this.aListCsvData[intDay])[13] = cntOutdoor;
            //起きている時間
            int waketime = 0;
            if ((int)((ArrayList)this.aListCsvData[intDay])[12] > 0 &&
(int)((ArrayList)this.aListCsvData[intDay])[11] > 0)
            {
              waketime = 1440 -
(int)((ArrayList)this.aListCsvData[intDay])[12] -
(int)((ArrayList)this.aListCsvData[intDay])[11];
              // [14]外出合計時間
              cntSencerSum = (cntSencerSum / intSencerCount) + 1; // セ
ンサーの数で割る
              cntSencerSum = cntSencerSum - waketime; //睡眠時間を
マイナス
              cntSencerSum = cntSencerSum + cntOutdoor *
flgOutdoorTime; // 外出時間と判定される時間をプラス
              if (cntSencerSum < 0) cntSencerSum = 0;
            }
            else
            {
              cntSencerSum = 0;
            }
            ((ArrayList)this.aListCsvData[intDay])[14] = cntSencerSum;
            cntSencerFirst = 0;
            cntSencerLast = 0;
            cntSencerLast2 = 0;
            cntOutdoor = 0;
            cntSencerSum = 0;
            cntSencer = 0;

          } // forループ終了
          // データーグリッドビューへ表示
          //データグリッドビューをクリアする
          this.dataGridView1.Columns.Clear();
          /*** イニシャライズ ***/
          // センサー
          DataGridViewTextBoxColumn colDay = new
DataGridViewTextBoxColumn();
          colDay.HeaderText = "センサー";
          colDay.FillWeight = 4;
          colDay.Width = 80;
          colDay.SortMode = DataGridViewColumnSortMode.NotSortable;
//ソート禁止
          this.dataGridView1.Columns.Add(colDay);
          // 日付
          for (int intDay = 0; intDay < this.aListCsvData.Count; intDay +=
this.intSencerCount)
          {
            DataGridViewTextBoxColumn column = new
DataGridViewTextBoxColumn();
            //column.HeaderText =
((ArrayList)this.aListCsvData[intDay])[0].ToString().Substring(5,5);
            column.HeaderText =
((ArrayList)this.aListCsvData[intDay])[0].ToString().Substring(8, 2);
            column.FillWeight = 2;
            column.Width = 30;
            column.SortMode =
DataGridViewColumnSortMode.NotSortable; //ソート禁止
            this.dataGridView1.Columns.Add(column);
          }
          // データ合計
          DataGridViewTextBoxColumn colSum = new
DataGridViewTextBoxColumn();
          colSum.HeaderText = "総合計";
          colSum.FillWeight = 3;
          colSum.Width = 40;
          colSum.SortMode = DataGridViewColumnSortMode.NotSortable;
//ソート禁止
          this.dataGridView1.Columns.Add(colSum);
          // データ1日平均
          DataGridViewTextBoxColumn colAve = new
DataGridViewTextBoxColumn();
          colAve.HeaderText = "平均";
          colAve.FillWeight = 3;
          colAve.Width = 40;
          colAve.SortMode = DataGridViewColumnSortMode.NotSortable;
//ソート禁止
          this.dataGridView1.Columns.Add(colAve);
          //初日の曜日を取得
          DateTime dt =
DateTime.Parse(((ArrayList)this.aListCsvData[0])[0].ToString().Substring(0,
10));
          int syonichi = int.Parse(dt.DayOfWeek.ToString("d"));
          // 曜日毎の平均値
          for (int week = 0; week < 7; week++)
          {
            DataGridViewTextBoxColumn column = new
DataGridViewTextBoxColumn();
            column.HeaderText = ("日月火水木金土").Substring(week, 1);
            column.FillWeight = 2;
            column.Width = 30;
            column.SortMode =
DataGridViewColumnSortMode.NotSortable; //ソート禁止
            this.dataGridView1.Columns.Add(column);
          }
          /*** データ表示 (DataGridView にデータ表示)(合計や平均は
ここで求める場合もある)***/
          int sumAllDay = 0;
          for (int i = 0; i < this.intSencerCount; i++) // センサー番号
          {
            int sumAllData = 0;
            int sumDayData = 0;
            int sumNightData = 0;
            int rowNum = 0;
            // 曜日毎合計と平均のための2次元配列
            this.aAveDweek = new int[this.intSencerCount * 3, 7]; //日中
            this.aAveNweek = new int[this.intSencerCount * 3, 7]; //夜間
            this.aAveAweek = new int[this.intSencerCount * 3, 7]; //一日
            for (int j = 0; j < 3; j++) // 0日中:1夜間:2合計
            {
              rowNum = (i * 3) + j;
```

```
                this.dataGridView1.Rows.Add();
                if (j == 0) //日中
                {
                    //センサー番号

this.dataGridView1.Rows[rowNum].Cells[0].Style.BackColor =
Color.AliceBlue;
                    this.dataGridView1.Rows[rowNum].Cells[0].Value =
((string)((ArrayList)this.aListCsvData[i])[1]) + "(日中)";
                    for (int cnt = 0; cnt < this.arraySencer.GetLength(0); cnt++)
                    {
                        if (this.arraySencer[cnt, 1] ==
(string)((ArrayList)this.aListCsvData[i])[1])
                        {
                            this.dataGridView1.Rows[rowNum].Cells[0].Value =
this.arraySencer[cnt, 0] + "(日中)";
                        }
                    }
                    //センサーデータ
                    int z = 0;
                    for (int intDay = i; intDay < this.aListCsvData.Count; intDay
+= this.intSencerCount)//1日毎
                    {
                        this.dataGridView1.Rows[rowNum].Cells[z +
1].Style.BackColor = Color.AliceBlue;
                        this.dataGridView1.Rows[rowNum].Cells[z + 1].Value =
((ArrayList)this.aListCsvData[intDay])[9];
                        this.aAveDweek[rowNum, z % 7] +=
(int)((ArrayList)this.aListCsvData[intDay])[9]; //曜日毎
                        sumDayData +=
(int)((ArrayList)this.aListCsvData[intDay])[9]; // 合計
                        z++;
                    }
                    this.dataGridView1.Rows[rowNum].Cells[z +
1].Style.BackColor = Color.AliceBlue;
                    this.dataGridView1.Rows[rowNum].Cells[z + 1].Value =
sumDayData;
                    this.dataGridView1.Rows[rowNum].Cells[z +
2].Style.BackColor = Color.AliceBlue;
                    this.dataGridView1.Rows[rowNum].Cells[z + 2].Value =
(double)sumDayData / z;
                    //曜日毎の平均値
                    for (int w = 0; w < 7; w++)
                    {
                        this.dataGridView1.Rows[rowNum].Cells[z + 3 +
w].Style.BackColor = Color.AliceBlue;
                        if (w < z % 7)
                            this.dataGridView1.Rows[rowNum].Cells[z + 3 + ((w +
syonichi) % 7)].Value = ((Double)this.aAveDweek[rowNum, w] / (int)(z / 7 +
1)).ToString();
                        else
                            this.dataGridView1.Rows[rowNum].Cells[z + 3 + ((w +
syonichi) % 7)].Value = ((Double)this.aAveDweek[rowNum, w] / (int)(z /
7)).ToString();
                    }
                }
                if (j == 1) //夜間
                {
                    //センサー番号

