●人の「密」を検知し遠隔地に知らせるセンサーを開発しました。

※以下は、ハードウエア、ソフトウエアともに「できたての生乾き」状態の「試作品」です。
※ここからの UI/UX の拡張、流布、製品化を有志によってお願いいたします。

【概要】
本センサーは、小さな筐体の中に、人感センサー(人の「動き」を検知する)、超小型コンピュータ、無線 LAN インターフェイスを持ち、指定されたインターバルで、インターバル間の人の動きの「数」をカウントすることによって、その場所(最大で半径 12m 以内の半円)の人の「密」を測ることができます。計測された数値は、インターネット上でやり取りされるため、世界中のどこでも、インターネット接続さえあれば、それを知らせ、集計することができます(今後のプログラムの拡張次第です)。また、人が密になって危険な数値になった場合は、その場所の「密アラーム」を出すこともできます(これも、今後のプログラムの拡張次第です)。また、AWS などのクラウド上にサーバーを別途用意し、そこで動くソフトウエアを開発し、用意すると、数台から数万台の世界各地に散らばった同センサーの統計処理も可能です。統計データが集まれば、人が密になる場所の予測などを AI で行うことも可能になります(別途 AI ソフトウエアの開発が必要)。

【特徴】
1．人の「密」を検知します。値は「人の動き」を示すカウント値になります。
2．検知した「密」に対し、カウント地が一定以上であることを検知し、アラームを出せます。
3．統計情報が出せます。
4．リアルタイムでメールなどに知らせることができます。
5．インターネットを通じて、世界のどこでも、人の「密」のデータを知ることができます。
6．必要であれば、センサー側の蓄電池と太陽電池での駆動も可能です(試作品ではこの機能なし)。

【製造コスト概算】
現在の試作品でのコスト概要は一台あたり以下です。
実際の製品には、美しい画面を作るなど、ソフトウエア開発のコストが別途かかります。

1．人感センサー(約 500 円)
　Panasonic 製の「PaPIRs 12m EKMC1603111」(半径 12m 用)または、
　Panasonic 製の「PaPIRs 5m EKMC160111」(半径 5m 用)を使用しています。
　※今回はこのセンサーを使いましたが、同様の他社製品もあるはずです。

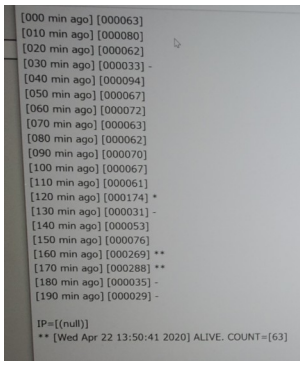
2．超小型ボードコンピュータ(約\3,000-)
　Raspberry Pi 3 Model A+を使いました。
　※Raspberry Pi の他のモデルも利用可能ですが、CPU のスピードにより、検知のカウント値が変わります。

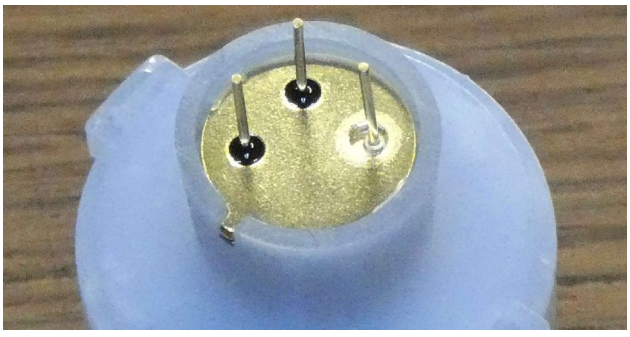
3．電源供給用 USB ケーブル(約\100-)
4．筐体(約\800-)
5．他は、政策人件費。ソフトウエアが完備していれば、1 台あたり約 30 分。

