Lampiran A :

1.1 Membangun dynamic link libraries (dll) dengan Visual C++ 6.0 dengan output “libglobal.dll”

1.2 Program on.exe

1.3 Program off.exe

1.4 Program TA.tcl

LAMPIRAN A

1.1 Membangun dynamic link libraries (dll) dengan Visual C++ 6.0 dengan output "libglobal.dll"

```
#include <math.h>
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include "snack.h"
#include <tcl.h>
#define beki 9
#define nlpc 240
#define inv 1.0f
#define inv2 -1.0f
static unsigned int dist_I[300];
static unsigned int dist_J[300];
static unsigned int st_cs[10],st_ce[10],in_cs[10],in_ce[10];
static float G[300][300];
static float st_CEP[300][16];
static float in_CEP[300][16];
static float out_pow_in[5000],out_pow_1[5000],out_pow_2[5000],out_pow_3[5000];
static int st_blk,in_blk;
float DP_match1(int);
float distance(int,int);
void trace();

int Signal(ClientData cdata, Tcl_Interp *interp, int objc, Tcl_Obj *CONST objv[])
{
                        Sound *sound;
                        int i;
                        sound = Snack_GetSound(interp, Tcl_GetStringFromObj(objv[0],
NULL));
                        for(i=0;i<Snack_GetLength(sound);i++)
                        {
                                    if((i/10)%2)
                                    {
                                    Snack_SetSample(sound,0,i,10000);
                                    }
                                    else
                                    {
                                    Snack_SetSample(sound,0,i,-10000);
                                    }
                        }
return TCL_OK;
}


int Reset(ClientData cdata, Tcl_Interp *interp, int objc, Tcl_Obj *CONST objv[])
{
                        FILE *freset,*freset1;
                        freset=fopen("data\\hasil\\out.txt","w");
                        fprintf(freset,"%d\n",0);
```

```
                freset1=fopen("data\\hasil\\threshold.txt","w");
                fprintf(freset1,"%d\n",0);

                fclose(freset);
                fclose(freset1);
                return TCL_OK;
}

int front_end_clip(xx,yy,jml_sin)
float *xx,*yy;
int jml_sin;
{
                int i,mulai,akhir,jml_sin_new;
                float jml=0,mean,dev,jumdev=0,sd,batas;

                for(i=0;i<jml_sin;i++)
                        jml+=xx[i];
                mean=jml/jml_sin;
                for(i=0;i<jml_sin;i++)
                {
                        dev=(float)fabs(xx[i]-mean);
                        jumdev+=dev*dev;
                }

                sd=(float)sqrt(jumdev/jml_sin);
                batas=mean+sd;
                for(i=0;i<jml_sin;i++)
                {
                        if(xx[i]>=batas)
                        {
                                mulai=i;
                                break;
                        }
                }

                for(i=jml_sin;i>0;i--)
                {
                        if(xx[i]>=batas)
                        {
                                akhir=i;
                                break;
                        }
                }

                jml_sin_new=0;
                for(i=mulai;i<akhir;i++)
                {
                        yy[jml_sin_new]=xx[i];
                        jml_sin_new++;
                }

                return(jml_sin_new);
}
```

```
float Power(data)
float *data;
{
		int i;
		float jum=0,power;
		for(i=0;i<240;i++)
			jum+=(float)pow(data[i],2);
			power=(float)sqrt(jum);
		return(power);
}


FILE *fsignal,*finfo,*fframe,*fsignalframe,*fpower,*fhamm,*ffft,*flift,*flog,*fceps;
Sound *sound;
double temp_log1;
float x[30000],y[30000],temp[30000];
float real[512],imag[512];
float ms,spd;
int k,frame,frame_no,signal_length,signal_length_new,start;
void hammr();
void cal_fft();


int Input(ClientData cdata, Tcl_Interp *interp, int objc, Tcl_Obj *CONST objv[])
{
	fsignal=fopen("data\\input\\sinyal_in.txt","w");
	finfo=fopen("data\\input\\info_in.txt","w");
	fframe=fopen("data\\input\\frame_in.txt","w");
	fsignalframe=fopen("data\\input\\signalframe_in.txt","w");
	fpower=fopen("data\\input\\power.txt","w");
	fhamm=fopen("data\\input\\hamm_in.txt","w");
	ffft=fopen("data\\input\\fft_in.txt","w");
	flift=fopen("data\\input\\lift_in.txt","w");
	flog=fopen("data\\input\\log_in.txt","w");
	fceps=fopen("data\\input\\ceps_in.txt","w");


		/****** Get the sound structure for this sound ******/
