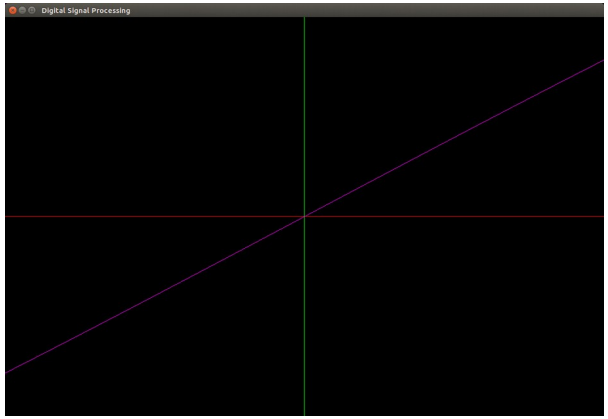
TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018-06-11 (71 회차)

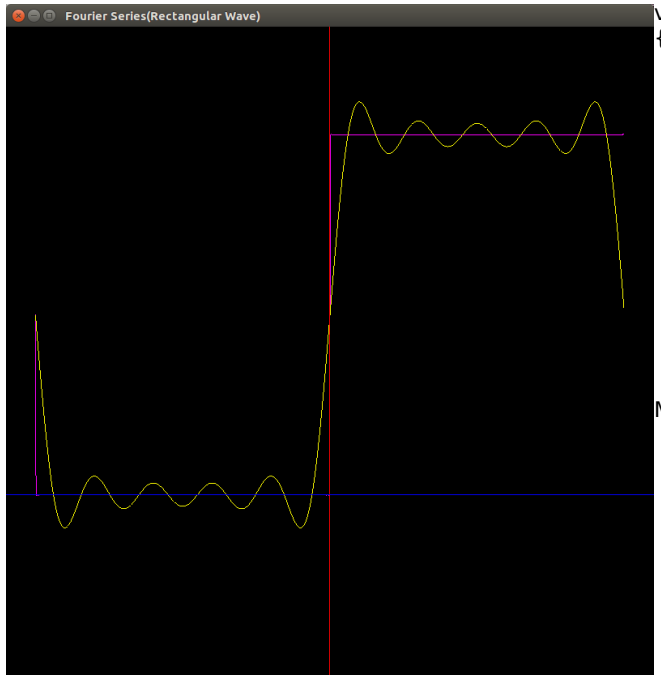
강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - 정유경

gcc fourier_series_line.c -lGL -lGLU -lglut -lm



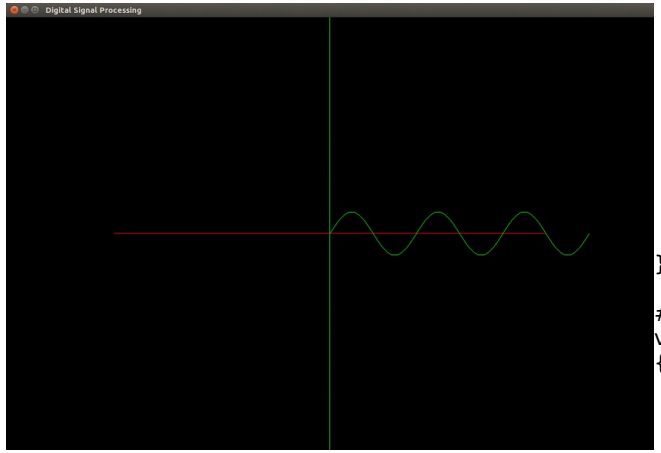

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define _USE_MATH_DEFINES
#include <math.h>
#include <GL/glut.h>
#define SLICE     360

void rect_pulse_signal(void)
{
        float t, T = 0.01;
        float amp = 100;
        float step = 0.0;
        float omega = 2 * M_PI * 100;        // frequency =
100 Hz
        float x = 0, x2 = 0, y, cx, cy;
        int cache = 0;
        int i;

        //t = step = T / 100;
        step = T / 100;
        t = -1 * T;

        //printf("step = %f\n", step);

        if(t > T)
                t = 0.0;

        glColor3f(1.0, 0.0, 1.0);
        glBegin(GL_LINES);
        for(; ; t += step)
        {
                y = 0;

                if(t > 1 * T)
                {
                        break;
                        t = 0.0;
                }

                //y = amp * (sin(omega * t) / (omega * t));
                for(i = 1; i < 101; i++)
                        y += 100 * ((1 - cos(i * M_PI)) / (i
* M_PI) * sin(i * t));

                if(cache)
                {
                        glVertex2f(cx * 6000, cy * 1);
                        glVertex2f(t * 6000, y * 1);
                }

                cache = 1;
                cx = t;
                cy = y;
                //printf("t = %f, y = %f\n", t * 6000, y * 1);
        }
        glEnd();
}

void display(void)
{
        glClearColor(0.0, 0.0, 0.0, 1.0);
        glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();

