UNIVERSITAS INDONESIA

RANCANG BANGUN APLIKASI PENGENDALI ROBOT DENGAN VISUALISASI 3D YANG DINAMIS

SKRIPSI

MUHAMMAD WAHYU
0806366163

FAKULTAS TEKNIK
PROGRAM SARJANA EKSTENSI TEKNIK ELEKTRO
DEPOK
Juni 2010

UNIVERSITAS INDONESIA

RANCANG BANGUN APLIKASI PENGENDALI ROBOT DENGAN VISUALISASI 3D YANG DINAMIS

SKRIPSI

Diajukan sebagai salah satu persyaratan menjadi sarjana teknik pada program Sarjana Teknik

MUHAMMAD WAHYU
0806366163

FAKULTAS TEKNIK
PROGRAM SARJANA EKSTENSI TEKNIK ELEKTRO
DEPOK
Juni 2010

HALAMAN PERNYATAAN ORISINALITAS

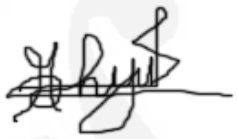
Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar

Nama : MUHAMMAD WAHYU

NPM : 0806366163

Tanda Tangan : 

Tanggal : 14 JUNI 2010

HALAMAN PENGESAHAN

TUGAS AKHIR ini diajukan oleh:

Nama	: Muhammad Wahyu
NPM	: 0806366163
Program Studi	: Elektro
Judul TUGAS AKHIR	: Rancang Bangun Aplikasi Pengendali Robot Dengan Visualisasi 3D Yang Dinamis

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Elektro, Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Dr. Abdul Muis S.T, M.Eng ( )

Penguji 1 : Ir Wahidin Wahab MSc, PhD ( )

Penguji 2 : Dr. Ir. Feri Yusivar M.Eng ( )

Ditetapkan di : Depok

Tanggal : 30 Juni 2010

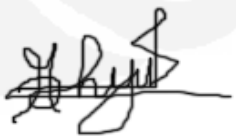
UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada ALLAH SWT, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

(1) Dr. Abdul Muis. S.T,M.Eng, selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
(2) Orang tua tercinta, kakak-kakak dan keluarga besar yang telah memberikan bantuan dukungan material dan moral;
(3) Nia Kurniasih,yang telah membantu memberikan dukungan dan doa.
(4) Teman-teman yang telah banyak membantu saya dalam menyelesaikan skripsi ini baik teman-teman di kampus maupun di forum yang tidak bisa saya sebutkan satu per satu.

Akhir kata, saya berharap ALLAH SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu

Depok, 14 Juni 2010



Muhammad Wahyu

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	:	Muhammad Wahyu
NPM	:	0806366163
Program Studi	:	Teknik Elektro
Departemen	:	Teknik Elektro
Fakultas	:	Teknik
Jenis karya	:	Skripsi

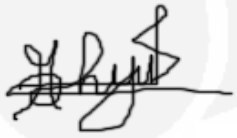
Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

Rancang Bangun Aplikasi Pengendali Robot dengan Visualisasi 3D yang Dinamis

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan skripsi saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 14 Juni 2010
Yang menyatakan



(Muhammad Wahyu)

Muhammad Wahyu
NPM : 0806366163
Departemen Teknik Elektro

Dosen Pembimbing
Dr. Abdul Muis S.T, M.Eng

RANCANG BANGUN APLIKASI PENGENDALI ROBOT DENGAN VISUALISASI 3D YANG DINAMIS

ABSTRAK

Perkembangan teknologi robotika telah membuat kualitas kehidupan manusia semakin tinggi. I*nterfacing* untuk mengontrol robot dapat menjadi suatu persoalan tersendiri, dikarenakan tidak terdapat visualiasasi dalam bentuk simulasi dan gerakan si robot, program interface pengendali robot yang beredar saat ini juga terbatas hanya untuk mengendalikan robot yang sudah dibuat sehingga tidak bisa diaplikasikan ke robot yang lain. Sistem interface yang dirancang ini akan membentuk visualisasi 3D dari robot dengan sendi-sendi yang secara dinamis bisa dikendalikan , sehingga bisa diaplikasikan ke robot mana saja dengan syarat tertentu. Sistem ini di buat dengan bahasa pemrograman Java yang memanfaatkan teknologi 3D. Dengan sistem yang dibuat ini akan bisa menampilkan simulasi gerakan dari si robot sebelum di perintahkan ke mikrokontroler yang disimulasikan rangkaiannya dengan program proteus.

Kata kunci : Robotika, Pengendali Robot, Dinamis, Java, Tiga Dimensi.

Muhamamd Wahyu
NPM : 0806366163
Electrical Engineering Department

The lecturer of consultant
Dr. Abdul Muis S.T, M.Eng

DESIGN ROBOT CONTROLLER APPLICATION WITH DYNAMIC 3D VISUALIZATION

ABSTRACT

The development of robotics technology have created a higher quality of human life. In the current robot technology, interfacing to control the robot can be a separate issue, because no visualization in the form of simulation and robot movement, currently the robot controller interface programs is limited only to control a robot that has been made so can't be applied to other robot. This interface system is designed to form a 3D visualization of the robot with dynamic joints that can be controlled, so this system can be applied to any robot with particular specification. This system develop using Java programming language that uses 3D technology. With this system will do simulation movement of robot before sending command to microcontroller that the schematic simulated using application name proteus.

Keywords : Robotics, Robot Controller, Dynamic, Java, Three Dimension.

DAFTAR ISI

DAFTAR GAMBAR

BAB 1
Pendahuluan

1.1 LATAR BELAKANG

Perkembangan teknologi robotika telah membuat kualitas kehidupan manusia semakin tinggi. Saat ini perkembangan teknologi robotika telah mampu meningkatkan kualitas maupun kuantitas produksi berbagai pabrik. Teknologi robotika juga telah menjangkau sisi hiburan dan pendidikan bagi manusia.

Pada teknologi robotika saat ini, permasalahan *interfacing* untuk mengontrol robot dapat menjadi suatu persoalan tersendiri. Interfacing ini sendiri terbatas tidak disediakan simulasi pergerakan dari robot tersebut, sehingga kita tidak tahu nanti jika diberi intruksi dari aplikasi interface yang dibuat gerakan apa yang akan robot lakukan, karena data yang dikirim langsung ke robot tersebut, tidak divisualisasikan/disimulasikan terlebih dahulu gerakan robot tersebut. Dengan aplikasi interface yang dibuat ini akan memvisualisasikan / simulasikan gerakan robot tersebut secara 3D(3 Dimensi), sehingga bisa diperkirakan gerakan robot akan seperti bagaimana, selain memvisualisasikan aplikasi ini juga bisa langsung memberikan intruksi atau data langsung ke robot saat simulasi, ataupun sesudah. Selain visualisasi keterbatasan platform OS(Operating System) yang dituju juga terbatas hanya satu platform saja, untuk itu dirancang aplikasi yang bisa jalan di semua platform, maka dari itu digunakan bahasa pemrogaman java dengan teknologi applet, karena java dengan motto "Write Once Run Everywhere", hampir semua platform bisa menjalankan aplikasi java, dengan teknologi applet maka aplikasinya nanti bisa dikembangkan menjadi aplikasi berbasis web dimana bisa menggunakan browser untuk menjalankannya, tentunya setelah JVM(Java virtual Machine) sudah terinstal.

1.2 TUJUAN

Tujuan dari pembuatan aplikasi ini adalah memberikan *interface* yang bisa *running* disemua platform menggunakan Java dan dapat digunakan untuk mengendalikan robot, tidak hanya mengendalikan tetapi juga akan menampilkan

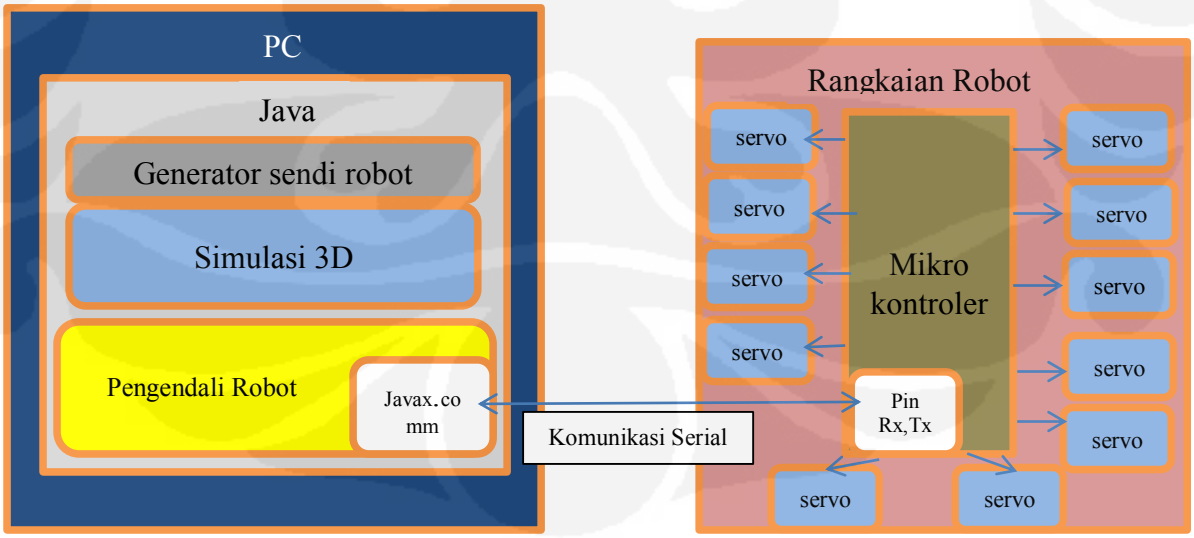
visual secara 3D. Aplikasi ini akan memvisualisasikan dan juga mengendalikan robot dengan bentuk dan jumlah sendi yang dinamis, sehingga dapat diterapkan pada robot mana saja dengan spesifikasi tertentu, robot yang digunakan akan disimulasikan rangkaiannya menggunakan program proteus.