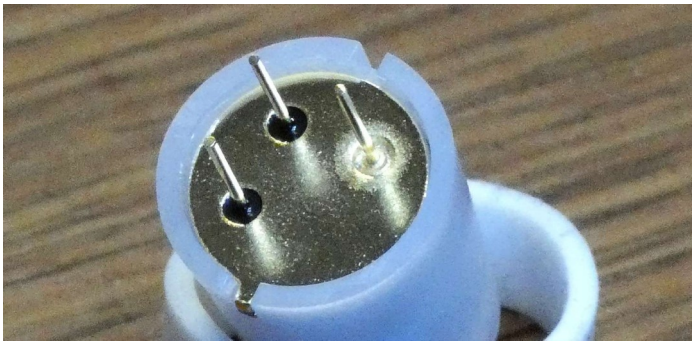
以上、人件費を除いた部品代合計は、￥4,400-(量産時は、さらに個々の部品の数量値引きがあるかと思います)

以下、試作品の写真です。ご覧のように、コンピュータに更に小さいものを使うことによって、更に小さなものにできます。消費電力も更に小さくできます。現在の消費電力は 5V/0.1A です。

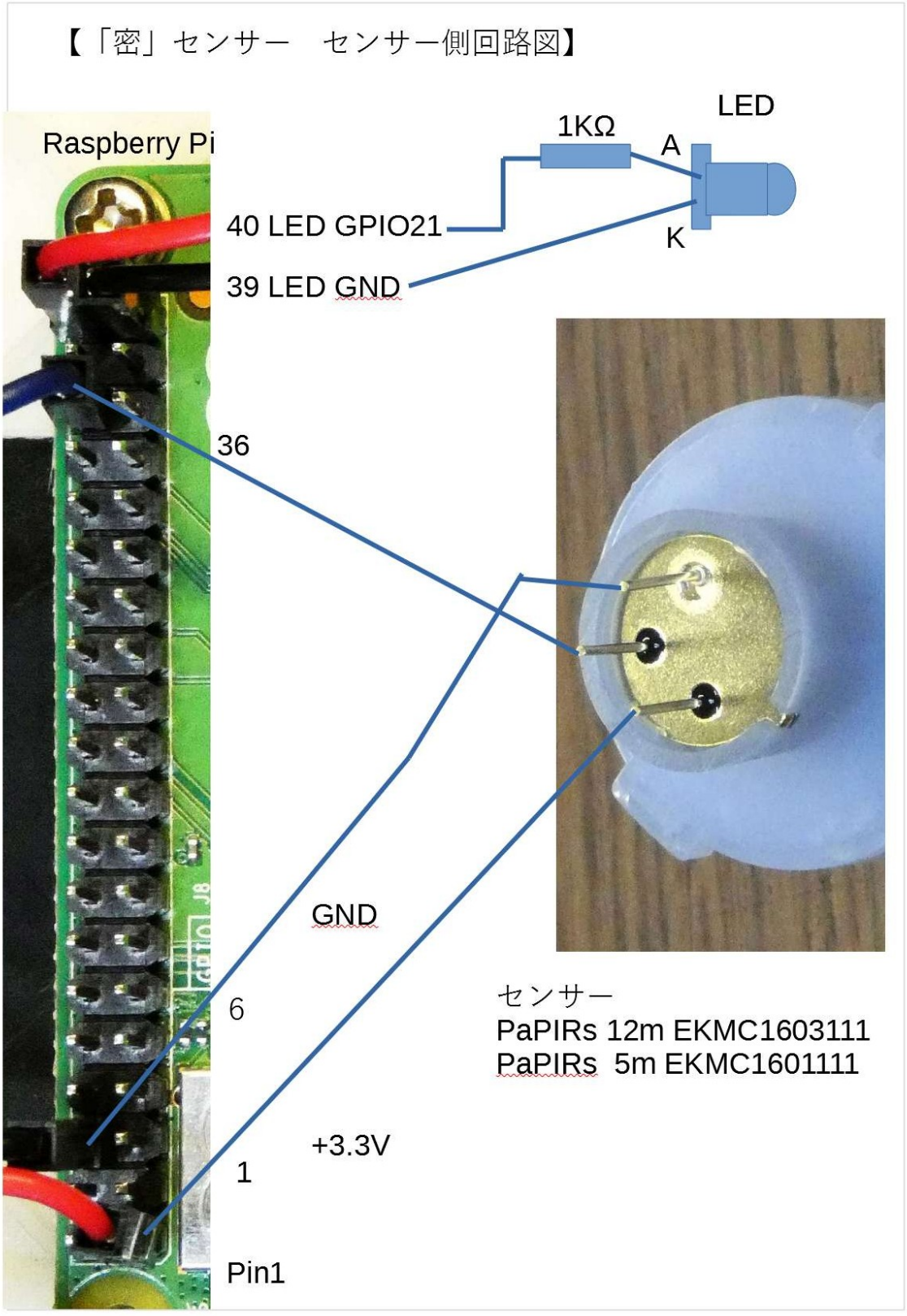
※一番右の写真は、試作の簡易型ではありますが、メールで 10 分ごとに送られてくる画面で、メール到着時点から、190 分前までの人の動きが見られます。現在のところ、1 年間の稼働で問題なく動いています。このインターバルの時間はプログラムで設定できます(試作プログラムの場合)。