        //gluLookAt(0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0,
0.0);

        glColor3f(1, 0, 0);

        glBegin(GL_LINE_LOOP);
        glVertex3f(100.0, 0.0, 0.0);
    glVertex3f(-100.0, 0.0, 0.0);
    glEnd();

    glColor3f(0.0, 1.0, 0.0);

    glBegin(GL_LINE_LOOP);
    glVertex3f(0.0, 100.0, 0.0);
    glVertex3f(0.0, -100.0, 0.0);
    glEnd();

        rect_pulse_signal();
        glutSwapBuffers();
}

void reshape(int w, int h)
{
    GLfloat n_range = 20.0f;

    if(h == 0)
        h = 1;

    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    if(w <= h)
        glOrtho(-n_range, n_range, -n_range * h / w,
n_range * h / w, -n_range, n_range);
    else
        glOrtho(-n_range * w / h, n_range * w / h,
-n_range, n_range, -n_range, n_range);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y)
{
        switch(key)
        {
                case 27:
                        exit(0);
                        break;
        }
}

int main(int argc, char **argv)
{
//      glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE);
        glutInitWindowSize(1200, 800);
        glutInitWindowPosition(0, 0);
        glutCreateWindow("Digital Signal Processing");

        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutMainLoop();

        return 0;
}
```

gcc scale_rect_wave.c -lGL -lGLU -lm -lglut




```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <GL/glu.h>
#include <GL/gl.h>
#include <GL/freeglut.h>

void originAxis(void);
void sineWave(void);
void idle(void);

void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);

	originAxis();
	sineWave();
	glutSwapBuffers();
}

void sineWave(void)
{
	float wavelength = 2.0 * M_PI;
	float amplitude = 1;
	float inc = 2.0 * M_PI / 1024.0;
	float k, x, y, yp = 0, y2, y2p = 0, cx, cy, cy2;
	int i, cache = 0;

	glBegin(GL_LINES);
	glColor3f(1,1,0);
	for(x=-M_PI;x<=M_PI;x+=inc)
	{
		yp = 0;

		for(i = 1; i < 10; i++)
			yp += ((1.0 - cos(i * M_PI)) / (i *
M_PI)) * sin(i * x);

		y = yp + 0.5;

		if(cache)
		{
			glVertex2f(cx, cy);
			glVertex2f(x, y);
		}

		cache = 1;
		cx = x;
		cy = y;
	}
	glEnd();

	cache = 0;

	glBegin(GL_LINES);
	glColor3f(1,0,1);
	for(x=-M_PI;x<=M_PI;x+=inc)
	{
		yp = 0;

		for(i = 1; i < 10000; i++)
			yp += ((1.0 - cos(i * M_PI)) / (i *
M_PI)) * sin(i * x);

		y = yp + 0.5;

		if(cache)
		{
			glVertex2f(cx, cy);
			glVertex2f(x, y);
		}

		cache = 1;
		cx = x;
		cy = y;
	}
	glEnd();
}

void originAxis(void)
{
	glBegin(GL_LINES);
	glColor3f(0,0,1);
	glVertex3f(-100,0,0);
	glVertex3f(100, 0, 0);
	glColor3f(1,0,0);
	glVertex3f(0,-100,0);
	glVertex3f(0, 100, 0);
	glColor3f(0,0,1);
	glVertex3f(0,0,0);
	glVertex3f(0, 0, 1);
	glEnd();
}

int main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE |
GLUT_DEPTH);
	glutInitWindowSize(800, 800);
	glutCreateWindow("Fourier Series(Rectangular
Wave)");

	glOrtho(-1.1 * M_PI, 1.1 * M_PI, -0.5, 1.3, -1.0, 1.0);
	glEnable(GL_DEPTH_TEST);

	glutDisplayFunc(display);
	glutMainLoop();