		sound = Snack_GetSound(interp, Tcl_GetStringFromObj(objv[0], NULL));
		signal_length = Snack_GetLength(sound);
		for (k=0;k<signal_length;k++)
			{
				x[k]=(float)Snack_GetSample(sound, 0, k);
				fprintf(fsignal,"%f\n",x[k]);
			}

		signal_length_new = front_end_clip(x,&y,signal_length);
		for (k=0;k<signal_length_new;k++)
			fprintf(finfo,"%f\n",y[k]);
		frame_no = signal_length_new/120;
		fprintf(fframe,"%d\n",frame_no);
		start=0;
		ms=0;
		spd=(float)20/240;
		for(frame=0;frame<frame_no-1;frame++)
			{
```

```
                start=(frame)*120;
                for(k=0;k<240;k++)
                        {
                                temp[k]=y[start];
                                fprintf(fsignalframe,"%f\n",temp[k]);
                                start++;
                        }
                out_pow_in[frame]=Power(temp);
                fprintf(fpower,"%f\n",out_pow_in[frame]);
        }

for(frame=0;frame<frame_no-1;frame++)
        {
                for (k=0;k<240;k++)
                        {
                                real[k]=temp[k];
                                imag[k]=0;
                        }


/***** hamming window *****/
hammr(real,imag,nlpc);
for(k=240;k<256;k++)
        fprintf(fhamm,"%f\t%f\n",ms=ms+spd,real[k]);
for(k=256;k<512;k++)
        {
                real[k]=0;
                imag[k]=0;
        }

/***** FFT *****/
cal_fft(real,imag,beki,inv);

/***** Mutlak Log *****/
for(k=0;k<512;k++)
        {
                temp_log1=(double)(real[k]*real[k]+imag[k]*imag[k]);
                if(temp_log1>0)
                        real[k]=(float)log10(temp_log1);
                else
                        real[k]=(float)log10(0.000001);
                        imag[k]=0;
                        fprintf(ffft,"%f\t%f\n",k*6000.f/256.f,real[k]);
        }

/***** IFFT *****/
cal_fft(real,imag,beki,inv2);

/***** Liftering *****/
for(k=16;k<512-16;k++)
        {
                real[k]=0.0;
                imag[k]=0.0;
        }

for(k=0;k<256;k++)
```

```
                    fprintf(flift,"%f\t%f\n",k*6000.f/256.f,real[k]);

                /***** Mutlak Log (Data Sample yg diambil) *****/
                for(k=0;k<256;k++)
                    {
                        temp_log1=(double)(real[k]*real[k]+imag[k]*imag[k]);
                        if(temp_log1>0)
                            real[k]=(float)log10(temp_log1);
                        else
                            real[k]=(float)log10(0.000001);
                            imag[k]=0;
                            fprintf(flog,"%f\t%f\n",k*6000.f/256.f,real[k]);
                    }

                /***** Cepstrum *****/
                cal_fft(real,imag,beki,inv);
                for(k=0;k<256;k++)
                    {
                        temp_log1=(double)(real[k]*real[k]+imag[k]*imag[k]);
                        if(temp_log1>0)
                            real[k]=(float)log10(temp_log1);
                        else
                            real[k]=(float)log10(0.000001);
                            imag[k]=0;
                            fprintf(fceps,"%f\t%f\n",k*6000.f/256.f,real[k]);
                    }
                    }

fclose(fsignal);
fclose(finfo);
fclose(fframe);
fclose(fsignalframe);
fclose(fpower);
fclose(fhamm);
fclose(ffft);
fclose(flift);
fclose(flog);
fclose(fceps);
return TCL_OK;
}


FILE *fsignal,*finfo,*fframe,*fsignalframe,*fpower,*fhamm,*ffft,*flift,*flog,*fceps;
Sound *sound;
double temp_log1;
float x[30000],y[30000],temp[30000];
float real[512],imag[512];
float ms,spd;
int k,frame,frame_no,signal_length,signal_length_new,start;
void hammr();
void cal_fft();


int dtbase(ClientData cdata, Tcl_Interp *interp, int objc, Tcl_Obj *CONST objv[])
{
        fsignal=fopen("data\\standart\\sinyal_w1.txt","w");
```

```
finfo=fopen("data\\standart\\info_w1.txt","w");
fframe=fopen("data\\standart\\frame_w1.txt","w");
fsignalframe=fopen("data\\standart\\signalframe_w1.txt","w");
fpower=fopen("data\\standart\\power_w1.txt","w");
fhamm=fopen("data\\standart\\hamm_w1.txt","w");
ffft=fopen("data\\standart\\fft_w1.txt","w");
flift=fopen("data\\standart\\lift_w1.txt","w");
flog=fopen("data\\standart\\log_w1.txt","w");
fceps=fopen("data\\standart\\ceps_w1.txt","w");

        /****** Get the sound structure for this sound ******/
        sound = Snack_GetSound(interp, Tcl_GetStringFromObj(objv[0], NULL));
        signal_length = Snack_GetLength(sound);