1.3 BATASAN MASALAH

Aplikasi ini dalam penggambaran grafis 3D dari robot tidak menggambarkan secara detail hanya mengambil tampilan dari dasar model robot seperti Kubus/Balok, Silinder, Bola yang disusun atau disatukan untuk membentuk model robot tersebut. Robot dapat melakukan gerakan dengan cara sendi pada robot tersebut dilakukan rotasi dan translasi, dimana translasi ini bisa dilakukan dengan syarat hardware robot yang digunakan mendukung perintah ini. Untuk pengendalian robot disini menggunakan sistem open loop, dimana motor yang digunakan untuk dikendalikan menggunakan motor servo.

1.4 DESKRIPSI SINGKAT

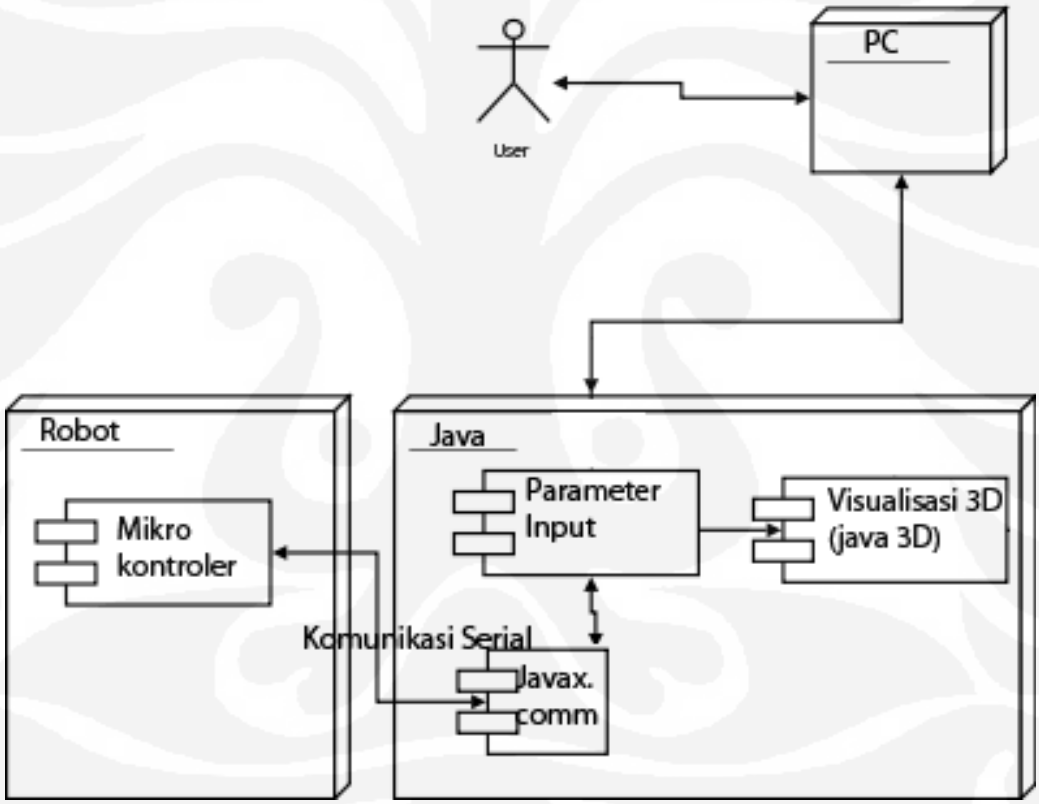
Dalam proses perancangan dan pengerjaan aplikasi/sistem ini dibuat menggunakan bahasa pemrogaman JAVA, dimana grafis 3d nya menggunakan library java3D, bagan sistemnya sebagai berikut :



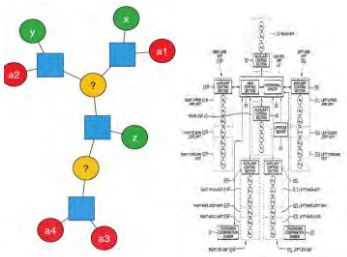
Gambar 1.1 Bagan system

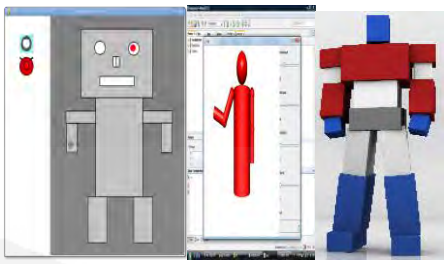
Dari gambar bagan diatas alur dari sistem ini dijelaskan singkat sebagai berikut :

1. PC yang terdapat program java, mengenerate parameter-parameter dari sendi-sendi robot dalam format file XML.
2. Program java akan membentuk visualisasi 3D dengan menggunakan library “java3D”, yang nantinya bisa disimulasikan untuk pergerakan robot secara 3D
3. Untuk mengendalikan robot menggunakan komunikasi serial yang terhubung ke mikrokontroler robot, didalam program java untuk melakukan komunikasi serial menggunakan library “javax.comm”
4. Rangkaian robot akan disimulasikan menggunakan program proteus dimana proteus tersebut terhubung dengan port serial, rangkaian robot tersebut terdapat motor servo sebagai sendi robot yang akan dikendalikan.



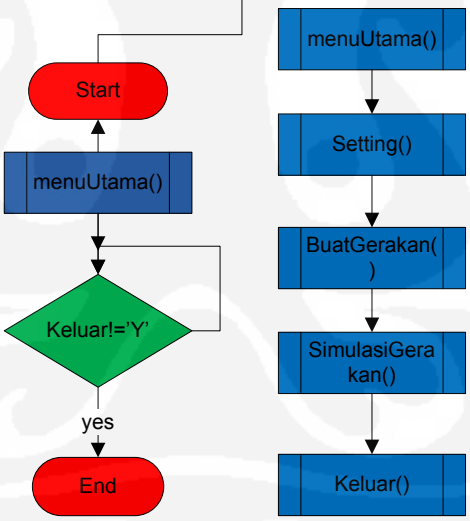
Gambar 1.2 UML System





Gambar 1.3.a Pengisian parameter Gambar 1.3.b. hasil visualisasi parameter

Dari gambar diatas dapat dijelaskan, seorang user menggunakan sebuah PC yang didalamnya terdapat aplikasi sistem yang dibuat menggunakan bahasa pemrograman java. User mengisi parameter-parameter untuk menentukan join atau sendi-sendi dari robot, lalu sistem memvisualisasikan parameter-parameter yang diinput tadi menjadi objek 3D,setelah terbentuk visualisasi 3D dari robot, user dapat mensimulasikan gerakan-gerakan robot pada sistem tersebut atau juga gerakan-gerakan bisa diteruskan oleh sistem ke mikrokontroler robot menggunakasan komunikasi serial sehingga dapat mengendalikan gerakan robot secara realtime.



Gambar 1.4 garis besar flowchart system

1.5 SISTEMATIKA PENULISAN

Dalam penulisan skripsi ini akan disusun secara sistematis yang terdiri atas bagian–bagian yang saling berhubungan sehingga diharapkan akan mudah

dipahami dan dapat diambil manfaatnya. Adapun uraian singkat tentang hal ini adalah sebagai berikut.

BAB 1 PENDAHULUAN

Pada bab ini berisi tentang latar belakang, tujuan penelitian, pembatasan masalah, metode penelitian dan sistematika penulisan.

BAB 2 TEORI DASAR

Pada Bab ini berisi tentang pengertian JAVA, apa itu JAVA Programming, apa itu JAVA3D, XML, Komunikasi serial, Microcontroller, Motor Servo

BAB 3 PERANCANGAN DAN CARA KERJA SISTEM

Merupakan penjelasan pembuatan rancangan sistem pengendali Robot menggunakan JAVA dengan simulasi/visualisasi 3D yang dinamis, system pengendali motor servo dengan mikrokontroler, system komunikasi serial antara java dengan mikrokontroler.

BAB 4 PENGUJIAN SISTEM

Sistem yang telah dirancang kemudian diuji dengan parameter-parameter yang terkait. Pengujian ini meliputi pengujian dalam menemukan error, kemungkinan ada *bug*. Dari hasil pengujian ini dapat dilakukan analisa terhadap kerja sistem, sehingga dapat diketahui apa yang menjadi penyebab dari error atau kesalahan dan juga kemungkinan penutupan bug(patch) bila selama pengujian ditemui hal-hal tersebut.

BAB 5 KESIMPULAN

Pada Bab ini berisi tentang kesimpulan dari hasil pengujian yang telah dilakukan yang berkaitan dengan sistem yang dibuat dan mengidentifikasi error ataupun kekurangan sistem ini.

Bab 2
Dasar Teori

Pengertian Robot secara umum adalah suatu perangkat keras berupa sistem mekanik dan elektrik yang dikendalikan oleh perangkat lunak yang dibuat oleh manusia untuk membantu tugas-tugas manusia itu sendiri. Sekarang pengertian Robot itu sendiri sudah berkembang dan menjadi lebih luas lagi. Pembentukan sendi-sendi robot pada umumnya sering menggunakan motor servo.