this.dataGridView1.Rows[rowNum].Cells[0].Style.BackColor =
Color.MistyRose;
                    this.dataGridView1.Rows[rowNum].Cells[0].Value =
((string)((ArrayList)this.aListCsvData[i])[1]) + "(夜間)";
                    for (int cnt = 0; cnt < this.arraySencer.GetLength(0); cnt++)
                    {
                        if (this.arraySencer[cnt, 1] ==
(string)((ArrayList)this.aListCsvData[i])[1])
                        {
                            this.dataGridView1.Rows[rowNum].Cells[0].Value =
this.arraySencer[cnt, 0] + "(夜間)";
                        }
                    }
                    //センサーデータ
                    int z = 0;
                    for (int intDay = i; intDay < this.aListCsvData.Count; intDay
+= this.intSencerCount)//1日毎
                    {
                        this.dataGridView1.Rows[rowNum].Cells[z +
1].Style.BackColor = Color.MistyRose;
                        this.dataGridView1.Rows[rowNum].Cells[z + 1].Value =
((ArrayList)this.aListCsvData[intDay])[10];
                        this.aAveNweek[rowNum, z % 7] +=
(int)((ArrayList)this.aListCsvData[intDay])[10]; //曜日毎
                        sumNightData +=
(int)((ArrayList)this.aListCsvData[intDay])[10];
                        z++;
                    }
                    this.dataGridView1.Rows[rowNum].Cells[z +
1].Style.BackColor = Color.MistyRose;
                    this.dataGridView1.Rows[rowNum].Cells[z + 1].Value =
sumNightData;
                    this.dataGridView1.Rows[rowNum].Cells[z +
2].Style.BackColor = Color.MistyRose;
                    this.dataGridView1.Rows[rowNum].Cells[z + 2].Value =
(double)sumNightData / z;
                    //曜日毎の平均値
                    for (int w = 0; w < 7; w++)
                    {
                        this.dataGridView1.Rows[rowNum].Cells[z + 3 +
w].Style.BackColor = Color.MistyRose;
                        if (w < z % 7)
                            this.dataGridView1.Rows[rowNum].Cells[z + 3 + ((w +
syonichi) % 7)].Value = ((Double)this.aAveNweek[rowNum, w] / (int)(z / 7 +
1)).ToString();
                        else
                            this.dataGridView1.Rows[rowNum].Cells[z + 3 + ((w +
syonichi) % 7)].Value = ((Double)this.aAveNweek[rowNum, w] / (int)(z /
7)).ToString();
                    }
                }
                if (j == 2) //終日
                {
                    //センサー番号

this.dataGridView1.Rows[rowNum].Cells[0].Style.BackColor =
Color.LightYellow;
                    this.dataGridView1.Rows[rowNum].Cells[0].Style.Font =
new Font(this.dataGridView1.Font, FontStyle.Bold);
                    this.dataGridView1.Rows[rowNum].Cells[0].Value =
((string)((ArrayList)this.aListCsvData[i])[1]) + "(合計)";
                    for (int cnt = 0; cnt < this.arraySencer.GetLength(0); cnt++)
                    {
                        if (this.arraySencer[cnt, 1] ==
(string)((ArrayList)this.aListCsvData[i])[1])
                        {
                            this.dataGridView1.Rows[rowNum].Cells[0].Value =
this.arraySencer[cnt, 0] + "(合計)";
                        }
                    }
                    //センサーデータ
                    int z = 0;
                    for (int intDay = i; intDay < this.aListCsvData.Count; intDay
+= this.intSencerCount)//1日毎
```

```
                {
                    this.dataGridView1.Rows[rowNum].Cells[z + 1].Style.BackColor = Color.LightYellow;
                    this.dataGridView1.Rows[rowNum].Cells[z + 1].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
                    this.aAveAweek[rowNum, z % 7] += (int)((ArrayList)this.aListCsvData[intDay])[8]; //曜日毎
                    this.dataGridView1.Rows[rowNum].Cells[z + 1].Value = ((ArrayList)this.aListCsvData[intDay])[8];
                    sumAllData += (int)((ArrayList)this.aListCsvData[intDay])[8];
                    z++;
                }
                //センサー毎の合計値
                this.dataGridView1.Rows[rowNum].Cells[z + 1].Style.BackColor = Color.LightYellow;
                this.dataGridView1.Rows[rowNum].Cells[z + 1].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
                this.dataGridView1.Rows[rowNum].Cells[z + 1].Value = sumAllData;
                //センサー毎の1日平均値
                this.dataGridView1.Rows[rowNum].Cells[z + 2].Style.BackColor = Color.LightYellow;
                this.dataGridView1.Rows[rowNum].Cells[z + 2].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
                this.dataGridView1.Rows[rowNum].Cells[z + 2].Style.BackColor = Color.LightYellow;
                this.dataGridView1.Rows[rowNum].Cells[z + 2].Value = sumAllData / (Double)z;
                //曜日毎の平均値
                for (int w = 0; w < 7; w++)
                {
                    this.dataGridView1.Rows[rowNum].Cells[z + 3 + w].Style.BackColor = Color.LightYellow;
                    this.dataGridView1.Rows[rowNum].Cells[z + 3 + w].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
                    if (w < z % 7)
                        this.dataGridView1.Rows[rowNum].Cells[z + 3 + ((w + syonichi) % 7)].Value = ((Double)this.aAveAweek[rowNum, w] / (int)(z / 7 + 1)).ToString();
                    else
                        this.dataGridView1.Rows[rowNum].Cells[z + 3 + ((w + syonichi) % 7)].Value = ((Double)this.aAveAweek[rowNum, w] / (int)(z / 7)).ToString();
                }
                sumAllDay += sumAllData;
            }