        for (k=0;k<signal_length;k++)
                {
                        x[k]=(float)Snack_GetSample(sound, 0, k);
                        fprintf(fsignal,"%f\n",x[k]);
                }

        signal_length_new = front_end_clip(x,&y,signal_length);
        for (k=0;k<signal_length_new;k++)
                fprintf(finfo,"%f\n",y[k]);
        frame_no = signal_length_new/120;
        fprintf(fframe,"%d\n",frame_no);
        start=0;
        ms=0;
        spd=(float)20/240;
        for(frame=0;frame<frame_no-1;frame++)
                {
                        start=(frame)*120;
                        for(k=0;k<240;k++)
                                {
                                        temp[k]=y[start];
                                        fprintf(fsignalframe,"%f\n",temp[k]);
                                        start++;
                                }
                        out_pow_in[frame]=Power(temp);
                        fprintf(fpower,"%f\n",out_pow_in[frame]);
                }

        for(frame=0;frame<frame_no-1;frame++)
                {
                        for (k=0;k<240;k++)
                                {
                                        real[k]=temp[k];
                                        imag[k]=0;
                                }

        /***** hamming window *****/
        hammr(real,imag,nlpc);
        for(k=240;k<512;k++)
                fprintf(fhamm,"%f\t%f\n",ms=ms+spd,real[k]);
        for(k=240;k<512;k++)
                {
                        real[k]=0;
                        imag[k]=0;
```

```
        }

/***** FFT *****/
cal_fft(real,imag,beki,inv);

/***** Mutlak Log *****/
for(k=0;k<512;k++)
        {
                temp_log1=(double)(real[k]*real[k]+imag[k]*imag[k]);
                if(temp_log1>0)
                        real[k]=(float)log10(temp_log1);
                else
                        real[k]=(float)log10(0.000001);
                        imag[k]=0;
                        fprintf(ffft,"%f\t%f\n",k*6000.f/256.f,real[k]);
        }

/***** IFFT *****/
cal_fft(real,imag,beki,inv2);

/***** Liftering *****/
for(k=16;k<512-16;k++)
        {
                real[k]=0.0;
                imag[k]=0.0;
        }

for(k=0;k<256;k++)
        fprintf(flift,"%f\t%f\n",k*6000.f/256.f,real[k]);

/***** Mutlak Log (Data Sample yg diambil) *****/
for(k=0;k<256;k++)
        {
                temp_log1=(double)(real[k]*real[k]+imag[k]*imag[k]);
                if(temp_log1>0)
                        real[k]=(float)log10(temp_log1);
                else
                        real[k]=(float)log10(0.000001);
                        imag[k]=0;
                        fprintf(flog,"%f\t%f\n",k*6000.f/256.f,real[k]);
        }

/***** Cepstrum *****/
cal_fft(real,imag,beki,inv);
for(k=0;k<256;k++)
        {
                temp_log1=(double)(real[k]*real[k]+imag[k]*imag[k]);
                if(temp_log1>0)
                        real[k]=(float)log10(temp_log1);
                else
                        real[k]=(float)log10(0.000001);
                        imag[k]=0;
                        fprintf(fceps,"%f\t%f\n",k*6000.f/256.f,real[k]);
        }
}
```

```
fclose(fsignal);
fclose(finfo);
fclose(fframe);
fclose(fsignalframe);
fclose(fpower);
fclose(fhamm);
fclose(ffft);
fclose(flift);
fclose(flog);
fclose(fceps);
return TCL_OK;
}


int Dtw(ClientData cdata, Tcl_Interp *interp, int objc, Tcl_Obj *CONST objv[])
{
	FILE *fin,*fstd1,*fstd2;
	FILE *fframe,*fout1,*fout2,*frata1,*fthreshold;
	int i,j,r;
	static int out=1;
	float DP_dist,hasil[101],min,rata,rata1,threshold;
	float in_data,std_data_1,std_data_2;
	float in_in,in_1,in_2;
	fin=fopen("data\\input\\log_in.txt","r");
	fstd1=fopen("data\\standart\\log_w1.txt","r");
	fstd2=fopen("data\\standart\\log_w2.txt","r");
	fout1=fopen("data\\hasil\\out1.txt","a");
	fout2=fopen("data\\hasil\\out.txt","w");
	frata1=fopen("data\\hasil\\rata.txt","w");
	fthreshold=fopen("data\\hasil\\threshold.txt","w");
	fframe=fopen("data\\input\\frame_in.txt","r");
	fscanf(fframe,"%d\n",&in_blk);
	fclose(fframe);
	for(i=0;i<in_blk;i++)
		{
			for(j=0;j<16;j++)
				{
					fscanf(fin,"%f\t%f\n",&in_in,&in_data);
					in_CEP[i][j]=in_data;
				}
		}

		/***** Menghitung jarak antara sinyal input dan sinyal standard 1 *****/
		fframe=fopen("data\\standart\\frame_w1.txt","r");
		fscanf(fframe,"%d\n",&st_blk);
		fclose(fframe);