2.1 Motor Servo

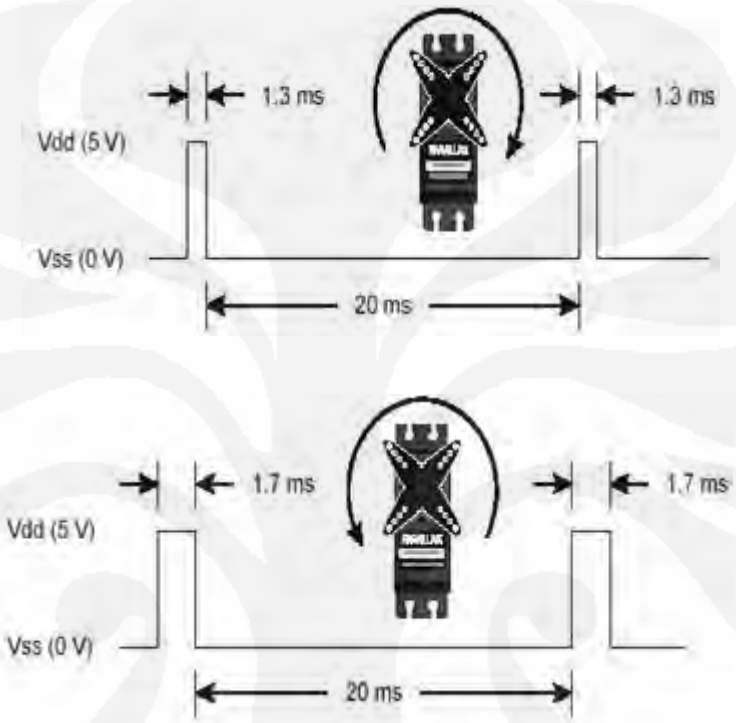
Motor servo adalah sebuah motor dengan sistem umpan balik tertutup di mana posisi dari motor akan diinformasikan kembali ke rangkaian kontrol yang ada di dalam motor servo. Motor ini terdiri dari sebuah motor DC, serangkaian gear, potensiometer dan rangkaian kontrol. Potensiometer berfungsi untuk menentukan batas sudut dari putaran servo. Sedangkan sudut dari sumbu motor servo diatur berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor. Karena motor DC servo merupakan alat untuk mengubah energi listrik menjadi energi mekanik, maka magnit permanent motor DC servolah yang mengubah energi listrik ke dalam energi mekanik melalui interaksi dari dua medan magnit. Salah satu medan dihasilkan oleh magnit permanent dan yang satunya dihasilkan oleh arus yang mengalir dalam kumparan motor. Resultan dari dua medan magnit tersebut menghasilkan torsi yang membangkitkan putaran motor tersebut. Saat motor berputar, arus pada kumparan motor menghasilkan torsi yang nilainya konstan.

- Jenis Motor Servo

  Secara umum terdapat 2 jenis motor servo. Yaitu motor servo standard dan motor servo Continous. Motor servo standard sering dipakai pada sistim robotika misalnya untuk membuat " Robot Arm" (Robot Lengan) sedangkan motor servo Continous sering dipakai untuk Mobile Robot. Pada badan servo tertulis tipe servo yang bersangkutan.

Untuk menggerakkan motor servo ke kanan atau ke kiri, tergantung dari nilai *positive pulse width* yang kita berikan. Untuk membuat servo pada posisi center, berikan pulsa 1.5ms. Untuk memutar servo ke kanan, berikan pulsa <1.5ms, dan pulsa > 1.5ms untuk berputar ke kiri dengan frekunsi 5Hz, seperti ilustrasi berikut:

.



Gambar.2.1 N*ilai pulsa untuk menggerakkan motor servo*

Robot yang sudah jadi diperlukan sebuah otak dimana nantinya akan mengendalikan robot tersebut, sehingga robot dapat melakukan gerakan ataupun berinteraksi. Otak ini sendiri biasanya berupa mikrokontroler yang didalamnya dimasukkan program-program, dimana nantinya program ini akan diterjemahkan oleh mikrokontroller sebagai set perintah ke robot atau sendi-sendi robot tersebut.

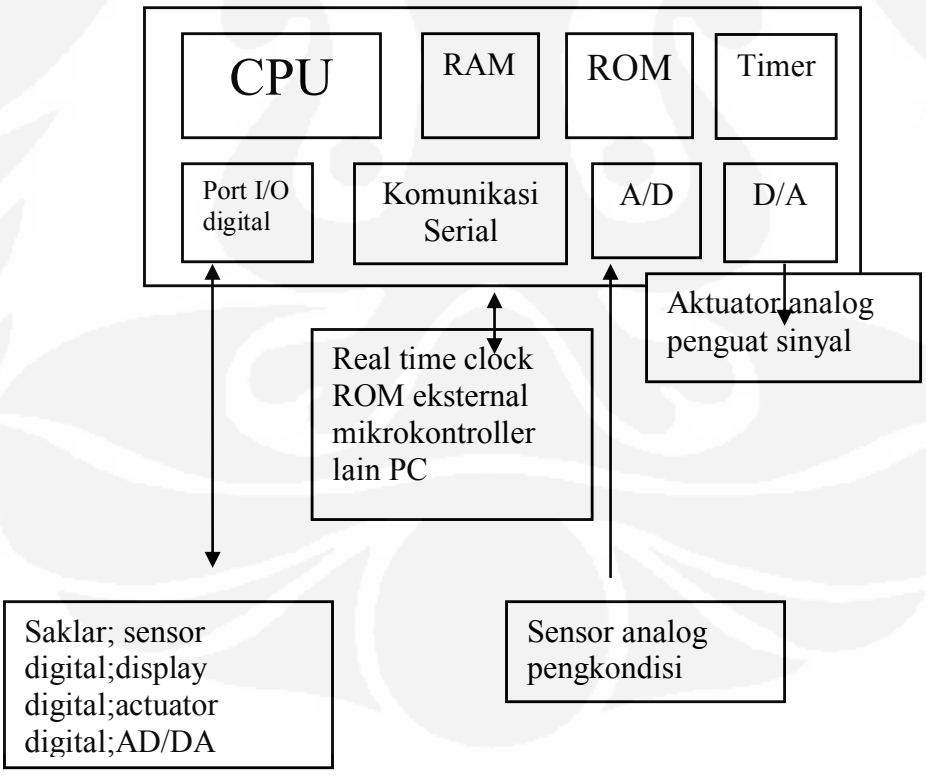
2.2 Mikrokontroler

Mikrokontroler pada dasarnya adalah komputer dalam satu chip, yang didalamnya terdapat mikroprosesor, memori, jalur input/output (I/O) dan perangkat pelengkap lainnya. Kecepatan pengolahan data pada mikrokontroler

lebih rendah jika dibandingkan den PC. Pada PC kecepatan mikroprosesor yang digunakan saat ini telah mencapai orde GHz, sedangkan kecepatan operasi mikrokontroler pada umumnya berkisar antara 1-16 MHz. begitu juga kapasitsas RAM dan ROM pada PC yang bias mencapai orde Gbyte, dibandingkan dengan mikrokontroler yang hanya berkisar pada orde byte/Kbyte.

Meskipun kecepatan pengolahan data dan kapasitas memori pada mikrokontroler jauh lebih kecil jika dibandingkan dengan komputer personal, namun kemampuan mikrokontroler sudah cukup untuk dapat digunakan pada banyak aplikasi terutama karena ukurannya yang kompak. Mikrokontroller sering digunakan pada system yang tidak terlalu kompleks dan tidak memerlukan kemampuan komputasi yang tinggi.

System yang menggunakan mikrokontroller sering disebut sebagai *embedded system* atau dedicated system. Embedded system adalah system pengendali yang tertanam pada suatu produk, sedangkan dedicated system adalah system pengendali yang dimaksudkan hanya untuk suatu fungsi tertentu.



Gambar 2.2 Komponen mikrokontroller

Gambar 2.2 menunjukkan kompnen-komponen dari suatu mikrokontroler yang mempunyai fasilitas lengkap beserta piranti eksternal yang biasanya dihubungkan ke/dari mikrokontroler. Tidak semua mikrokontroler mempunyai semua komponen tersebut, misalnya converter A/D dan D/A hanya terdapat pada beberapa jenis mikrokontroler tertentu.

Untuk dapat berkomunikasi antara PC dengan mikrokontroler, diperlukan sebuah aplikasi yang dibuat di PC, dimana aplikasi tersebut bisa mengirim dan menerima data ke mikrokontroler. Aplikasi ini bisa dibuat dengan bahasa pemrograman apa saja. Bahasa pemrograman yang sedang berkembang saat ini dan juga mempunyai keunggulan bisa running di semua platform, adalah bahasa pemrograman java.

2.3 Pemrogaman Java

Bahasa pemrograman Java pertama lahir dari The Green Project, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas1992[1]. Proyek tersebut belum menggunakan versi yang dinamakan Oak[1]. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta sembilan pemrogram lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot *Duke* yang dibuat oleh Joe Palrang[1].

Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "bapak java", James Gosling[1]. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java"[1]. Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling[1].

2.3.1 Applet

Java applet adalah sebuah program kecil yang ditulis dengan menggunakan bahasa Pemrograman Java, yang diakses melalui halaman Web dan dapat di-*download* ke dalam mesin klien yang kemudian menjalankannya di dalam *web browser*. *Java applet* dapat secara dinamis menambahkan beberapa fungsi kepada halaman-halaman Web yang bersifat statis. Akan tetapi, untuk

menjalankannya sebuah komputer harus memiliki program *web browser* yang dapat menjalankan Java, seperti Microsoft Internet Explorer 4.0 ke atas, Netscape Navigator, Mozilla Firefox, Chrome, Safari dan Opera.