        }
    }
    int rowNum2 = this.intSencerCount*3;
    // 曜日毎合計と平均のための2次元配列
    this.aAveWakeupWeek = new int[this.intSencerCount * 4, 7]; //起床
    this.aAveSleepWeek = new int[this.intSencerCount * 4, 7]; //就寝
    this.aAveOutdoorCntWeek = new int[this.intSencerCount * 4, 7]; //外出回数
    this.aAveOutdoorTimeWeek = new int[this.intSencerCount * 4, 7]; //外出時間
    int sumAllWakeupData = 0;
    int sumAllSleepData = 0;
    int sumAllOutdoorCntData = 0;
    int sumAllOutdoorTimeData = 0;
    for (int j = 0; j < 4; j++)
    {
        this.dataGridView1.Rows.Add();
        if (j == 0)
        {
            this.dataGridView1.Rows[rowNum2 + j].Cells[0].Style.BackColor = Color.WhiteSmoke;
            this.dataGridView1.Rows[rowNum2 + j].Cells[0].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
            this.dataGridView1.Rows[rowNum2 + j].Cells[0].Value = "起床時間";
            //センサーデータ
            int z = 0;
            int z_er = 0;
            int[] z_ar = { 0, 0, 0, 0, 0, 0, 0 };
            for (int intDay = 0; intDay < this.aListCsvData.Count; intDay += this.intSencerCount)
            {
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.BackColor = Color.WhiteSmoke;
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
                if ((int)((ArrayList)this.aListCsvData[intDay])[11] != 0)
                    this.aAveWakeupWeek[rowNum2 + j, z % 7] += (int)((ArrayList)this.aListCsvData[intDay])[11]; //曜日毎
                else
                    z_ar[z % 7]++;
                string zikan = ((int)((ArrayList)this.aListCsvData[intDay])[11] / 60).ToString("d2") + ":" + ((int)((ArrayList)this.aListCsvData[intDay])[11] % 60).ToString("d2");
                if ((int)((ArrayList)this.aListCsvData[intDay])[11] != 0)
                {
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = zikan;
                    sumAllWakeupData += (int)((ArrayList)this.aListCsvData[intDay])[11];
                }
                else
                {
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = "-";
                    z_er++;
                }

                z++;
            }

            //センサー毎の合計値
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.BackColor = Color.WhiteSmoke;
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
            //this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = sumAllWakeupData;
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = null;
            //センサー毎の1日平均値
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Style.BackColor = Color.WhiteSmoke;
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Style.BackColor = Color.WhiteSmoke;
            string zikan2 = ((int)(sumAllWakeupData / (z - z_er) / 60)).ToString("d2") + ":" + ((int)(sumAllWakeupData / (z - z_er) % 60)).ToString("d2");
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Value = zikan2;
            //曜日毎の平均値
            for (int w = 0; w < 7; w++)
            {
```

```
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + w].Style.BackColor = Color.WhiteSmoke;
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + w].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
                if (w < z % 7)
                {
                    string zikan3 = ((this.aAveWakeupWeek[rowNum2 + j, w] / ((z / 7 + 1) - z_ar[w])) / 60).ToString("d2") + ":" + ((this.aAveWakeupWeek[rowNum2 + j, w] / ((z / 7 + 1) - z_ar[w])) % 60).ToString("d2");
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + ((w + syonichi) % 7)].Value = zikan3;
                }
                else
                {
                    string zikan3 = ((this.aAveWakeupWeek[rowNum2 + j, w] / ((z / 7) - z_ar[w])) / 60).ToString("d2") + ":" + ((this.aAveWakeupWeek[rowNum2 + j, w] / ((z / 7) - z_ar[w])) % 60).ToString("d2");
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + ((w + syonichi) % 7)].Value = zikan3;
                }
            }

        }

        if (j == 1)
        {
            this.dataGridView1.Rows[rowNum2 + j].Cells[0].Style.BackColor = Color.WhiteSmoke;
            this.dataGridView1.Rows[rowNum2 + j].Cells[0].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
            this.dataGridView1.Rows[rowNum2 + j].Cells[0].Value = "就寝時間";
            //センサーデータ
            int z = 0;
            int z_er = 0;
            int[] z_ar = { 0, 0, 0, 0, 0, 0, 0 };
            for (int intDay = 0; intDay < this.aListCsvData.Count; intDay += this.intSencerCount)
            {
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.BackColor = Color.WhiteSmoke;
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
                if ((int)((ArrayList)this.aListCsvData[intDay])[12] != 0)
                    this.aAveSleepWeek[rowNum2 + j, z % 7] += (int)((ArrayList)this.aListCsvData[intDay])[12]; //曜日毎
                else
                    z_ar[z % 7]++;
                string zikan = ((int)((ArrayList)this.aListCsvData[intDay])[12] / 60).ToString("d2") + ":" + ((int)((ArrayList)this.aListCsvData[intDay])[12] % 60).ToString("d2");
                if ((int)((ArrayList)this.aListCsvData[intDay])[12] != 0)
                {
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = zikan;
                    sumAllSleepData += (int)((ArrayList)this.aListCsvData[intDay])[12];
                }
                else
                {
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = "-";
                    z_er++;
                }
                z++;
            }
            //センサー毎の合計値
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.BackColor = Color.WhiteSmoke;
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
            //this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = sumAllSleepData;
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = null;
            //センサー毎の１日平均値
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Style.BackColor = Color.WhiteSmoke;
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Style.BackColor = Color.WhiteSmoke;
            string zikan2 = ((int)(sumAllSleepData / (z - z_er) / 60)).ToString("d2") + ":" + ((int)(sumAllSleepData / (z - z_er) % 60)).ToString("d2");
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Value = zikan2;
            //曜日毎の平均値
            for (int w = 0; w < 7; w++)
            {
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + w].Style.BackColor = Color.WhiteSmoke;
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + w].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
                if (w < z % 7)
                {
                    string zikan3 = ((this.aAveSleepWeek[rowNum2 + j, w] / ((z / 7 + 1) - z_ar[w])) / 60).ToString("d2") + ":" + ((this.aAveSleepWeek[rowNum2 + j, w] / ((z / 7 + 1) - z_ar[w])) % 60).ToString("d2");
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + ((w + syonichi) % 7)].Value =zikan3;
                }
                else
                {
                    string zikan3 = ((this.aAveSleepWeek[rowNum2 + j, w] / ((z / 7) - z_ar[w])) / 60).ToString("d2") + ":" + ((this.aAveSleepWeek[rowNum2 + j, w] / ((z / 7) - z_ar[w])) % 60).ToString("d2");
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + ((w + syonichi) % 7)].Value = zikan3;
                }
            }
        }

        if (j == 2)
        {
            this.dataGridView1.Rows[rowNum2 + j].Cells[0].Style.BackColor = Color.WhiteSmoke;
            this.dataGridView1.Rows[rowNum2 + j].Cells[0].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
            this.dataGridView1.Rows[rowNum2 + j].Cells[0].Value = "外出回数";
            //センサーデータ
            int z = 0;
            for (int intDay = 0; intDay < this.aListCsvData.Count; intDay += this.intSencerCount)
            {
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.BackColor = Color.WhiteSmoke;
                this.dataGridView1.Rows[rowNum2 + j].Cells[z +
```