		for(i=0;i<st_blk;i++)
			{
				for(j=0;j<16;j++)
					{

	fscanf(fstd1,"%f\t%f\n",&in_1,&std_data_1);
							st_CEP[i][j]=std_data_1;
					}
			}
```

```
		r=7;
		DP_dist=DP_match1(r);
		hasil[1]=DP_dist;
		trace();

		/***** Menghitung jarak antara sinyal input dan sinyal standard 2 *****/
		fframe=fopen("data\\standart\\frame_w2.txt","r");
		fscanf(fframe,"%d\n",&st_blk);
		fclose(fframe);
		for(i=0;i<st_blk;i++)
			{
				for(j=0;j<16;j++)
					{

fscanf(fstd2,"%f\t%f\n",&in_2,&std_data_2);
							st_CEP[i][j]=std_data_2;
					}
			}
		r=7;
		DP_dist=DP_match1(r);
		hasil[2]=DP_dist;
//		trace();
		for(i=1;i<=2;i++)
			fprintf(fout1,"%f\n",hasil[i]);
		min=hasil[1];
		for(i=1;i<=2;i++)
			{
				if(min>=hasil[i])
					{
						min=hasil[i];
						out=i;
					}
			}
		threshold=hasil[out];
		rata=0;
		for(i=1;i<=2;i++)
			{
				rata+=hasil[i];
			}
		rata1=rata/2;
		fprintf(fout2,"%d\n",out);
		fprintf(fthreshold,"%f\n",hasil[out]);
		fprintf(frata1,"%f\n",rata1);
		fclose(fin);
		fclose(fstd1);
		fclose(fout1);
		fclose(fout2);
		fclose(frata1);
		fclose(fthreshold);
		return TCL_OK;
}


	/*
	Initialize the square package and create a new sound command 'global'.
	The syntax is: sndName global
```

```
	*/
	EXPORT(int, Global_Init)(Tcl_Interp *interp)
	{
	#ifdef USE_TCL_STUBS
	if (Tcl_InitStubs(interp, "8", 0) == NULL) {
	return TCL_ERROR;
	}
	#endif

	#ifdef USE_SNACK_STUBS
	if (Snack_InitStubs(interp, "2", 0) == NULL) {
	return TCL_ERROR;
	}
	#endif

	if (Tcl_PkgProvide(interp, "global", "1.0") != TCL_OK) {
	return TCL_ERROR;
	}

	Snack_AddSubCmd(SNACK_SOUND_CMD, "input",(Snack_CmdProc *)Input,NULL);
	Snack_AddSubCmd(SNACK_SOUND_CMD, "dtbase",(Snack_CmdProc
*)dtbase,NULL);
	Snack_AddSubCmd(SNACK_SOUND_CMD, "dtw",(Snack_CmdProc *)Dtw,NULL);
	Snack_AddSubCmd(SNACK_SOUND_CMD, "reset",(Snack_CmdProc *)Reset,NULL);
	Snack_AddSubCmd(SNACK_SOUND_CMD, "signal",(Snack_CmdProc
*)Signal,NULL);
	return TCL_OK;
	}


	EXPORT(int, Global_SafeInit)(Tcl_Interp *interp)
	{
	return Global_Init(interp);
	}
	#ifdef __cplusplus
	#endif


void hammr(x,y,n)
float *x,*y;
int n;
{
	int i;
	--x;
	--y;
	for(i=0;i<=n;++i)
		{
			x[i]=x[i]*(float)(0.54-0.46*cos(2.0*3.141592654*(i)/(float)n));
			y[i]=y[i]*(float)(0.54-0.46*cos(2.0*3.141592654*(i)/(float)n));
		}
}


void cal_fft(x,y,l,mode)
float *x,*y,mode;
int l;
```

```
{
        int np,lmx,lo,lix,lm,li,j1,j2,nv2,npm1,i,j,k;
        float scl,arg,c,s,t1,t2;
        for(i=0;i<pow(2,l);i++)
                y[i]=0;
        --x;
        --y;

        /* ====== radix-2 fft ====== */
        np=(int)pow(2.0,(float)(l));
        lmx=np;
        scl=(float)(6.283185303/(float)np);
        for(lo=1;lo<=l;++lo)
                {
                        lix=lmx;
                        lmx=(int)(lmx/2.0);
                        arg=0.0;
                        for(lm=1;lm<=lmx;++lm)
                                {
                                        c=(float)cos(arg);
                                        s=(float)(mode*sin(arg));
                                        arg=arg+scl;
                                        for (li=lix; lix<0 ? li>=np : li<=np; li+=lix)
                                                {
                                                        j1=li-lix+lm;
                                                        j2=j1+lmx;
                                                        t1=x[j1]-x[j2];
                                                        t2=y[j1]-y[j2];
                                                        x[j1]=x[j1]+x[j2];
                                                        y[j1]=y[j1]+y[j2];
                                                        x[j2]=c*t1+s*t2;
                                                        y[j2]=c*t2-s*t1;
                                                }

                                }

                        scl=(float)(2.0*scl);
                }

        /* ====== bit reversal ====== */
        j=1;
        nv2=(int)(np/2.0);
        npm1=np-1;
        for(i=1;i<=npm1;++i)
                {
                        if(i>=j)
                                goto L30;
                        t1=x[j];
                        t2=y[j];
                        x[j]=x[i];
                        y[j]=y[i];
                        x[i]=t1;
                        y[i]=t2;
                L30:
                        k=nv2;
                L40:
```