【センサー側回路図】


【「密」センサー　センサー側回路図】
Raspberry Pi
1KΩ
LED
A
K
40 LED GPIO21
39 LED GND
36
GND
6
+3.3V
1
Pin1
センサー
PaPIRs 12m EKMC1603111
PaPIRs 5m EKMC1601111

【プログラム利用・改造・発表・商用利用は自由です】

※本ハードウエアの回路図、及びプログラムの改変、改造を歓迎します。
※本プログラム等の資料は、全て「参考」として、使ってください。完全なものではありません。
※本プログラム等は、完全に Public Domain に置きます。改変、改造には、三田の許可なく行っても問題はありません。
※実際のアラームの設置、アラームの値は、サーバー側のプログラムの改造で行いますが、実機でのテストが必要です。実機によって、カウント数値が変わります。

ちなみに、本試作プログラムとハードウエアでは

- 45 カウント以下は、人がいなくてもカウントされる可能性がある。
- カウント数によって、メールの文面、Web の HTML で検知カウントの評価を変えている表示をしています。
- 本試作プログラムでは、LED も点灯させており、常時点灯させ、カウントが行われると、LED が一瞬消える、というようになっています。

この試作プログラムも、この方針で書かれています。

※プログラムはそれぞれ 200 行以内の小さなものです。十分理解したうえ、改造を加えてください。

【センサー側プログラム(サンプル)】

●Raspberry Pi 内でコンパイルするときの簡単なコンパイル用シェルスクリプト
xxxx.c と tcpcli.c がプログラムです。wiringPi ライブラリを使います。

```
cc xxxx.c tcpcli.c -o xxxx -lwiringPi
```