Ketika sebuah *Java applet* dibuat, semua pernyataan Java yang terkandung di dalam kode sumbernya akan dikompilasi menjadi ***Java bytecode***, yakni sebuah bahasa mesin semu (*virtual engine/machine language*) yang dibentuk oleh Java. Sebuah halaman Web yang hendak menggunakan *applet* tersebut harus menggunakan tag<APPLET>...</APPLET> di dalam kode sumber-nya. Ketika sebuah penjelajah Web milik klien melakukan *request* kepada halaman Web tersebut dan menemukan bahwa di dalamnya terdapat tag <APPLET>...</APPLET>, bytecode di dalam Java class file akan dieksekusi oleh mesin semu di dalam jendela penjelajah Web, yang dapat berupa Microsoft Java Virtual Machine atau Java Runtime Engine dari Sun Microsystems[2].

Bahkan untuk memulai eksekusi pada method *main* seperti dalam aplikasi khas Java, browser atau applet viewer berhubungan dengan applet melalui method-method berikut:

1. *init()*

   init adalah method yang dipanggil pertama kali. Yang sebenarnya berisi permintaan pertama ketika applet di load.

2. *start()*

   Setelah meminta method *init*, mulai dengan method yang dipangil selanjutnya. method ini meminta dokumen HTML yang ditampilkan applet setiap waktu. Eksekusi ringkasan dengan method ini dilakukan ketika applet ditampilkan kembali.

3. *stop()*

   Ketika web browser meninggalkan dokumen HTML applet,method ini dipanggil untuk menginformasikan applet bahwa dia harus menghentikan proses eksekusinya.

4. *destroy()*

   Method ini dipanggil ketika applet perlu dihapus dari memory. Method *stop* selalu dipanggil sebelum method ini diminta untuk dijalankan.

2.3.2 Visualisasi 3D

Dalam pembuatan program pengendali robot, ada baiknya program tersebut terdapat simulasi ataupun tampilan secara tiga dimensi dari robot yang dikendalikan, sehingga terlihat nantinya gerakan atau interaksi secara apa yang akan terjadi pada robot secara visual 3D (tiga dimensi) jika perintah program dijalankan.

2.3.2.1 OpenGL

OpenGL adalah *environment* utama untuk mengembangkan portabel, interaktif 2D dan aplikasi grafis 3D. Sejak diperkenalkan pada tahun 1992, OpenGL telah menjadi yang paling banyak digunakan dan didukung 2D dan 3D industri pemrograman grafis antarmuka aplikasi (API), membawa ribuan aplikasi ke berbagai platform computer. OpenGL menumbuhkan inovasi dan pengembangan aplikasi dengan menggabungkan kecepatan yang luas rendering, pemetaan tekstur, efek khusus, dan lain fungsi visualisasi yang kuat. Pengembang dapat memanfaatkan kekuatan OpenGL populer di seluruh desktop dan platform workstation, memastikan penyebaran aplikasi yang luas.

Operasi dasar OpenGL adalah untuk menerima bentuk dasar seperti titik, garis dan poligon, dan mengkonversikannya ke piksel. Hal ini dilakukan dengan grafik pipeline dikenal sebagai OpenGL state. OpenGL adalah low-level, API prosedural, yang memerlukan programmer untuk menentukan langkah-langkah yang tepat diperlukan untuk membuat scene. Hal ini bertentangan dengan deskriptif, di mana seorang programmer hanya perlu untuk menggambarkan adegan/scene dan dapat membiarkan library mengatur renderingnya. desain OpenGL tingkat rendah membutuhkan programmer untuk memiliki pengetahuan yang baik dari grafis pipeline, tapi juga memberikan sejumlah kebebasan untuk mengimplementasikan algoritma rendering.

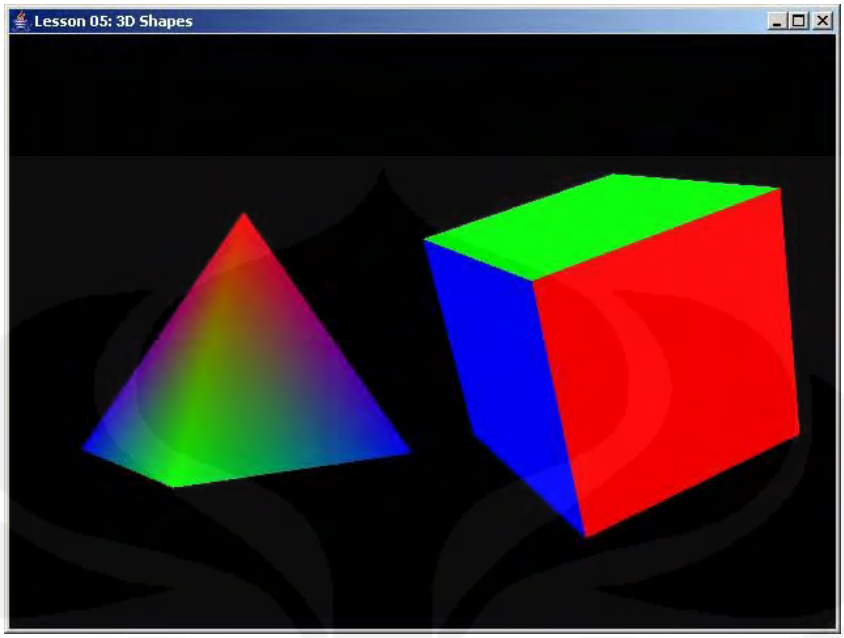
Didalam bahasa pemrograman terdapat library untuk membuat aplikasi OpenGL yaitu bernama “JOGL(Java OpenGL)” untuk membuat aplikasi menggunakan “JOGL” aplikasi tersebut harus menurunkan fungsi-fungsi yang terdapat pada JOGL yaitu dengan script “import javax.media.opengl” berikut contoh script menggunakan JOGL :

```
package org.RiderSystem;
import com.sun.opengl.util.Animator;
```

```
import java.awt.Frame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.media.opengl.GL;
import javax.media.opengl.GLAutoDrawable;
import javax.media.opengl.GLCanvas;
import javax.media.opengl.GLEventListener;
import javax.media.opengl.glu.GLU;

/**
 * ShirouSimpleJOGL.java <BR>
 *
 *
 *
 */
public class ShirouSimpleJOGL implements GLEventListener {
    public static void main(String[] args) {
    Frame frame = new Frame("Simple JOGL Application");
    GLCanvas canvas = new GLCanvas();

     canvas.addGLEventListener(new ShirouSimpleJOGL());
     frame.add(canvas);
     frame.setSize(640, 480);
     final Animator animator = new Animator(canvas);
     frame.addWindowListener(new WindowAdapter() {

     @Override
      public void windowClosing(WindowEvent e) {
       // Run this on another thread than the AWT event queue to
       // make sure the call to Animator.stop() completes before
        // exiting
        new Thread(new Runnable() {

        public void run() {
                animator.stop();
                System.exit(0);
              }
          }).start();
         }
    });
       // Center frame
     frame.setLocationRelativeTo(null);
     frame.setVisible(true);
     animator.start();
    }

    public void init(GLAutoDrawable drawable) {
       // Use debug pipeline
       // drawable.setGL(new DebugGL(drawable.getGL()));

       GL gl = drawable.getGL();
       System.err.println("INIT GL IS: " + gl.getClass().getName());

       // Enable VSync
       gl.setSwapInterval(1);

       // Setup the drawing area and shading mode
       gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
       gl.glShadeModel(GL.GL_SMOOTH); // try setting this to GL_FLAT and see what happens.
    }
    public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {
       GL gl = drawable.getGL();
       GLU glu = new GLU();
```

```
        if (height <= 0) { // avoid a divide by zero error!

            height = 1;
        }
        final float h = (float) width / (float) height;
        gl.glViewport(0, 0, width, height);
        gl.glMatrixMode(GL.GL_PROJECTION);
        gl.glLoadIdentity();
        glu.gluPerspective(45.0f, h, 1.0, 20.0);
        gl.glMatrixMode(GL.GL_MODELVIEW);
        gl.glLoadIdentity();
    }

    public void display(GLAutoDrawable drawable) {
        GL gl = drawable.getGL();

        // Clear the drawing area
        gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
        // Reset the current matrix to the "identity"
        gl.glLoadIdentity();

        // Move the "drawing cursor" around
        gl.glTranslatef(-1.5f, 0.0f, -6.0f);

        // Drawing Using Triangles
        gl.glBegin(GL.GL_TRIANGLES);
            gl.glColor3f(1.0f, 0.0f, 0.0f);    // Set the current drawing color to red
            gl.glVertex3f(0.0f, 1.0f, 0.0f);   // Top
            gl.glColor3f(0.0f, 1.0f, 0.0f);    // Set the current drawing color to green
            gl.glVertex3f(-1.0f, -1.0f, 0.0f); // Bottom Left
            gl.glColor3f(0.0f, 0.0f, 1.0f);    // Set the current drawing color to blue
            gl.glVertex3f(1.0f, -1.0f, 0.0f);  // Bottom Right
        // Finished Drawing The Triangle
        gl.glEnd();

        // Move the "drawing cursor" to another position
        gl.glTranslatef(3.0f, 0.0f, 0.0f);
        // Draw A Quad
        gl.glBegin(GL.GL_QUADS);
            gl.glColor3f(0.5f, 0.5f, 1.0f);    // Set the current drawing color to light blue
            gl.glVertex3f(-1.0f, 1.0f, 0.0f);  // Top Left
            gl.glVertex3f(1.0f, 1.0f, 0.0f);   // Top Right
            gl.glVertex3f(1.0f, -1.0f, 0.0f);  // Bottom Right
            gl.glVertex3f(-1.0f, -1.0f, 0.0f); // Bottom Left
        // Done Drawing The Quad
        gl.glEnd();

        // Flush all drawing operations to the graphics card
        gl.glFlush();
    }

    public void displayChanged(GLAutoDrawable drawable, boolean modeChanged, boolean deviceChanged)
{
    }
}
```

Pada contoh script diatas akan menampilkan visual 3D “JOGL” seperti pada gambar berikut :



Gambar 2.3 Visual 3D dengan JOGL

Selain dengan OpenGL java menyediakan library 3D yaitu java3D.