```
1].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
                this.aAveOutdoorCntWeek[rowNum2 + j, z % 7] += (int)((ArrayList)this.aListCsvData[intDay])[13]; //曜日毎
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = ((ArrayList)this.aListCsvData[intDay])[13];
                sumAllOutdoorCntData += (int)((ArrayList)this.aListCsvData[intDay])[13];
                z++;
            }
            //センサー毎の合計値
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.BackColor = Color.WhiteSmoke;
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = sumAllOutdoorCntData;
            //センサー毎の1日平均値
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Style.BackColor = Color.WhiteSmoke;
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Style.BackColor = Color.WhiteSmoke;
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Value = sumAllOutdoorCntData / (Double)z;
            //曜日毎の平均値
            for (int w = 0; w < 7; w++)
            {
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + w].Style.BackColor = Color.WhiteSmoke;
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + w].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
                if (w < z % 7)
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + ((w + syonichi) % 7)].Value = ((Double)this.aAveOutdoorCntWeek[rowNum2 + j, w] / (int)(z / 7 + 1)).ToString();
                else
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + ((w + syonichi) % 7)].Value = ((Double)this.aAveOutdoorCntWeek[rowNum2 + j, w] / (int)(z / 7)).ToString();
            }
        }
        if (j == 3)
        {
            this.dataGridView1.Rows[rowNum2 + j].Cells[0].Style.BackColor = Color.WhiteSmoke;
            this.dataGridView1.Rows[rowNum2 + j].Cells[0].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
            this.dataGridView1.Rows[rowNum2 + j].Cells[0].Value = "外出時間合計";
            //センサーデータ
            int z = 0;
            int z_er = 0;
            int[] z_ar = { 0, 0, 0, 0, 0, 0, 0 };
            for (int intDay = 0; intDay < this.aListCsvData.Count; intDay += this.intSencerCount)
            {
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.BackColor = Color.WhiteSmoke;
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
                if ((int)((ArrayList)this.aListCsvData[intDay])[14] != 0)
                    this.aAveOutdoorTimeWeek[rowNum2 + j, z % 7] += (int)((ArrayList)this.aListCsvData[intDay])[14]; //曜日毎
                else
                    z_ar[z % 7]++;
                string zikan = ((int)((ArrayList)this.aListCsvData[intDay])[14] / 60).ToString("d2") + ":" + ((int)((ArrayList)this.aListCsvData[intDay])[14] % 60).ToString("d2");
                if ((int)((ArrayList)this.aListCsvData[intDay])[14] != 0)
                {
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = zikan;
                    sumAllOutdoorTimeData += (int)((ArrayList)this.aListCsvData[intDay])[14];
                }
                else
                {
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = "-";
                    z_er++;
                }

                z++;
            }
            //センサー毎の合計値
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.BackColor = Color.WhiteSmoke;
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
            //this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = sumAllOutdoorTimeData;
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 1].Value = null;
            //センサー毎の1日平均値
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Style.BackColor = Color.WhiteSmoke;
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Style.BackColor = Color.WhiteSmoke;
            string zikan2 = ((int)(sumAllOutdoorTimeData / (z-z_er) / 60)).ToString("d2") + ":" + ((int)(sumAllOutdoorTimeData / (z-z_er) % 60)).ToString("d2");
            this.dataGridView1.Rows[rowNum2 + j].Cells[z + 2].Value = zikan2;
            //曜日毎の平均値
            for (int w = 0; w < 7; w++)
            {
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + w].Style.BackColor = Color.WhiteSmoke;
                this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + w].Style.Font = new Font(this.dataGridView1.Font, FontStyle.Bold);
                if (w < z % 7)
                {
                    string zikan3 = ((this.aAveOutdoorTimeWeek[rowNum2 + j, w] / ((z / 7 + 1) - z_ar[w])) / 60).ToString("d2") + ":" + ((this.aAveOutdoorTimeWeek[rowNum2 + j, w] / ((z / 7 + 1) - z_ar[w])) % 60).ToString("d2");
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + ((w + syonichi) % 7)].Value = zikan3;
                }
                else
                {
                    string zikan3 = ((this.aAveOutdoorTimeWeek[rowNum2 + j, w] / ((z / 7) - z_ar[w])) / 60).ToString("d2") + ":" + ((this.aAveOutdoorTimeWeek[rowNum2 + j, w] / ((z / 7) - z_ar[w])) % 60).ToString("d2");
                    this.dataGridView1.Rows[rowNum2 + j].Cells[z + 3 + ((w + syonichi) % 7)].Value = zikan3;
                }
            }
        }
```

```
            }

        }
        catch (Exception ex)
        {
            //メッセージボックスを表示する
            MessageBox.Show("解析できません。設定ファイル等を確認し
てください。¥r¥n"+ex.ToString(),
                "エラー",
                MessageBoxButtons.OK,
                MessageBoxIcon.Error);
            Console.Write(ex.ToString());
        }

    }
    /// <summary>
    /// 閾値をセットする
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void comboBox7_SelectedIndexChanged(object sender,
EventArgs e)
    {
        this.intSencerThreshold =
int.Parse(this.comboBox7.SelectedItem.ToString());
    }
    // GridView のデータをエクセルにエクスポート
    public void SaveExcel()
    {
        // EXCEL起動
        Excel.Application objExcel = null;
        Excel.Workbook objWorkBook = null;
        Excel.Worksheet objWorkSheet = null;
        objExcel = new Excel.Application();
        objWorkBook = objExcel.Workbooks.Add(
            Excel.XlWBATemplate.xlWBATWorksheet);
        // DataGridViewのセルのデータ取得
        String[,] v = new String[
            this.dataGridView1.Rows.Count+1,
this.dataGridView1.Columns.Count];
        for (int r = 0; r <= this.dataGridView1.Rows.Count - 1; r++)
        {
            for (int c = 0; c <= this.dataGridView1.Columns.Count - 1; c++)
            {
                String sdt = "";
                // DataGridViewのヘッダーデータ取得
                if (r == 0)
                {
                    sdt =
this.dataGridView1.Columns[c].HeaderCell.Value.ToString();
                    v[r, c] = sdt;
                }