```
                        if(k>=j)
                                goto L50;
                        j-=k;
                        k=(int)(k/2.0);
                        goto L40;
                L50:
                        j=j+k;
                }
}



float DP_match1(r)
int r;
{
        int i,j;
        int I,J,up,dp;
        float dist,g,g0,g1,g2,g3,a;
        float DP_mdist=0.0;
        for(i=0;i<in_blk;i++)
                {
                        for(j=0;j<st_blk;j++)
                                {
                                        G[i][j] = (float)1.0e+30;
                                }

                }

        I=in_blk-1;
        J=st_blk-1;
        a=(float)st_blk/(float)in_blk;
        dist=distance(0,0);
        G[0][0]=(float)(2.0*dist);
        dist=0.0;
        for(i=0;i<=I;i++)
                {
                        up=(int)(a*i+r);
                        if(up>J)
                                up=J;
                                dp=(int)(a*i-r);
                        if(dp<0)
                                dp=0;
                        for(j=dp;j<=up;j++)
                                {
                                        if(i==0 && j==0)
                                                j++;
                                                g0=(float)1.0e+30;
                                                g1=(float)1.0e+30;
                                                g2=(float)1.0e+30;
                                                dist = distance(i,j);

                                        if(j-1>=0)
                                                g0 = (float)(G[i][j-1]+dist);

                                        if(i-1>=0 && j-1>=0)
                                                g1 = (float)(G[i-1][j-1]+2.0*dist);
```

```
                                        if(i-1>=0)
                                                g2 = (float)(G[i-1][j]+dist);
                                                g3 = (g0<g1) ? g0:g1;
                                                g = (g2<g3) ? g2:g3;
                                                G[i][j] = g;
                                }

                        }

        DP_mdist = G[I][J]/(st_blk+in_blk);
        return(DP_mdist);
}


float distance(ab_t,ab_r)
int ab_t,ab_r;
{
        int i;
        float a,kyori;
        a=0.0;
        kyori=0.0;
        for(i=0;i<16;i++)
                {
                        a=(in_CEP[ab_t][i]-st_CEP[ab_r][i]);
                        kyori+=a*a;
                }

        return(kyori);
}

void trace() /*WARPING FUNCTION decision (in case dp_match1)*/
{
        FILE *fjejak,*fsin_warp;
        FILE *fframe_in,*fframe_w1;
        FILE *fpower,*fpower_w1;
        int i,j,k,l,n,p;
        int slope_i,slope_i1,slope_j;
        int jml_in,jml_1;
        float D0,D1,D2,D3;
        float dist,distance();
        float power_in[5000],power_w1[5000];
        fjejak=fopen("data\\hasil\\jejak.txt","w");
        fframe_in=fopen("data\\input\\frame_in.txt","r");
        fframe_w1=fopen("data\\standart\\frame_w1.txt","r");
        fpower=fopen("data\\input\\power_in.txt","r");
        fpower_w1=fopen("data\\standart\\power_w1.txt","r");
        fsin_warp=fopen("data\\hasil\\sin_warp.txt","w");
        fscanf(fframe_in,"%d\n",&jml_in);
        for(l=0;l<jml_in;l++)
                fscanf(fpower,"%f\n",&power_in[l]);
        fscanf(fframe_w1,"%d\n",&jml_1);
        for(l=0;l<jml_1;l++)
                fscanf(fpower_w1,"%f\n",&power_w1[l]);
        n=in_blk+st_blk-1;
        for(p=0;p<n;p++)
```

```
		{
			dist_I[p]=0;
			dist_J[p]=0;
		}

	i=in_blk-1;
	j=st_blk-1;
	dist_I[0]=i;
	dist_J[0]=j;
	for(k=1;k<n;k++)
		{
			dist=distance(i,j);
			D0=(float)1.0e+30;
			D1=(float)1.0e+30;
			D2=(float)1.0e+30;
			D3=(float)1.0e+30;
			if(i>0)
				{
					D0=G[i-1][j]+dist;
					D3=D0;
				}

			if(j>0)
				{
					if(i>0)
						D1=(float)(G[i-1][j-1]+dist*2.0);
						D2=G[i][j-1]+dist;
						D3=(D1<D2) ? D1:D2;
						slope_i1=(D1<D2) ? 1:0;
				}

			slope_i=(D3<D0) ? slope_i1:1;
			slope_j=(D3<D0) ? 1:0;
			i=i-slope_i;
			j=j-slope_j;
			if(i<0 || j<0)
				break;

			dist_I[k]=i; /*absis dan ordinat jejak dtw */
			dist_J[k]=j;
			fprintf(fjejak,"%d\t%d\n",dist_I[k],dist_J[k]);
			fprintf(fsin_warp,"%f\t%f\n",power_in[jml_in -
			dist_I[k]],power_w1[jml_1 - dist_J[k]]);
		}

	fclose(fjejak);
	fclose(fsin_warp);
}
```