2.3.2.2 Java 3D

Java 3D API memungkinkan penciptaan tiga-dimensi aplikasi grafis dan 3D berbasis internet applet. Menyediakan konstruksi tingkat tinggi untuk menciptakan dan manipulasi 3D geometri dan membangun struktur yang digunakan dalam rendering geometri. Dengan software ini, kita dapat secara efisien menentukan dan membuat dunia maya sangat besar[3].

Java 3D menggunakan konvensi pemrograman berikut:

- Sistem koordinat default adalah tangan kanan, dengan + *y* vertikal keatas, + *x* horizontal ke kanan, dan + *z* diarahkan kedepan.
- Semua sudut atau representasi rotasi dalam radian.
- Semua jarak dinyatakan dalam satuan meter.

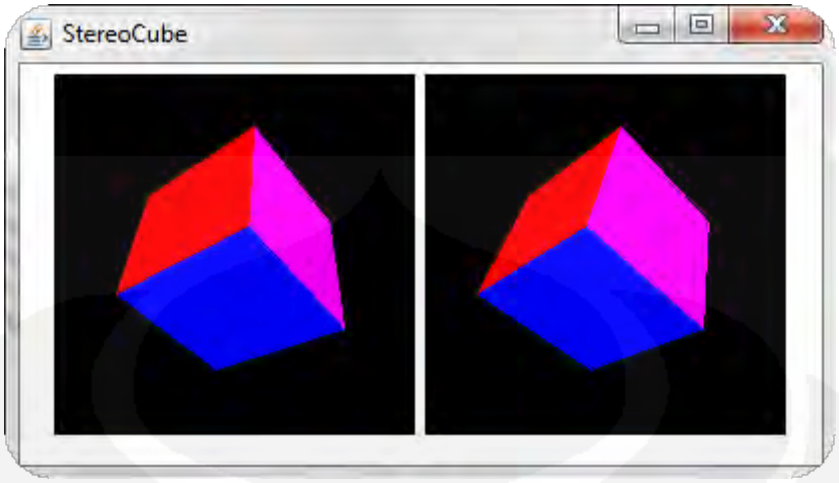
Dalam membuat aplikasi menggunakan Java3D aplikasi java tersebut harus menurunkan fungsi-fungsi dari library Java3D yaitu dengan script "import javax.media.j3d" berikut contoh script menggunakan Java3D :

```
/*
 * To change this template, choose Tools | Templates
```

```
 * and open the template in the editor.
 *//** *
 * @author shirou
 */
import java.applet.Applet;
import java.awt.FlowLayout;
import java.awt.GraphicsConfiguration;
import javax.media.j3d.BoundingSphere;
import javax.media.j3d.BranchGroup;
import javax.media.j3d.Canvas3D;
import javax.media.j3d.PhysicalBody;
import javax.media.j3d.Transform3D;
import javax.media.j3d.TransformGroup;
import javax.media.j3d.View;
import javax.vecmath.Point3d;

import com.sun.j3d.utils.applet.MainFrame;
import com.sun.j3d.utils.behaviors.keyboard.KeyNavigatorBehavior;
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;
import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;
import com.sun.j3d.utils.geometry.ColorCube;
import com.sun.j3d.utils.universe.SimpleUniverse;

public class StereoCube extends Applet {
    Canvas3D c1 = new Canvas3D(SimpleUniverse.getPreferredConfiguration());
    Canvas3D c2 = new Canvas3D(SimpleUniverse.getPreferredConfiguration());
    static MainFrame mf;
    private SimpleUniverse u = null;
    private BranchGroup scene = null;
    public void init() {
        setLayout(new FlowLayout());
        GraphicsConfiguration config =
          SimpleUniverse.getPreferredConfiguration();
        c1.setSize(180, 180);
        c1.setMonoscopicViewPolicy(View.LEFT_EYE_VIEW);
        add(c1);
        c2.setSize(180, 180);
        c2.setMonoscopicViewPolicy(View.RIGHT_EYE_VIEW);
        add(c2);

        // Create a simple scene and attach it to the virtual universe
        scene = createSceneGraph(0);
        u = new SimpleUniverse(c1);

        View view0 = u.getViewer().getView();
        View view = new View();
        PhysicalBody myBod = view0.getPhysicalBody();
        myBod.setLeftEyePosition(new Point3d(-.006,0.0, 0.0)); // default is(-0.033, 0.0, 0.0)
        myBod.setRightEyePosition(new Point3d(+.006,0.0, 0.0));
        view.setPhysicalBody(myBod);
        view.setPhysicalEnvironment(view0.getPhysicalEnvironment());
        view.attachViewPlatform(u.getViewingPlatform().getViewPlatform());
        view.addCanvas3D(c2);

        // This will move the ViewPlatform back a bit so the
        // objects in the scene can be viewed.
        u.getViewingPlatform().setNominalViewingTransform();
        u.addBranchGraph(scene);

    }
    public BranchGroup createSceneGraph(int i) {
        // Create the root of the branch graph
        BranchGroup objRoot = new BranchGroup();
        TransformGroup objTrans = new TransformGroup();
```

```
      objTrans.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
      Transform3D t = new Transform3D();
    TransformGroup tg = new TransformGroup(t);
    tg.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
    tg.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
    objTrans.addChild(tg);
      tg.addChild(new ColorCube(0.4));
      MouseRotate behavior = new MouseRotate();
    BoundingSphere bounds =
        new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);
    behavior.setTransformGroup(tg);
    objTrans.addChild(behavior);
      // Create the translate behavior node
    MouseTranslate behavior3 = new MouseTranslate();
    behavior3.setTransformGroup(tg);
    objTrans.addChild(behavior3);
    behavior3.setSchedulingBounds(bounds);
    KeyNavigatorBehavior keyNavBeh = new KeyNavigatorBehavior(tg);
  keyNavBeh.setSchedulingBounds(new BoundingSphere(
    new Point3d(),1000.0));
  objTrans.addChild(keyNavBeh);
    behavior.setSchedulingBounds(bounds);
      objRoot.addChild(objTrans);
      return objRoot;
  }
  public StereoCube() {
  }
  public void destroy() {
      u.removeAllLocales();
  }
  public void setSize(int width, int height) {
      System.out.println("setsize " + width +"," +height);
      super.setSize(width, height);
      int minDimension = Math.min(width/2, height);
      c1.setSize((minDimension - 20),(minDimension - 20));
      c2.setSize((minDimension - 20),(minDimension - 20));
      if (mf != null) {
          mf.appletResize(width, height);
      }
      validate();
  }
  public static void main(String[] args) {
      mf = new MainFrame(new StereoCube(), 400, 200);
  }
}
```

Pada contoh script diatas akan tampil visual 3D sebagai berikut :



Gambar 2.4 Visual 3D dengan Java3D

OpenGL vs Java3D

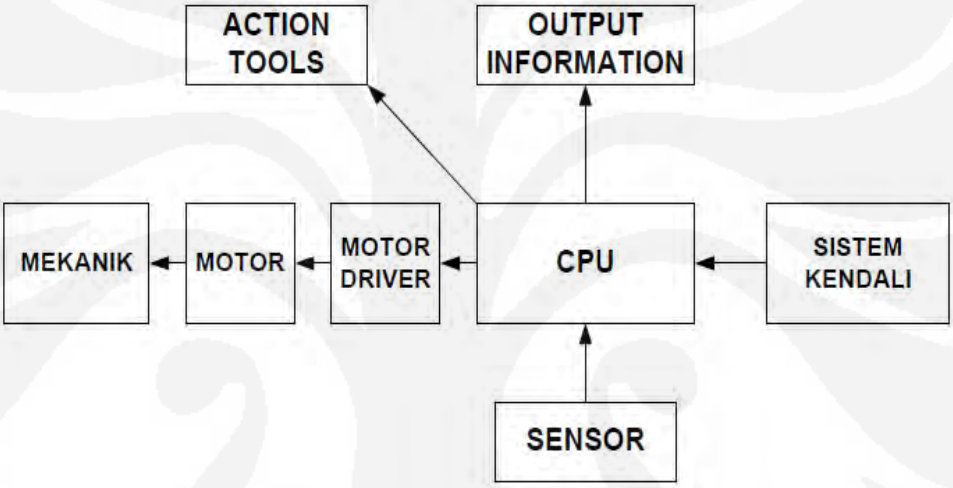
OpenGL	Java 3D
Rumit dipelajari bagi programmer pemula	Mudah dipelajari bagi programmer pemula
Performance rendering bagus	Ada sedikit flickering saat melakukan render
Low level hardware control	High level content control
Detail control	Fokus di konten

Visualisasi secara tiga dimensi banyak dimanfaatkan di jaman sekarang ini, salah satu keunggulannya dengan tiga dimensi bisa didapatkan simulasi ataupun visual dari objek yang akan dibuat atau dirancang sebelum nantinya objek tersebut direalisasikan. Dalam dunia robotic, visualisasi secara tiga dimensi bisa juga dimanfaatkan sebagai pembuatan model tiga dimensi dari si robot dan juga mensimulasikannya.