                if (this.dataGridView1.Rows[r].Cells[c].Value != null)
                {
                    sdt = this.dataGridView1.Rows[r].Cells[c].Value.ToString();
                }
                v[r+1, c] = sdt;
            }
        }
        // EXCELにデータ転送
        objWorkSheet = (Excel.Worksheet)objWorkBook.Sheets[1];
        objWorkSheet.Range[
            objWorkSheet.Cells[1, 1], objWorkSheet.Cells[
            this.dataGridView1.Rows.Count+1,
this.dataGridView1.Columns.Count]].Value2 = v;
        // エクセル表示
        objExcel.Visible = true;
        // EXCEL解放
        Marshal.ReleaseComObject(objWorkBook);
        Marshal.ReleaseComObject(objExcel);
        Marshal.ReleaseComObject(objWorkSheet);
        objWorkSheet = null;
        objWorkBook = null;
        objExcel = null;
    }
    /// <summary>
    /// CSVデータ書き出し処理
    /// </summary>
    /// <param name="fp"></param>
    public void SaveCsv(String fp)
    {
        // CSVファイルオープン
        StreamWriter sw =
            new StreamWriter(fp, false,
            System.Text.Encoding.GetEncoding("SHIFT-JIS"));
        for (int r = 0; r <= this.dataGridView1.Rows.Count - 1; r++)
        {
            // DataGridViewのヘッダーデータ取得
            if (r == 0)
            {
                for (int c = 0; c <= this.dataGridView1.Columns.Count - 1; c++)
                {
                    String sdt = "";
                    sdt =
this.dataGridView1.Columns[c].HeaderCell.Value.ToString();
                    if (c < this.dataGridView1.Columns.Count - 1)
                    {
                        sdt = sdt + ",";
                    }
                    // CSVファイル書込
                    sw.Write(sdt);
                }
                sw.Write("¥n");
            }
            for (int c = 0; c <= this.dataGridView1.Columns.Count - 1; c++)
            {
                // DataGridViewのセルのデータ取得
                String dt = "";
                if (this.dataGridView1.Rows[r].Cells[c].Value != null)
                {
                    dt = this.dataGridView1.Rows[r].Cells[c].Value.ToString();
                }
                if (c < this.dataGridView1.Columns.Count - 1)
                {
                    dt = dt + ",";
                }
                // CSVファイル書込
                sw.Write(dt);
            }
            sw.Write("¥n");
        }
        // CSVファイルクローズ
        sw.Close();
    }
    /// <summary>
    /// エクセル出力ボタンをクリックする
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void button4_Click(object sender, EventArgs e)
    {
        if (this.dataGridView1.Rows.Count > 0)
        {
```

```
            // エクセルに表示
            this.SaveExcel();
        }
        else
        {
            //メッセージボックスを表示する
            MessageBox.Show("解析データがありません",
                "エラー",
                MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
    }
    /// <summary>
    /// CSV出力ボタンをクリックする
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void button5_Click(object sender, EventArgs e)
    {
        if (this.dataGridView1.Rows.Count > 0)
        {
            //SaveFileDialogクラスのインスタンスを作成
            SaveFileDialog sfd = new SaveFileDialog();
            //デフォルトのファイル名を指定
            sfd.FileName = this.readCsvFileName;
            //[ファイルの種類]に表示される選択肢を指定する
            sfd.Filter = "csvファイル(*.csv;*.txt)|*.csv;*.txt|すべてのファイル
(*.*)|*.*";
            //ダイアログを表示する
            if (sfd.ShowDialog() == DialogResult.OK)
            {
                // CSVに書き出し
                this.SaveCsv(sfd.FileName);
            }
        }
        else
        {
            //メッセージボックスを表示する
            MessageBox.Show("解析データがありません",
                "エラー",
                MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
    }
    /// <summary>
    /// データグリッドビューのセルをクリックすると、
    /// 該当行のデータを棒グラフで表示する
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void dataGridView1_CellContentDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        // グリッドを基準としたカーソル位置のポイントを取得
        Point p = dataGridView1.PointToClient(Cursor.Position);