●メインプログラム(xxxx.c)

```
#include <stdio.h>
#include <wiringPi.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>

#define PORT (27)
#define LED (29)
#define IPADDR  "192.168.1.4"  // IP Address for Server. Please change this value.
#define TCPPORT (9022)
#define BUFFER_SIZE     (2048)
#define MAX_COUNT       (10000)
#define FETCH_TIME      (600)
#define NOCNT           "NOCNT\0"
#define CNT_BUFFER_SIZE (100)
#define LEDON           (1)
#define LEDOFF          (0)

// #define      DEBUG   1
```

```
int     timeover = 0;

extern void     ClientSendData(char *,char *,int);

void    timeisover()
{
timeover = 1;
}

#if 1
void    LEDAction(void)
{
int     fret;

if(0 > (fret = fork())) // fork Error
            exit(1);
else if (0 < fret)      // Parent
            {
            (void)signal(SIGCHLD,SIG_IGN);
            return;
            }
else if(0 == fret)      // Child
            {
            digitalWrite(LED,LEDOFF);
            delay(300);
            digitalWrite(LED,LEDON);
            exit(0);
            }
}
#else
void    LEDAction(void)
{
digitalWrite(LED,LEDOFF);
}
#endif

int main(void){
            int data,datap,i;

#ifndef DEBUG
            if(0 != fork())
                        exit(0);
#endif

            if(0 > wiringPiSetup())
                        {
                        printf("No wiringPi.\n");
                        return 1;
```

```
                }

        pinMode(PORT,INPUT);
        pinMode(LED ,OUTPUT);

        datap = digitalRead(PORT);
        i = 0;

#ifdef  DEBUG
        printf("Now Status=[%d]\n",datap);
#endif

        (void)signal(SIGALRM,timeisover);
        alarm(FETCH_TIME);
        timeover = 0;

        ClientSendData("00000\0",IPADDR,TCPPORT); // Send Activated Sign for the Server.
        digitalWrite(LED,1);                    // Connection OK. LED ON.

        for(;;){
                if(timeover)
                        {
                        char bufx[CNT_BUFFER_SIZE + 2];

                        sprintf(bufx,"%05d\0",i);               // Counter Value writeout.
                        i = 0;                                  // Counter Reset.
                        ClientSendData(bufx,IPADDR,TCPPORT);    // Send Data to Server.
#ifdef DEBUG
                        printf("SEND[%s]\n",bufx);
#endif
                        (void)signal(SIGALRM,timeisover);       // Set next signal.
                        alarm(FETCH_TIME);                      // Set next fetch time.
                        timeover = 0;                           // Reset TIMEOUT flag.
                        continue;
                        }

                data = digitalRead(PORT);

                if(datap != data)
                        {
                        char    buffer[BUFFER_SIZE +2];

                        datap = data;

                        if(data != 0)
                                {
                                i++;
#ifdef  DEBUG
                                printf("[%06d]GPIO%d = %d\n",i,PORT,data);
```

```
#endif
                                        LEDAction();
                                        }
                                }
        }
        return 0;
}
```

●メインプログラム内部で使われているサーバーとの通信部分のプログラム**(tcpcli.c)**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

/* Global valiables */

struct sockaddr_in client_addr; // write socket

/*
        Connect to Server
*/

int     ConnectServer(char *ipaddr, int port)
{
int     sockfd;         // Listen Socket File Descripter

if(0 > (sockfd = socket(PF_INET, SOCK_STREAM,0))) // Make Socket
        {
        perror("reader: socket");
        return (-1);
        }

bzero((char *)(&client_addr),sizeof(client_addr));      // clear struct for read
client_addr.sin_family = PF_INET;                       // set value to struct
client_addr.sin_addr.s_addr = inet_addr(ipaddr);        // set value to struct
client_addr.sin_port = htons(port);                     // set value to struct

if(0 > connect(sockfd,(struct sockaddr *)(&client_addr),sizeof(client_addr)))
```

```
            {
            perror("Client: Cannot Connect\n");
            close(sockfd);
            return (-1);
            }

return sockfd;
}

/*
            Send Data to Server
*/

int     SendDataToServer(int fd,char *data,int datalen)
{
if(0 > write(fd,data,datalen))
            {
            perror("Client: Cannot Send.\n");
            close(fd);
            return (-1);
            }

return datalen;
}

/*
Client systen Send Data
*/

int     ClientSendData(char *bufx,char *ipaddrx,int portx)
{
int     sockax;
int     len;
char    cntbuff[256];

if(0 > (sockax = ConnectServer(ipaddrx,portx)))
            {
            printf("CLIENT: Cannot Connect Server.\n");
            return(-1);
            }

len = strlen(bufx);
sprintf(cntbuff,"%-6d\0",len);