1.2 Program on.exe (untuk menghidupkan lampu LED)

```
#include "stdafx.h"
#include "conio.h"
#include "stdlib.h"

void _stdcall Out32(short PortAddress, short data);
short  _stdcall Inp32(short PortAddress);

int main ()
{
        Out32(0x378,255);
                return 0;
}
```

1.3 Program off.exe (untuk mematikan lampu LED)

```
#include "stdafx.h"
#include "conio.h"
#include "stdlib.h"

void _stdcall Out32(short PortAddress, short data);
short  _stdcall Inp32(short PortAddress);

int main ()
{
        Out32(0x378,0);
                return 0;
}
```

1.4 Program TA.tcl

```
#!/bin/sh
# the next line restarts using wish \
exec wish8.4 "$0" "$@"
# 'info sharedlibext' returns '.dll' on Windows and '.so' on most Unix systems
load lib\\libglobal

snack::sound u
snack::sound v
snack::sound w
snack::sound x
snack::sound z -channels 2
u configure -rate 12000 -channels MONO -encoding LIN16
#image create photo test -speaker.gif

set width 600
set height 200
set pps 300
set color black
set frame 1

option add *font {Helvetica 10 bold}
pack [frame .a1]
```

```
pack [frame .a2]
pack [frame .a3]
pack [frame .a4]
pack [label .a1.t1 -text "VOICE RECOGNITION UNTUK \n DOOR LOCKING"] -pady 10

canvas .a1.canvas -bg black -width 300 -height 200

snack::levelMeter .a1.levelMeter -width 20 -length 200 -orient vertical -oncolor blue
pack .a1.levelMeter .a1.canvas -side left -padx 5

pack [label .a2.time -text "0.000 sec" -width 10]

button .a3.b1 -text Play -command {u play}
button .a3.b2 -text Record -command {u record;z record}
button .a3.b3 -text Stop -command {u stop;u input;u dtw;w length 1000;w signal;w play;z
stop;Announce}
button .a3.b4 -text Reset -command {u reset;Announce}
button .a3.b5 -text Load -command OpenSound
button .a3.b6 -text Save -command SaveSound
button .a3.b7 -text Exit -command exit

pack .a3.b1 .a3.b2 .a3.b3 .a3.b4 .a3.b5 .a3.b6 .a3.b7 -side left -padx 5 -pady 10

pack [label .a4.l -text "Stephanus Arnold \n 0222021 \n Teknik Elektro \n UNIVERSITAS
KRISTEN MARANATHA" -pady 10]

proc OpenSound {} {
   set filename [snack::getOpenFile]
   u configure -file $filename
}


proc SaveSound {} {
   set filename [snack::getSaveFile]
   u write $filename
}

after 1 Update

proc Update {} {
set l [z max -start 0 -end -1 -channel 0]
.a2.time config -text [format "%.3f sec" [u length -unit seconds]]
z length 0
.a1.levelMeter configure -level $l
after 50 Update
}

proc Announce {} {
            set data [read [open data\\hasil\\threshold.txt]]
            set data1 [read [open data\\hasil\\out.txt]]
            if { $data <= 1 } {
            if { $data1 == 1 } {
            exec on &}
            if {$data1 == 2 } {
            exec off &}
}
```