        // 取得したポイントからHitTestでセル位置取得
        DataGridView.HitTestInfo ht = dataGridView1.HitTest(p.X, p.Y);
        //グラフ表示のためにForm2クラスのインスタンスを作成する
        Form2 f = new Form2();
        // グラフデータを受け渡し
        //選択した行のタイトル
        f.legend =
this.dataGridView1.Rows[ht.RowIndex].Cells[0].Value.ToString();
        //データ
        int datacount = 31; //31日で固定
        f.xValues = new string[datacount];
        f.yValues = new string[datacount];
        Boolean flg_break = false;
        for (int i = 0; i < datacount; i++)
        {
            if (this.dataGridView1.Columns[i + 1].HeaderCell.Value.ToString()
== "総合計")
            {
                flg_break = true;
            }
            if(flg_break)
            {
                f.xValues[i] = "";
                f.yValues[i] = "0";
            }
            else
            {
                f.xValues[i] = this.dataGridView1.Columns[i +
1].HeaderCell.Value.ToString();
                int intReturn;
                if (int.TryParse(this.dataGridView1.Rows[ht.RowIndex].Cells[i +
1].Value.ToString(), out intReturn))
                {
                    f.yValues[i] = this.dataGridView1.Rows[ht.RowIndex].Cells[i +
1].Value.ToString();
                }
                else
                {
                    if ((f.legend == "就寝時間" || f.legend == "起床時間" ||
f.legend == "外出時間合計") &&
this.dataGridView1.Rows[ht.RowIndex].Cells[i + 1].Value.ToString() != "-")
                    {
                        int x =
int.Parse(this.dataGridView1.Rows[ht.RowIndex].Cells[i +
1].Value.ToString().Substring(0, 2)) * 60 +
int.Parse(this.dataGridView1.Rows[ht.RowIndex].Cells[i +
1].Value.ToString().Substring(3, 2));
                        f.yValues[i] = (x/60.0).ToString();
                    }
                    else
                    {
                        f.yValues[i] = "0";
                    }
                }
            }
            //Form2を表示する
            //ここではモーダルダイアログボックスとして表示する
            //オーナーウィンドウにthisを指定する
            f.ShowDialog(this);
            //フォームが必要なくなったところで、Disposeを呼び出す
            f.Dispose();
        }
        private void comboBox14_SelectedIndexChanged(object sender,
EventArgs e)
        {
            this.flgOutdoorTime =
int.Parse(this.comboBox14.SelectedItem.ToString());
        }
        private void comboBox15_SelectedIndexChanged(object sender,
EventArgs e)
        {
            this.cntFilter = int.Parse(this.comboBox15.SelectedItem.ToString());
        }
    }
}
```

厚生労働科学研究費補助金（認知症対策総合研究事業）分担研究報告書

第2部　自立支援機器による認知機能低下高齢者の状態把握の試み

第 2 章　転倒・傷害の予防的効果からみた自立支援機器の検討

細井孝之

国立長寿医療研究センター

【要旨】

高齢者を対象とした遠隔パッシブモニタリングシステムであり、個人の活動や行動パターンを監視し、転倒や不具合の発生など健康問題や緊急事態となりうる状況をケア担当者に通知する「高齢者見守りセンサーシステム(Quiet Care)」の試験運用を通じて得られた知見をもとに、地域在住高齢者向けの自立支援機器に求められる機能について検討した。地域在住高齢者向けの自立支援機器においても寝室からトイレへの移動回数とその時間帯について把握する機能が備わっていることが有用であると考えられた。

A．目的

高齢者人口が急増する中、独居の認知機能低下者(認知症もしくはMild cognitive impairment,MCI等)も増加している。これらの独居者は転倒・傷害を始めとするさまざまな自立阻害要因にさらされるリスクを有していることは注目すべきことであり、それらをより早期に発見し対処することが、健康障害や生活機能低下を予防するために重要である。本研究班全体の目標はそのための機能を有する支援機器を開発・導入し、その機器を地域包括支援センターや介護事業者等(以後、地域ケア機関)が効果的・効率的に活用できるシステムを呈示することである。昨年度は独居高齢者住宅における自立支援機器になる可能性があるGEヘルスケア社の「見守りセンサーシステム(Quiet Care)」の特徴を転倒・傷害の予防効果の観点から検討したが、本年度は本システムの試験運用を通じて得られた知見をもとに、地域在住高齢者向けの自立支援機器に求められる機能について検討した。

B．方法

GEヘルスケア社の「高齢者見守りセンサーシステム(Quiet Care)」を用いて行われている臨床研究(「介護付き有料老人ホームならびに高齢者専用賃貸住宅に入居している独居高齢者を対象としたQuietCare(見守りセンサーシステム)のランダム化比較試験」主任研究者：細井孝之、研究責任医師：藤原佳典)を通じて得られた知見をもとに、地域在住高齢者向けの自立支援機器に求められる機能について検討した。

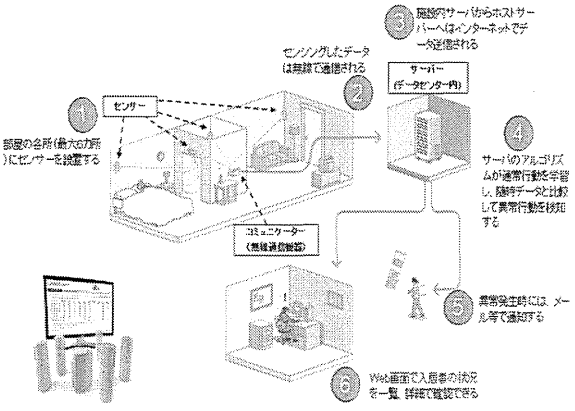
（倫理面への配慮）

個人情報は全く扱わない研究であり、倫理的な問題はない。

C．結果