2.4 Robot Modelling

Pengertian Robot secara umum adalah suatu perangkat keras berupa sistem mekanik dan elektrik yang dikendalikan oleh perangkat lunak yang dibuat oleh manusia untuk membantu tugas-tugas manusia itu sendiri. Bahkan pengertian

Robot itu sendiri sekarang sudah berkembang dan menjadi lebih luas lagi. Dalam perkembangannya, Robot kini telah dikenal oleh semua orang dan dapat dibuat oleh semua orang dalam berbagai golongan baik itu orang profesional maupun orang yang masih awam. Karena dalam proses untuk membuat suatu Robot sangat mudah dan dapat dipelajari oleh semua orang, bahkan anak-anak sekarang sudah bisa membuat robot karya mereka sendiri. Bahkan saat ini Robot sudah masuk di lingkungan sekolah-sekolah dan banyak sekali event-event perlombaan yang diikuti oleh sekolah-sekolah dari SD hingga SMA dan Universitas-universitas yang ada di Indonesia.



Gambar.2.5 Bagian-bagian Robot

Gambar di atas menunjukkan bagian-bagian robot secara garis besar. Tidak seluruh bagian ada pada setiap robot, hal ini dibedakan berdasarkan fungsinya saja. Contohnya, sistem kendali hanya digunakan pada robot yang kategori teleoperated saja.

Berdasarkan bentuk, robot terdiri dari kategori:

- **Turtle**

  Diciptakan tahun 1970 an dan nama Turtle diambil dari bentuknya yang mirip rumah kura-kura

- **Vehicle**

  Robot jenis ini berbentuk seperti kendaraan yang dilengkapi dengan roda dan bergerak seperti sebuah mobil. Perbedaan dengan mobil adalah kemampuan programmablenya

- **Rover**

  Bentuk robot ini cenderung pendek dan juga dilengkapi roda seperti jenis vehicle seperti pada R2-D2 dalam film Star Wars. Robot jenis ini juga dilengkapi beberapa fungsi contohnya kemampuan untuk mendeteksi api atau mendeteksi obyek.

- **Walker**

  Robot jenis ini tidak dilengkapi dengan roda seperti jenis vehicle dan rover melainkan bergerak dengan menggunakan kaki. Biasanya robot ini berbentuk mirip serangga dan dilengkapi dengan 6 kaki.

- **Appendage**

  Robot ini berupa lengan yang biasanya digunakan untuk mengambil dan memindahkan barang. Lengan ini dapat terpasang pada robot yang bergerak atau pada sebuah tempat yang statis.

- **Android**

  Robot ini didisain menyerupai manusia dan mempunyai kemampuan untuk berkomunikasi dengan manusia.

Sedangkan berdasarkan proses kendalinya robot terdiri dari:

- **Automatic Robot**

  Automatic Robot bergerak berdasarkan perintah-perintah yang telah diprogramkan sebelumnya atau berdasarkan masukan dari sensor-sensornya

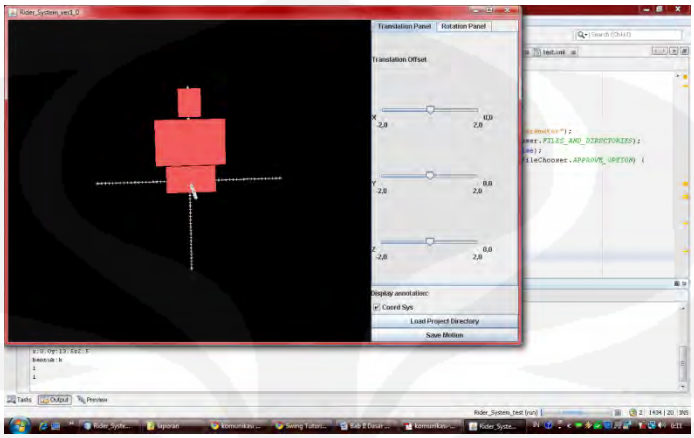
- **Teleoperated**

  Robot jenis ini bergerak berdasarkan perintah-perintah yang dikirimkan secara manual baik melalui remote control, PC atau joystick.

Robot modelling ini dimaksud perancangan model robot dimana bentuk robot nanti akan tersusun, biasanya bentuk tubuh robot tersusun dari bentuk dasar bidang seperti kubus,silinder,balok, bola dan lain-lain.

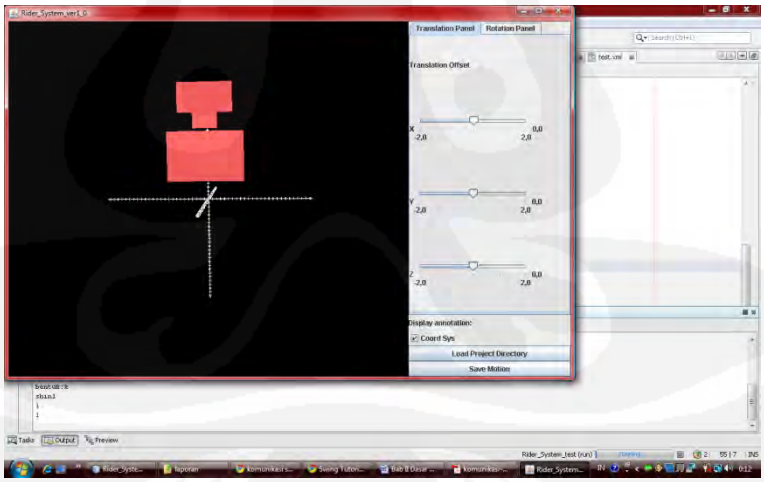
Sebuah robot dirancang dan dibangun dengan menyusun bentuk dasar bidang ruang tersebut dan menggabungkannya. Sebagai contoh sebuah robot dibangun dari kerangka badan misal badan lalu ditambahkan join pundak yang dimana letaknya(koordinat x,y,z) akan dimulai dari titik nol dari badan tersebut

lalu ditambah dengan join tangan jika tangan akan ditempel pada lengan maka letak koordinatnya dari titik nol dari lengan, namun jika ditempelkan ke badan maka koordinat titik letaknya dimulai dari titik nol dari badan, begitu seterusnya sehingga terbentuk robot yang diinginkan



Gambar 2.6. Semua bidang dibentuk menempel dAri badan

Pada gambar 2.4 diatas terlihat bahwa model 3D tersebut dibentukdari 3 kotak 1 kepala(join kepala),1 badan, 1 pinggul(join pinggul), dimana dibangun badan robot terlebih dahulu kemudian ditempelkan join-joinnya ke badan sehingga pengukuran letaknya diukur dari titik nol dari badan model tersebut.



Gambar 2.7. Bidang dibentuk menempel di badan dan ada yang di join lain

Pada gambar 2.6 diatas terlihat bahwa model tersebut dimulai dengan membangun badan dari model dan kemudian ditambahkan join-1 ke badan lalu juga menambah join-2 ke join-1, terlihat perbedaan pada letaknya pada saat menempelkan join-1 pada badan maka ukuran letaknya dihitung dari titik nol dari badan tersebut, akan tetapi saat menempelkan join-2 ditempel pada join1 ukuran

letak koordinatnya dimulai dari titik nol join-1 (gambar 2.6 dan gambar 2.7 menggunakan isian parameter-parameter yang sama, hanya berbeda pada join-2 diubah menempel pada join-1).

Pada pembentukan model robot membutuhkan parameter-parameter, dimana parameter-parameter tersebut akan dipanggil program untuk dibentuk menjadi model 3D dari robot, parameter-prameter tersebut akan disimpan dengan format tertentu.

2.5 XML

Format penyimpanan parameter-parameter untuk model robot dipilih menggunakan format file XML, Penggunaan XML dikarenakan mudah dimengerti struktur elemennya (karena menggunakan tag sesuai keinginan kita sendiri) begitu juga dengan scriptnya

2.5.1 Pengertian XML

XML kependekan dari eXtensible Markup Language, dikembangkan mulai tahun 1996 dan mendapatkan pengakuan dari W3C pada bulan Februari 1998. Teknologi yang digunakan pada XML sebenarnya bukan teknologi baru, tapi merupakan turunan dari SGML yang telah dikembangkan pada awal 80-an dan telah banyak digunakan pada dokumentasi teknis proyek-proyek berskala besar. Ketika HTML dikembangkan pada tahun 1990, para penggagas XML mengadopsi bagian paling penting pada SGML dan dengan berpedoman pada pengembangan HTML menghasilkan markup language yang tidak kalah hebatnya dengan SGML.

Seperti halnya HTML, XML juga menggunakan elemen yang ditandai dengan tag pembuka (diawali dengan „<’ dan diakhiri dengan „>’), tag penutup(diawali dengan „</ „diakhiri „>’) dan atribut elemen(parameter yang dinyatakan dalam tag pembuka misal <form name=”isidata”>). Hanya bedanya, HTML medefinisikan dari awal tag dan atribut yang dipakai didalamnya, sedangkan pada XML kita bisa menggunakan tag dan atribut sesuai kehendak kita. Untuk lebih jelasnya lihat contah dibawah:

```
<pesan>
<dari>Dekan</dari>
```

```
<buat>Ketua Jurusan</buat>
<buat>Dosen</buat>
<subyek>Keputusan SK</subyek>
<isi>Mohon segera diberlakukan SK Rektor</isi>
</pesan>
```

pada contoh diatas <pesan>, <dari> <buat>,dan <isi> bukanlah tag standard yang telah di tetapkan dalam XML. Tag-tag itu kita buat sendiri sesuai keinginan kita. Sampai di sini XML tidak melakukan apapun. Yang ada hanyalah informasi yang di kemas dengan tag-tag XML. Kita harus membuat software lagi untuk untuk mengirim, menerima atau menampilkan informasi di dalamnya.