	return EXIT_SUCCESS;
}
```

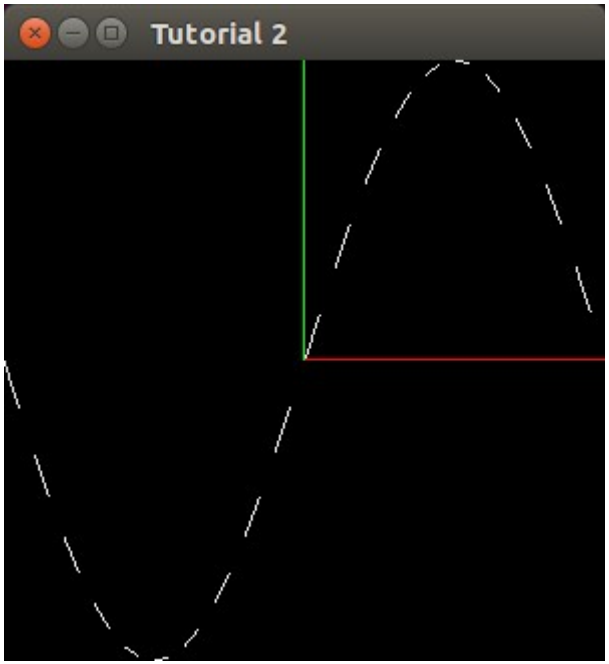
gcc signal_plot.c -lGL -lGLU -lglut -lm




```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#include <GL/glut.h>

#define SLICE     360

void draw_omega_sin(void);

float common_angles[5] = {15.0, 30.0, 45.0, 60.0, 75.0};
float freq_table[5] = {1000.0, 2400.0, 5000.0, 24000.0,
77000.0};

float theta = 0.0;

void display(void)
{
        glClearColor(0.0, 0.0, 0.0, 1.0);
        glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();

        //gluLookAt(0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0,
0.0);
            glColor3f(1, 0, 0);

            glBegin(GL_LINE_LOOP);
            glVertex3f(100.0, 0.0, 0.0);
        glVertex3f(-100.0, 0.0, 0.0);
        glEnd();

        glColor3f(0.0, 1.0, 0.0);

        glBegin(GL_LINE_LOOP);
        glVertex3f(0.0, 100.0, 0.0);
        glVertex3f(0.0, -100.0, 0.0);
        glEnd();

            draw_omega_sin();
            glutSwapBuffers();
}

#if 0
void reshape(int w, int h)
{
        glViewport(0, 0, (GLsizei)w, (GLsizei)h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(60, (GLfloat)w / (GLfloat)h, 0.1,
100.0);
        glMatrixMode(GL_MODELVIEW);
}
#endif

void reshape(int w, int h)
{
      GLfloat n_range = 100.0f;

      if(h == 0)
            h = 1;

      glViewport(0, 0, w, h);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();

      if(w <= h)
            glOrtho(-n_range, n_range, -n_range * h / w,
n_range * h / w, -n_range, n_range);
      else
            glOrtho(-n_range * w / h, n_range * w / h,
-n_range, n_range, -n_range, n_range);

      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y)
{
        switch(key)
        {
                case 27:
                        exit(0);
                        break;
        }
}

void set_rand_amplitude(float *amp)
{
        *amp = rand() % 3 + 3;
}

void set_angle_with_common_angles(float *angle)
{
        *angle = common_angles[rand() % 5];
}

void angle2radian(float *angle, float *radian)
{
        *radian = *angle * M_PI / 180.0;
}

void radian2angle(float *angle, float *radian)
{
        *angle = *radian * 180.0 / M_PI;
}

void set_rand_frequency(float *freq)
{
        *freq = freq_table[rand() % 5];
}

void calc_period(float *freq, float *period)
{
        *period = 1 / (*freq);
}

void calc_angular_velocity(float *freq, float *ang_vel)
{
        *ang_vel = 2 * M_PI * (*freq);
}

float get_step(float slice, float period)
{
        return period / slice;
}
```

```
void cos_sim(float amplitude, float ang_vel, float period)
{
	int cnt = 0;
	float step, t = 0.0;

	t = step = get_step(SLICE, period);

	while(cnt++ < 36)
	{
		printf("%.1fcos(%f * %.8f) = %f\n",
amplitude, ang_vel,
			t, amplitude * cos(ang_vel * t));
		t += step;
	}
}

void sin_sim(float amplitude, float ang_vel, float period)
{
	int cnt = 0;
	float step, t = 0.0;

	t = step = get_step(SLICE, period);

	while(cnt++ < 36)
	{
		printf("%.1fsin(%f * %.8f) = %f\n",
amplitude, ang_vel,
			t, amplitude * sin(ang_vel * t));
		t += step;
	}
}

void draw_omega_sin(void)
{
	float amp, angle, period, freq, rad, omega, t, step
= 0.0;
	float radius = 3.0;
	float x = 0, x2 = 0, y2, cx, cy;
	float tmp;
	int cache = 0;

	srand(time(NULL));

#if 0
	set_rand_amplitude(&amp);
	set_angle_with_common_angles(&angle);
	angle2radian(&angle, &rad);
	set_rand_frequency(&freq);
	calc_period(&freq, &period);
	calc_angular_velocity(&freq, &omega);
#endif