QuietCare(高齢者見守りセンサー)は、高齢者を対象とした遠隔パッシブモニタリングシステムで、個人の活動や行動パターンを監視し、転倒や不具合の発生など健康問題や緊急事態となりうる状況をケア担当者に通知するシステムである。米国で介護付き生活施設において、QuietCareの有用性の試験を行った結果、 QuietCareを設置した施設ではControlの施設に比べ転倒数が数的に少なかったことが示された。しかしながら、住宅環境ならびに介護体制の異なる日本においても、QuietCareが有用か否かについては臨床データを得ることを目的とした研究によって明らかにされなければならない。このため、現在「介護付き有料老人ホームならびに高齢者専用賃貸住宅に入居している独居高齢者を対象としたQuietCare(見守りセンサーシステム)のランダム化比較試験」（主任研究者：細井孝之、研究責任医師：藤原佳典、国立長寿医療研究センターIRB承認11-1)が行われている。この試験は9つの高齢者専用賃貸住宅に入居している高齢者200名をQuietCare 設置群100名、非設置群100名の2群にランダムに分け、急変等のイベント数、転倒数、イベント・転倒発生時のかけつけ時間を評価項目としてデザインされた。現在中間解析が為され、安全性と倫理の面から試験継続が認められたところである。

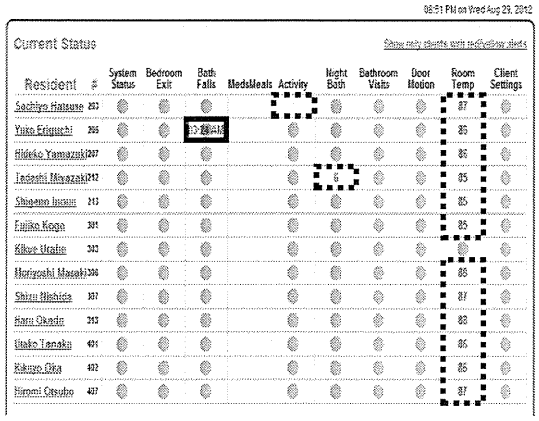
QuietCare設置イメージ



QuietCareは居室あたり最大7つ設置される赤外線センサー、コミュニケーター、ウェブシステムからなる見守りセンサーシステムである。室内に設置された赤外線センサーが、居住者の行動をセンシングし、行動パターンなどの情報がコミュニケーターおよびインターネット回線を通じてサーバーに転送される。転送されたデータはデータサーバーで自動的に解析され、その結果はインターネットを通じて施設の介護士に提供される。

下表は実際のコンピュータ画面を示すモデル図である。表記は英語である。表の縦方向に入居者の氏名が、横方向に監視項目が並んでいる。黄色の○（破線で囲んだ○）が注意、赤の○（実線で囲んだ○）が警告のアラームを表す。

システムの運用例



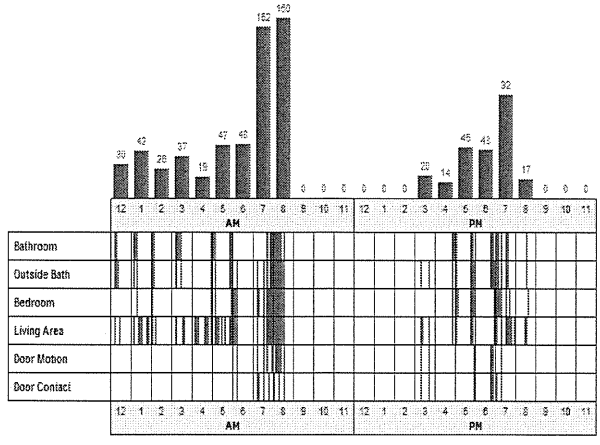
このシステムを使うにあたっては、あらかじめ対象者の行動パターンを反映させたアラーム発生の閾値を設定する必要がある。このプロセスが対象者の行動パターンを介護者があらためて把握する貴重な機会になると思われる。一方、このアラームは緊急時に即応するためのものではなく、行動パターンの変容をレポートするものであることには留意すべきであり、利用者は双方ともよく理解する必要がある。

転倒等のイベントが発生した場合に、その場にうずくまって動けないなどの行動パターン変容があればこのシステムで認識できる。しかしながら、このシステムは転倒等にむすび付き得る行動パターンの変容をあらかじめ捉えて、イベントを予防することに用いられて価値を発揮するものだと考えられる。システムの特徴を生かした利用方法は介護スタッフによる個々の検討の上で決定される必要がある。

また、居室内での使用を前提に開発されたものであるため、居室外に出たことは記録されるものの、外出中の行動については把握できない。

それぞれのセンサーによって感知された活動は定められた間隔毎にカウントされ、下図のように集計され、一日毎全体像をパターン化して把握することが可能である。

行動パターンのグラフ化（例）



現在進行している前記臨床試験において、寝室からトイレへの移動回数の多さから潜在する下部尿路障害の発見につながった例、活動度の全体的な低下がその後の脳血管イベントに先行していた例、自室内での活動度を把握することによってケアプランの見直しにつながった例、などが認められてきた。特にトイレへの移動回数を把握することは排尿・排便のパターンのみならず活動度全体の傾向を把握することに有用であると思われた。

現在わが国ではいくつかの見守りセンサーが使用可能である。下表はそれぞれの特徴をまとめたものである。施設向けに特化されたシステムは今回検討しているQuietCareのみかと思われる。本装置の大きな特徴は行動パターンの変容を指標にすることであ

る。行動パターンは定められたmodalityについて把握されるがそれぞれに関するアラートレベルは個人毎にカスタマイズされること、それぞれのmodalityについて日内ならびに日間の変動をグラフ化したデータをもとにビジュアル化できる点が特徴である。また、アラートレベルを決定する際には、ベースラインにおける入居者の生活パターンを客観的に見直す必要があり、この作業自体がケアプランの見直しにもつながる重要なステップかと思われる。

本センサーは自室内の活動度について把握するものであり、室外の行動についてはまったく関知しない。もともと自室での生活を送ることが可能である方向けの機器であることは念頭において運用しなければならない。

見守りセンサーの比較

	GE QuietCare	海外A社	国内B社
対象	施設向け	家庭向け	
センシング方式	Passive型	Active型	Passive・Active型
装置	施設サーバ 赤外線センサー（最大8個）	通信機 ペンダント	通信機 赤外線センサー（2～3個） ペンダント
見守り箇所	行動パターンの変化 Motionの有無	緊急通報装置	Motionの有無 緊急通報装置
緊急時対応者	職員・介護士	家族・協力員	家族・協力員
価格帯(調査時)	未定	加入料2000円 月額3980円（レンタル）	初期：数万円 月額：1万円以下

D. 考察

独居施設入所者用の「高齢者見守りセンサーシステム(Quiet Care)」を用いて実施されている臨床研究を通して、地域在住高齢者向けの自立支援機器においても寝室からトイレへの移動回数とその時間帯について把握する機能が備わっていることが有用であると考えられた。

G. 研究発表

1.論文発表
＜英文原著＞

1. Furuya T, Hosoi T,Tanaka E,Nakajima A,Taniguti A,Momohara S,Yamanaka H : Prevalence of and Factors associated with vitamin D deficiency in 4,793 Japanese patients with rheumatoid arthritis. Clinical Rheumatology , 2013 ; in Press

2. Haraikawa M, Tsugawa N, Sogabe N , Tanabe R, Kawamura Y, Okano T, Hosoi T, Goseki – Sone M : Effect of gamma-glutamyl carboxylase gene polymorphism(R325Q)on the association between dietary vitamin K intake and gamma-carboxylation of osteocalcin in young men and women. Asia Pacific Journal of Clinical Nutrition , 2012 ; in Press

3. Orimo H, Nakamura T, Hosoi T, Iki M, Uenishi K, Endo N, Ohta H, Shiraki M, Sugimoto T, Suzuki T, Soen S, Nishizawa Y, Hagino H, Fukunaga M, Fujiwara S : Japanese 2011 Guidelines for prevention and treatment of osteoporosis—executive summary. Arch Osteoporos, 2012,in press (Hosoi T as corresponding author).

4. Furuya T,Inoue E,Hosoi T,Taniguchi A,Momohara S,Yamanaka H: Risk factors associated with the occurrence of hip fracture in Japanese patients with rheumatoid arthritis:a prospective observational cohort study. Osteoporosis Int. , 2012.7.17.(Epub)