2.5.2 Bagian-Bagian dari Dokumen XML

Sebuah dokumen XML terdiri dari bagian bagian yang disebut dengan node. Node-node itu adalah:

- **Root node** yaitu node yang melingkupi keseluruhan dokumen. Dalam satu dokumen XML hanya ada satu root node. Node-node yang lainnya berada di dalam root node.
- **Element node** yaitu bagian dari dokumen XML yang ditandai dengan tag pembuka dan tag penutup, atau bisa juga sebuah tag tunggal elemen kosong seperti <anggota nama=”budi”/> . Root node biasa juga disebut root element
- **Attribute note** termasuk nama dan nilai atribut ditulis pada tag awal sebuah elemen atau pada tag tunggal.
- **Text node**, adalah text yang merupakan isi dari sebuah elemen, ditulis diantara tag pembuka dan tag penutup
- **Comment node** adalah baris yang tidak dieksekusi oleh parser
- **Processing Instruction node**, adalah perintah pengolahan dalam dokumen XML. Node ini ditandai awali dengan karakter <? Dan diakhiri dengan ?>. Tapi perlu diingat bahwa header standard XML <?xml version=”1.0” encoding=”iso-8859-1”?> bukanlah processing instruction node. Header standard bukanlah bagian dari hirarki tree dokumen XML.

- **NameSpace Node**, node ini mewakili deklarasi namespace

2.5.3 Sintaks XML

Dibandingkan dengan HTML, XML lebih cerewet. Kalau kita menulis sebuah dokumen HTML, beberapa kesalahan penulisan masih ditolerir. Misalnya kita menempatkan tag bersilangan seperti Huruf Tebal meskipun tidak dianjurkan, HTML masih bisa bekerja dan menampilkan hasil seperti yang diinginkan. Tidak demikian dengan XML. Setiap membuat dokumen XML diawali dengan heading standard XML. Formatnya adalah sebagai berikut**:**

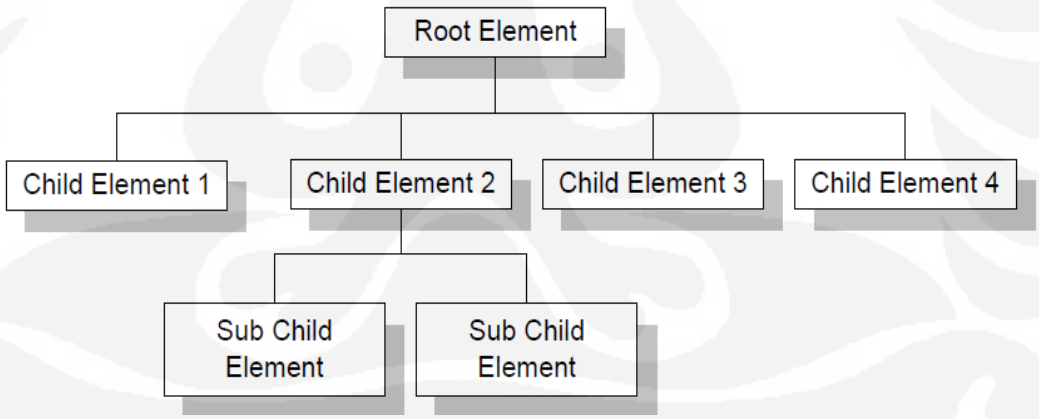
<?xml version=”1.0” encoding=”iso-8859-1”?>

Dokumen XML harus memiliki Root tag

Sebuah dokumen XML yang baik harus memiliki root tag. Yaitu tag yang melingkupi keseluruhan dari dokumen. Tag-tag yang lain, disebut child tag, berada didalam root membentuk hirarki seperti gambar

Contoh:

```
<root>
        <child>
                <subchild></subchild>
        </child>
</root>
```



Gambar 2.8 diagram hierarki XML

Untuk dapat melakukan komunikasi antara PC dan mikrokontoler di bisa melakukan dengan dua cara, yaitu komunikasi dengan port parallel dan juga komunikasi dengan port serial.

2.6. Komunikasi Serial

Komunikasi serial ialah pengiriman data secara serial (data dikirim satu persatu secara berurutan), sehingga komunikasi serial jauh lebih lambat daripada komunikasi paralel. Serial port lebih sulit ditangani karena peralatan yang dihubungkan ke serial port harus berkomunikasi dengan menggunakan transmisi serial, sedang data di komputer diolah secara parallel. Oleh karena itu data dari dan ke serial port harus dikonversikan ke dan dari bentuk paralel untuk bisa digunakan

Berdasarkan pengiriman dan penerimaan data, transmisi data dapat dibagi menjadi :

- Transmisi simplex, dimaan komunikasi dilakukan satu arah, satu piranti berfungsi sebagai pengirim dan piranti lainnya sebagai penerima.
- Transmisi duplex, dimana komunikasi dilakukan dua arah, pada masing-masing piranti terdapat pengirim dan penerima. Transmisi duplex dibagi menjadi dua, yaitu :
  - Full duplex, dimana data dapat dikirim dan diterima secara bersamaan.
  - Half duplex, dimana data hanya dapat dikirim atau diterima saja pada waktu bersamaan.

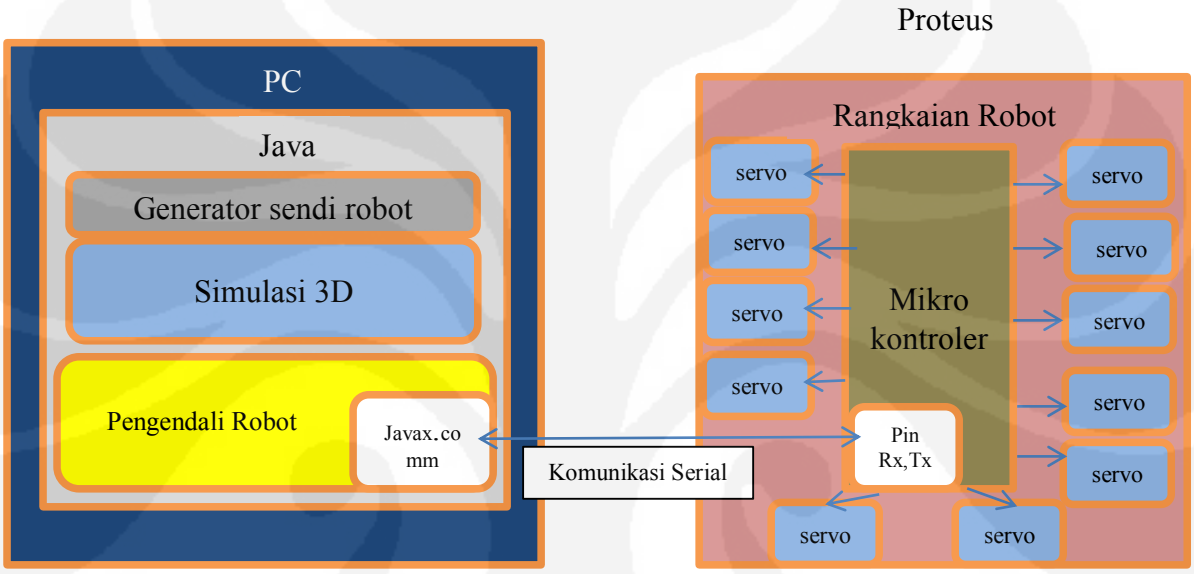
Laju transfer data pada komunikasi serial dinyatakan dalam bps (bit per second). Istilah lain yang dikenal adalah baud rate. Walaupun definisinya sedikit berbeda, baud rate menyatakan perubahan sinyal per detik sedangkan bps menyatakan jumlah bit per detik, namun untuk komunikasi bukan modem kedua istilah tersebut tidak mempunyai perbedaan pengertian, sehingga sering dipakai bergantian[9]. Supaya komunikasi serial dapat berjalan dengan baik maka harus dipastikan kecepatan transfer data dari kedua piranti yang dihubungkan mempunyai baud rate yang sama. Baud rater standar yang sering digunakan antara lain 2400, 4800, 9600, 19200 dan lain-lain.

Pada prinsipnya, komunikasi serial ialah komunikasi dimana pengiriman data yang dikirimkan berupa tegangan kemudian dibaca dalam bit. Jika akan melakukan komunikasi dengan mikrokontroler kebanyakan menggunakan komunikasi serial, di bahasa pemrograman java sendiri terdapat modul atau library yang bisa digunakan untuk melakukan komunikasi tersebut yaitu dengan menggunakan "Javax.comm" API, namun entah kenapa seri yang terbaru dari modul tersebut tidak mendukung OS Windows Seri terbaru, sehingga perlu dilakukan modifikasi dari modul tersebut.