SendDataToServer(sockax,cntbuff,CNTLEN);        // Send Data Length.
SendDataToServer(sockax,bufx,len);              // Send data.
close(sockax);                                  // Close Socket.
return len;
}
```

※コンパイル時に若干 **Warninng** が出ます。これは、**TCP/IP** のライブラリの新旧によるもので、通常は問題なく動きますが、問題がある場合は **Warninng** が出た部分を書き換えてください。

【サーバー側プログラム(サンプル)】

●コンパイルのためのシェルスクリプト

cc xxxx.c tcpserv.c -o xxxx

プログラムは「xxxx.c」と「tcpserv.c」、出来上がるバイナリプログラムは「xxxx」です。

●メインプログラム(xxxx.c)

※ヘッダの部分は各自の環境に合わせて書き換えてください。望むべくは、外部パラメータにして、設定するように、プログラムを書き換えたいですが、今回は緊急なので、エッセンスをわかっていただくためにも、簡単なプログラムにしています。

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <string.h>

// #define      DEBUG   1               // If debug, Set 1

#define PORT            (9022)
#define LOW_VALUE       (45)

extern int      ServerInit(int);
extern int      WaitIncomming(int);
extern void     ServerReadDatas(char *,int);
extern char     *AcceptedIP;

#define         BUFFER_SIZE     (2048)
#define         FETCH_TIME      (600)   // fetch time is 1-min
#define         HTML_FILE       "/var/www/xxxx/human/index.html"        // Result HTML file
#define         HTML_URL        "http://xxxx.or.jp/human/index.html"    // Result HTML URL
#define         EMAIL_FILE      "/home/xxxx/human/email"                // Email text temporaly file
#define         TO_LINE         "To:%s\n"
#define         TO_ADDRESS      "xxxx@xxxx.com"                         // Email Send to xxxx@xxxx.com
#define         FROM_LINE       "From:%s\n"
#define         FROM_ADDRESS    "yyyy@yyyy.com"                 // Email From Address
#define         SUBJECT_LINE    "Subject:%s\n"
#define         SUBJECT         "Human Moving Report"
#define         NOCNT           "NOCNT\0"
#define         NOT_ACTIVE      (2)
#define         ACTIVE          (500)
#define         CNTBUF_NUM      (20)
#define         BAR_LEN         (100)
#define         BAR_RATE        (100)
#define         NETSTAT_FILE    "/home/xxxx/human/netstat.txt"          // Netstat temporaly file
#define         NETSTAT_COMMAND "/bin/netstat -t -n | /bin/grep 192.168.1 | /bin/grep %d > %s"
```

```
int             cntbuf[CNTBUF_NUM];

void    Clearcntbuf(void)
{
int     i;

for(i = 0 ; i < CNTBUF_NUM ; ++i)
            cntbuf[i] = 0;
}


// SendEmail Real

/*
            Send E-Mail
*/


void    SendEmail(char *body)
{
FILE    *fp;
FILE    *fpin;
int     i;
char    EmailCommandLine[BUFFER_SIZE + 1];

if((FILE *)NULL == (fp = fopen(EMAIL_FILE,"w")))
            {
            printf("Cannot open Email File[%s]\n",EMAIL_FILE);
            exit(1);
            }

sprintf(EmailCommandLine,NETSTAT_COMMAND,PORT,NETSTAT_FILE);
system(EmailCommandLine);

if((FILE *)NULL == (fpin = fopen(NETSTAT_FILE,"r")))
            {
            printf("Cannt Open NETSTAT_FILE[%s]\n",NETSTAT_FILE);
            exit(1);
            }

fprintf(fp,TO_LINE,TO_ADDRESS);
fprintf(fp,FROM_LINE,FROM_ADDRESS);
fprintf(fp,SUBJECT_LINE,SUBJECT);
fprintf(fp,"\n\n");

for(i = 0;i < 100 ; ++i)
            {
            char *pp;
            char NetstatLine[BUFFER_SIZE];
```

```
        if((char *)NULL == fgets(NetstatLine,BUFFER_SIZE,fpin))
                {
                fclose(fpin);
                break;
                }
        else
                {
                fprintf(fp,"%s",NetstatLine);
                }
        }

fprintf(fp,"\n\n");
fprintf(fp,"%s\n\n%s\n\n.\n\n",body,HTML_URL);

fclose(fp);

sprintf(EmailCommandLine,"/usr/sbin/sendmail %s < %s",TO_ADDRESS,EMAIL_FILE);