5. Koudu Y, Onouchi T, Hosoi T, Horiuchi T :Association of CYP19 Gene Polymorphism with Vertebral Fractures in Japanese Postmenopausal. Biochemical Genetics 50:389-396, 2012.

＜和文原著＞

1. 細井孝之、黒田龍彦、中村利孝、白木正孝、太田博明、原田敦、森聖二郎、大橋靖雄、折茂肇:全国データベースを用いた骨粗鬆症性骨折の予防と治療に関する研究. Osteoporosis Japan ,2012.10.31, vol.20,No.4 , 41-48 , ライフサイエンス出版株式会社

＜和文総説＞

1.細井孝之:骨粗鬆症講座 Q&A ガイドラインの改訂. O.li.v.e.－骨代謝と生活習慣病の連関－Vol.2(No.2), 20-25,2012

2. 細井孝之:特集 骨粗鬆症の薬物療法の新戦略 アレンドロネート. 関節外科 基礎と臨床 Vol.31(No.6), 48-52,2012

3. 細井孝之:19 骨粗鬆症 . Medical Practice Vol.29, 252-256,2012

4. 細井孝之:骨粗鬆症.青淵 No.755,14-16,2012

5. 細井孝之:骨粗鬆症の予防と治療ガイドライン 2011 年版.Ortho community 2012 No.43, 11-12,2012

6. 細井孝之:臨床 骨折リスク評価ツール「ＦＲＡＸ®」の日本人への応用.ＯＲＴＨＯ－ＶＩＥＷＳ No.15, 6-7,2012

7. 細井孝之:特集　骨粗鬆症診療に関する新しい展開　骨粗鬆症治療における薬物治療介入のポイント（2011年改訂版ガイドラインの考え方）.内分泌・糖尿病・代謝内科 Vol.34(No.5), 410-414,2012

8. 細井孝之:5.特集　骨粗鬆症治療薬のBreakthrough-ガイドライン2011年版を踏まえて一骨折リスク評価の実際とＦＲＡＸ®.MEDICINAL Vol.2(No.8), 43-48,2012

9. 細井孝之:特集:知っておきたい最新骨粗鬆症診療マニュアル骨折危険因子から.Orthopaedics Vol.25(No.5), 25-30,2012

10. 細井孝之:マンスリーレクチャー老年内科　標榜をめざして.週刊　日本医事新報 No.4605, 41-45,2012

11. 細井孝之:骨粗鬆症の治療-新たな薬物治療開始基準. CLINICIAL Vol.598(No.610), 38-43,2012

12. 細井孝之:特集：骨密度測定の再考ＭＤ法（最新の＋ＤＩＰシステムについて）.骨粗鬆症治療 Vol.11(No.1), 14-18,2012

13. 細井孝之:特集：「骨粗鬆症の予防と治療ガイドライン2011年版」を踏まえた今後の骨粗鬆症治療.骨粗鬆症治療 Vol.10(No.4), 10-14,2012

14. 細井孝之:特集：変わる骨粗鬆症治療—内科医が知っておきたい最新トピックから基本的知識まで—骨粗鬆症の予防と治療ガイドライン2011年版. Mebio Vol.29(No.5),41-46,2012

15. 細井孝之:特集「骨粗鬆症の予防と治療ガイドライン2011」をめぐってＦＲＡＸ®のわが国での活用.CLINICIAL CALCIUM　Vol.22(No.6), 73-79,2012

16. 細井孝之:シリーズ　よく使う日常治療薬の正しい使い方骨粗鬆症に対する薬の使い方.レジデントノート Vol.14(No.10), 1927-1930,2012

17. 細井孝之:骨折リスクに基づいた骨粗鬆症の診断と対策の実際　既存骨折と骨折リスク.Medical Practice Vol29(No.11), 1886-1890,2012

18. 細井孝之:ロコモティブシンドロームの予防と骨粗鬆症.日本未病システム学会 Vol.18(No.3), 74-78,2012

19. 宗圓聰、福永仁夫、杉本利嗣、曽根照喜、細井孝之:診断基準の改定に向けて—骨粗鬆症診療の新たな展開をめざして.Osteoporosis Japan vol.20(No.4), 629-32,2012

＜和文著書＞

1.細井孝之:Ⅱ.運動器の評価　２．ロコモの疑いの人の診察法　３）主な疾患の診断と保存治療⑧骨粗鬆症.ロコモティブシンドローム, 167-175,メディカルレビュー社,2012

2. 細井孝之:Ⅱ.病態・疾患別のガイドライン　代謝性骨疾患１（骨粗鬆症）骨粗鬆症の評価・治療指針.運動器診療　最新ガイドライン,182-187,総合医学社,2012

3. 細井孝之:第１章　高齢者に多い疾患に対する薬の使い方　１）骨粗鬆症（Q13～Q16),高齢者の薬よろずお助けQ＆A 100,39-50,羊土社,2012

4. 細井孝之：第４章　ロコモティブシンドロームと遺伝子多型性.ロコモティブシンドロームと栄養,55-64,建帛社,2012

5. 細井孝之:VI 骨粗鬆症の治療　2．骨粗鬆症の治療薬　6）その他（カルシトニン、ビタミンK，イプリフラボン）.骨粗鬆症診療ハンドブック改訂５版,320-330,医薬ジャーナル社,2012

6. 細井孝之:4.骨粗鬆症　4.1　骨粗鬆症の概念と分類.高齢者用食品の開発と展望,23-28,株式会社シーエムーシー出版.2012

7. 細井孝之:A.骨粗鬆症の評価と指針　1.「骨粗鬆症の予防と治療ガイドライン2011年版」の概要とおもな改訂点.新しい骨粗鬆症治療, 2-4,診断と治療社,2012

8. 細井孝之:第 13 章　骨・運動器疾患. 標準理学療法学・作業療法学　専門基礎分野　老年学　第３版,146-158,医学書院,2012

2.学会発表　：　なし

H. 知的所有権の取得状況

なし

厚生労働科学研究費補助金（認知症対策総合研究事業）分担研究報告書

第2部　自立支援機器による認知機能低下高齢者の状態把握の試み

第3章　サービス付き高齢者住宅における本センサーを用いた入居者の健康状態の把握に関する研究 ― センサーデータと介護記録の照合による検討 ―

長谷部雅美

東京都健康長寿医療センター研究所　社会参加と地域保健研究チーム

【要旨】

サービス付高齢者住宅入居者２名を対象に，赤外線センサーで検知したデータと介護記録の照合から，センサーデータによる健康状態の把握の可能性について検討した。その結果，身体的な健康状態（特に，トイレ回数が増加する消化器系の疾患）は，センサーによる把握が可能であることが示唆された。また，精神的な症状・行動は，日中のセンサーデータには反映されにくいものの，夜間では把握できる可能性が示唆された。しかしながら，センサー検知回数の増減には，複数の要因が関連している可能性があるため，一時的な検知回数の増減については，慎重に対応することが必要である。

A．目的

独居の認知症高齢者への支援において，しばしば指摘される課題に「日常生活の様子が十分に把握できない」「生活リズムや体調変化を早期に発見しづらい」こと等があげられる[1)2)]。これらの課題は，高齢者が地域ケア機関をはじめ，他者からの介入に消極的だったり，認知症に伴う認知機能低下により，高齢者の訴えが必ずしも事実とは限らなかったりする等，的確な情報を随時収集するのが困難であることが要因の一つとして考えられる。

この点について，本研究事業では昨年度の成果として「健康障害の発生が反映するモニタリングデータの特徴」[2)]を明らかにし，センサーデータによる健康障害の発生や予兆の把握を試みた。結果として，センサーで夜間覚醒の兆候や転倒後の生活リズムの変化等を検知することができた。しかしながら，昨年度の課題として，対象者の居住形態が独居であるため，地域ケア機関または別居の家族が，健康障害の発生時期を正確に把握することには限界があった。

そこで本年度は，ケアスタッフが常駐し，日々の介護記録が残されるサービス付高齢者住宅の入居者を対象に，センサーで検知したデータと介護記録の照合を行い，センサーによる健康状態の把握の可能性をより詳細に検討することを目的とした。

B．方法

対象者は，サービス付き高齢者住宅（センサー設置居室）に入居している高齢者２名とした。センサーは，各居室の中央辺り