}
.a1.canvas itemconf wave -pixelspersecond 300 -width 300
.a1.canvas create waveform 152 102 -anchor c -sound u -height $height -tags wave -debug 0 -zerolevel 0 -frame $frame -fill red

Keterangan :

// Membangun ekstensi snack untuk Windows menggunakan MS Visual C++ 6.0.

1. *Buka file baru pilih "Win32 Dynamic-Link Library". (File/New...) Beri nama "global" dan click "Ok" lalu Finish.*
2. *Add file global.c ke project (Project/Add to project/Files...)*
3. *Spesifikasi lokasi Snack and Tcl include files. (Project/Settings..., C/C++ tab, Category: Preprocessor, Additional include directories : **C:\Program Files\Tcl\include\,C:\Program Files\Snack2.0\include**)*
4. *Spesifikasi stubs definitions (Project/Settings... C/C++ tab, Category: General, Preprocessor definitions : **USE_SNACK_STUBS, USE_TCL_STUBS, USE_TK_STUBS**)*
5. *Spesifikasi Snack and Tcl link libraries (Link tab, Category: Input, Object/library modules : **tclstub83.lib tkstub83.lib snackstub22.lib**) dan lokasinya (Additional library path : **C:\Program Files\Tcl\lib\,C:\Program Files\Snack2.0\lib**)*
6. *Spesifikasi output file name dari dll to be libglobal.dll. (Category: General, Output file name)*
7. *Build ekstensinya.*