#if 1
	amp = 10;
	angle = 45.0;
	freq = 100.0;

	angle2radian(&angle, &rad);
	calc_period(&freq, &period);
	calc_angular_velocity(&freq, &omega);
#endif

#if 0
	printf("amplitude = %f\n", amp);
	printf("angle = %f degree\n", angle);
	printf("radian = %f\n", rad);
	printf("frequency = %f\n", freq);
	printf("period = %f\n", period);
	printf("angular_velocity = %f\n", omega);
#endif

	t = step = get_step(SLICE, period);

	//printf("t = %f\n", t);
#if 1
	if(t > period)
		t = 0.0;
#endif

	glBegin(GL_LINES);
	for(; ; t += step)
	{
		if(t > 3 * period)
		{
			break;
			t = 0.0;
		}

		//float rad_angle = angle * (M_PI / 180.0);
		//x2 += x;			// time += step;
		//x2 += 0.1;
		y2 = amp * sin(omega * t);
		//y2 = radius * sin((double)rad_angle);

		if(cache)
		{
			glVertex2f(cx * 4000, cy);
			glVertex2f(t * 4000, y2);
		}

		cache = 1;
		cx = t;
		cy = y2;
		//printf("t = %f, y2 = %f\n", t * 4000, y2);
	}
	glEnd();
}

int main(int argc, char **argv)
{
	float amplitude, angle, period, frequency, radian,
angular_velocity;
	float step = 0.0;

	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE);
	glutInitWindowSize(1200, 800);
	glutInitWindowPosition(0, 0);
	glutCreateWindow("Digital Signal Processing");

#if 0
	srand(time(NULL));

	set_rand_amplitude(&amplitude);
	set_angle_with_common_angles(&angle);
	angle2radian(&angle, &radian);
	set_rand_frequency(&frequency);
	calc_period(&frequency, &period);
	calc_angular_velocity(&frequency,
&angular_velocity);

	printf("amplitude = %f\n", amplitude);
	printf("angle = %f degree\n", angle);
	printf("radian = %f\n", radian);
	printf("frequency = %f\n", frequency);
	printf("period = %f\n", period);
	printf("angular_velocity = %f\n", angular_velocity);

	cos_sim(amplitude, angular_velocity, period);
	sin_sim(amplitude, angular_velocity, period);
#endif

	glutDisplayFunc(display);
	//glutIdleFunc(display);
	glutReshapeFunc(reshape);
	//glutKeyboardFunc(keyboard);
	glutMainLoop();

	return 0;
}
```

gcc non_anim_sin.c -lGL -lGLU -lm -lglut




```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <GL/glu.h>
#include <GL/gl.h>
#include <GL/freeglut.h>

void originAxis(void);
void sineWave(void);
void idle(void);

void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);

	originAxis();
	sineWave();

	//printf("%s\n", gluErrorString(glGetError()));
	glutSwapBuffers();
}

void sineWave(void)
{
	float wavelength = 2;
	float amplitude = 1;
	float inc = 0.05;
	float k, x, y;
	glBegin(GL_LINES);
	glColor3f(1,1,1);
	for(x=-1;x<=1;x+=inc){
		k = 2 * 3.14 / wavelength;
		y = amplitude * sin(k * x);
		glVertex3f(x, y, 0);
	}
	glEnd();

}

void idle(void)
{
	float wavelength = 2;
	float amplitude = 1;
	float inc = 0.05;
	float k, x, y;
	for (x = -1; x <= 1; x += inc){
		glBegin(GL_POINTS);
		glPointSize(200);
		glColor3f(0, 1, 0);
		k = 2 * 3.14 / wavelength;
		y = amplitude * sin(k * x);
		glVertex3f(x, y, 0);
		glEnd();
	}

	glutPostRedisplay();
}

void originAxis(void)
{
	glBegin(GL_LINES);
	glColor3f(1,0,0);
	glVertex3f(0,0,0);
	glVertex3f(1, 0, 0);
	glColor3f(0,1,0);
	glVertex3f(0,0,0);
	glVertex3f(0, 1, 0);
	glColor3f(0,0,1);
	glVertex3f(0,0,0);
	glVertex3f(0, 0, 1);
	glEnd();
}

int main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE |
GLUT_DEPTH);
	glutCreateWindow("Tutorial 2");

	glOrtho(-1, 1, -1, 1, -1, 1);
	glEnable(GL_DEPTH_TEST);

	glutDisplayFunc(display);
	glutIdleFunc(idle);
	glutMainLoop();

	return EXIT_SUCCESS;
}
```