system(EmailCommandLine);
}

// Send Email Interval

void    SendEmailInterval(int mincnt)
{
char    buffern[BUFFER_SIZE + 2];
char    *p;
time_t  now;
char    *nowtime;

(void)time(&now);               /* Get time for now */
nowtime = ctime(&now);          /* convert to strings */

for(p = nowtime ; (*p) != (char)0 ; p++)
        if((char)('\n') == (*p))
                {
                *p = (char)0;
                break;
                }

#ifdef DEBUG
printf("[%s] Count=%d in %d sec.\n",nowtime,mincnt,FETCH_TIME);
#endif

if(NOT_ACTIVE > mincnt)
        sprintf(buffern,"[%s] NOT-ACTIVE: X In %d sec. there are %d actions.\
n",nowtime,FETCH_TIME,mincnt);
else if(ACTIVE > mincnt)
```

```
            sprintf(buffern,"[%s] ----ACTIVE: *** In %d sec. there are %d actions.\
n",nowtime,FETCH_TIME,mincnt);
else
            sprintf(buffern,"[%s] EXT-ACTIVE: ***** In %d sec. there are %d actions.\
n",nowtime,FETCH_TIME,mincnt);

SendEmail(buffern);
}

// Make Web Pages

void    MakeWebPage(int cnt_fetch)
{
FILE    *fp;
time_t  now;
char    *nowtime;
char    *p;
int     i;

for(i = (CNTBUF_NUM - 1) ; i > 0 ; --i)
            cntbuf[i] = cntbuf[i - 1];

cntbuf[0] = cnt_fetch;

if((FILE *)NULL == (fp = fopen(HTML_FILE,"w")))
            {
            printf("Cannot open FILE[%s]\n",HTML_FILE);
            exit(1);
            }

fprintf(fp,"<HTML>\n");
fprintf(fp,"<HEAD>\n");
fprintf(fp,"<meta http-equiv=\"Refresh\" content=\"%d\">\n",FETCH_TIME);
fprintf(fp,"<TITLE>Human Alive Check</TITLE>\n");
fprintf(fp,"</HEAD>\n");
fprintf(fp,"<BODY>\n");

(void)time(&now);               /* Get time for now */
nowtime = ctime(&now);          /* convert to strings */

for(p = nowtime ; (*p) != (char)0 ; p++)
            if((char)('\n') == (*p))
                        {
                        *p = (char)0;
                        break;
                        }

for(i = 0 ; i < CNTBUF_NUM ; ++i)
            {
```

```
		int blen;
		char bar[BAR_LEN + 2];
		int ix;

		if(LOW_VALUE >= cntbuf[i])
			{
			bar[0] =(char)('-');
			bar[1] =(char)0;
			}
		else if(BAR_RATE > cntbuf[i])
			{
			bar[0] = (char)0;
			}
		else
			{
			blen = cntbuf[i] / BAR_RATE;

			for(ix = 0 ; ix < blen ; ++ix)
				{
				bar[ix] = (char)('*');
				}

			bar[blen] = (char)0;
			}
		fprintf(fp,"[%03d min ago] [%06d] %s<br>\n",(i * 10),cntbuf[i],bar);
		}

fprintf(fp,"<br>\n");
fprintf(fp,"IP=[%s]",AcceptedIP);
fprintf(fp,"<br>\n");

if(0 < cnt_fetch)
		fprintf(fp,"** [%s] ALIVE. COUNT=[%d]<br><br>\n",nowtime,cnt_fetch);
else
		fprintf(fp,"** [%s] NOT ALIVE?<br><br>\n",nowtime);

fprintf(fp,"</BODY>\n");
fprintf(fp,"</HTML>\n");

fclose(fp);
}


// TIMEOUT FUNCTION


int main()
{
int	socka,sockb;
```

```
char    buffer[BUFFER_SIZE+2];

#ifndef DEBUG
if(0 != fork())         /* if it's a parent, then quit */
        exit(0);
#endif

Clearcntbuf();          /* Count Buffer Clear */

for(;;)
        {
        if(0 > (socka = ServerInit(PORT)))
                {
                printf("cannot init TCP/IP.\n");
                exit(1);
                }

        if(0 > (sockb = WaitIncomming(socka)))
                {
                printf("No Datas.\n");
                exit(1);
                }

        ServerReadDatas(buffer,sockb);

#ifdef  DEBUG
        printf("[%s]\n",buffer);
#endif
        close(socka);
        close(sockb);

        SendEmailInterval(atoi(buffer));
        MakeWebPage(atoi(buffer));
        }
}
```

●通信部分プログラム(tcpserv.c)

```
/*  TCP Server */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
```

```
#include <netinet/in.h>
#include <netdb.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <malloc.h>

int     writer_len;     // Data Write Length
struct sockaddr_in reader_addr; // read socket
struct sockaddr writer_addr; // write socket
int     readedbytes;
int     readbytes;
char    *p;
char    AcceptedIP[256];

#define CNT_BYTES       (6)
#define TESTPORT        (8001)

/*
            Read specified bytes from client
*/

void    ReadBytes(int fd,char *buff,int bytes)
{
int     readbytes;
int     readedbytes;
char    *p;

p = buff;
readbytes = bytes;
readedbytes = 0;
bzero(buff,bytes);

for(;;)
            {
            if(0 > (readedbytes = read(fd,p,readbytes)))
                        {
                        printf("ReadError in Line %d.\n",__LINE__);
                        }

            if(bytes > readbytes)
                        {
                        p = p + readedbytes;
                        readbytes = readbytes - readedbytes;
                        }
            else
                        break;
            }
}
```

```
/*
        Server Ready
*/

int     ServerInit(int port)    // return Socket
{
int     yes = 1;
int     usv;
int     sockfd;

if(0 > (sockfd = socket(PF_INET, SOCK_STREAM,0))) // Make Socket
        {
        perror("SERV: reader: socket");
        return -1;
        }

bzero((char *)(&reader_addr),sizeof(reader_addr));      // clear struct for read
reader_addr.sin_family = PF_INET;                       // set value to struct
reader_addr.sin_addr.s_addr = htonl(INADDR_ANY);        // set value to struct
reader_addr.sin_port = htons(port);                     // set value to struct

setsockopt(sockfd,
 SOL_SOCKET,SO_REUSEADDR,(const char *)&yes,sizeof(yes));       // Set Socket options

if(0 > bind(sockfd,(struct sockaddr *)(&reader_addr),sizeof(reader_addr)))      // set bind
        {
        perror("SERV: reader: bind");
        return -1;
        }

if(0 != listen(sockfd,50))      // wait the connection
        {
        perror("SERV: reader: listen");
        close(sockfd);
        return -1;
        }

return sockfd;
}

int     WaitIncomming(int waitsock)
{
int     new_sockfd;


writer_len = sizeof(writer_addr);
```

```
if(0 > (new_sockfd = accept(waitsock,(struct sockaddr *)(&(writer_addr)),(socklen_t *)
(&(writer_len)))))
        {
        perror("SERV: reader: accept");
        return(-1);
        }

bzero(AcceptedIP,255);
// (void)strcpy(AcceptedIP,(char *)inet_ntoa(writer_addr.sin_addr));

return new_sockfd;
}

void    ServerReadDatas(char *DataBuffer,int fda)
{
int     cnt;
int     len;
char    counter[CNT_BYTES+2];

ReadBytes(fda,counter,CNT_BYTES);
counter[CNT_BYTES]=(char)0;
len = atoi(counter);

ReadBytes(fda,DataBuffer,len);
DataBuffer[len] = (char)0;
}
```

※慌てて作ったので、コメントアウトしたものとか、デバッグメッセージとか残っていますが、当然、外して考えていただければと。