BAB 3
PERANCANGAN SISTEM

3.1 DESKRIPSI SISTEM

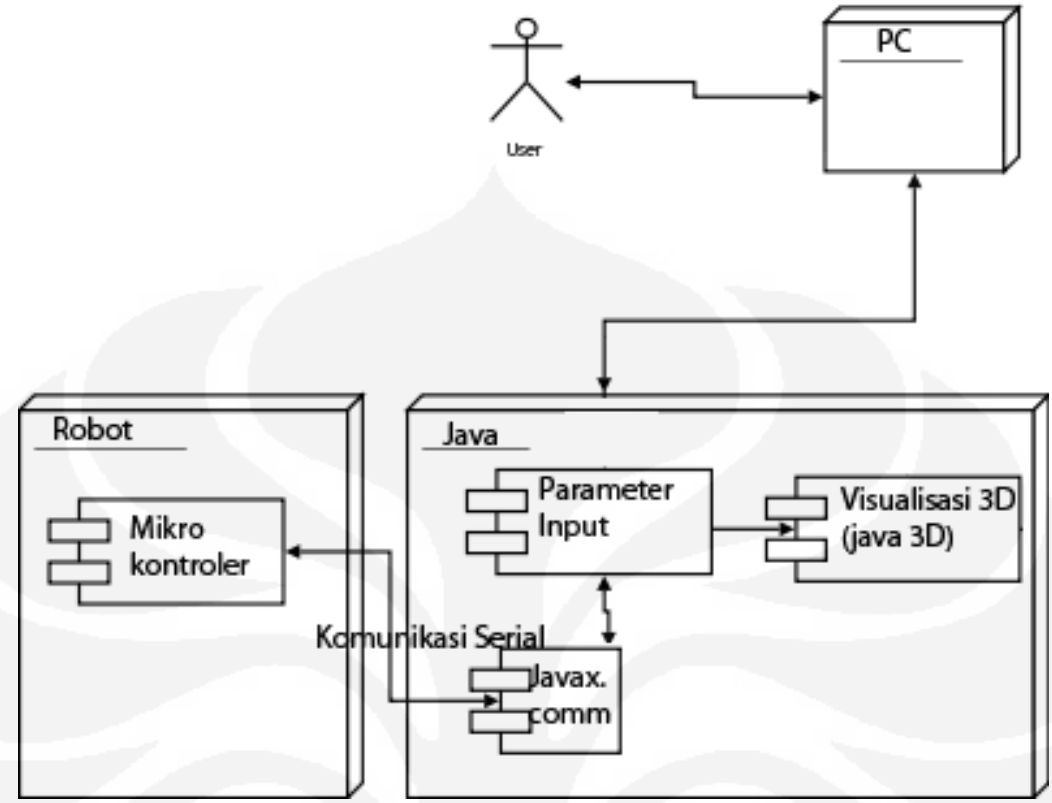
Dalam proses perancangan dan pengerjaan aplikasi/sistem ini dibuat menggunakan bahasa pemrogaman JAVA, dimana untuk grafis 3D nya menggunakan library java3D, bagan alur sistemnya sebagai berikut :



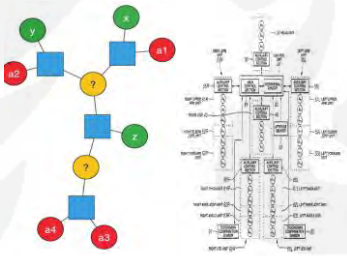
Gambar 3.1 Bagan system

Dari gambar bagan diatas alur dari sistem ini dijelaskan singkat sebagai berikut :

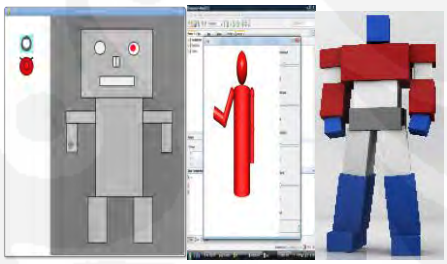
1. PC yang terdapat program java, mengenerate parameter-parameter dari sendi-sendi robot dalam format file XML.
2. Program java akan membentuk visualisasi 3D dengan menggunakan library “java3D”, yang nantinya bisa disimulasikan untuk pergerakan robot secara 3D
3. Untuk mengendalikan robot menggunakan komunikasi serial yang terhubung ke mikrokontroler robot, didalam program java untuk melakukan komunikasi serial menggunakan library “javax.comm”
4. Rangkaian robot akan disimulasikan menggunakan program proteus dimana proteus tersebut terhubung dengan port serial, rangkaian robot tersebut terdapat motor servo sebagai sendi robot yang akan dikendalikan.



Gambar 3.2 UML Sistem

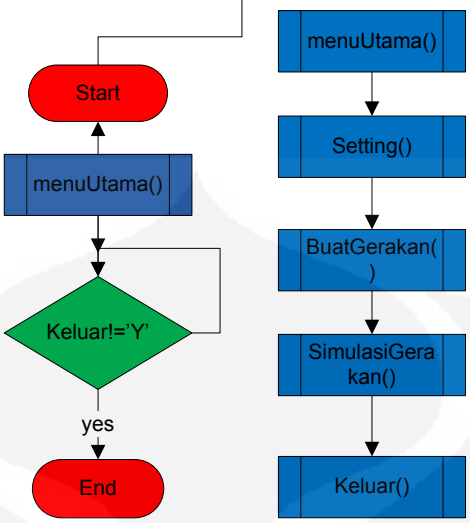


Gambar 3.3.a Pengisian parameter

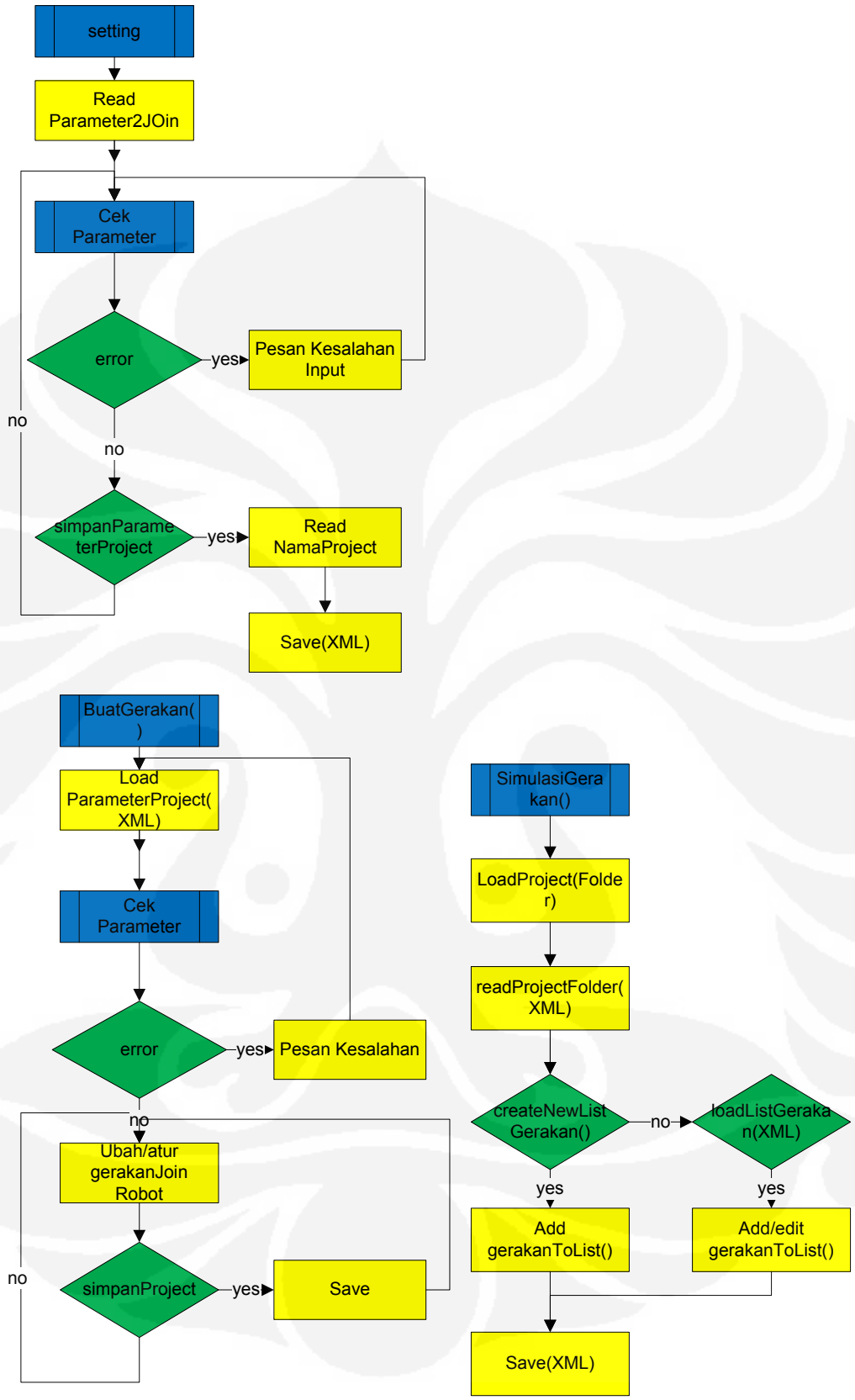


Gambar 3.3.b. hasil visualisasi parameter

Dari gambar diatas dapat dijelaskan, seorang user menggunakan sebuah PC yang didalamnya terdapat aplikasi sistem yang dibuat menggunakan bahasa pemrograman java. User mengisi parameter-parameter untuk menentukan join atau sendi-sendi dari robot, lalu sistem memvisualisasikan parameter-parameter yang diinput tadi menjadi objek 3D,setelah terbentuk visualisasi 3D dari robot, user dapat mensimulasikan gerakan-gerakan robot pada sistem tersebut atau juga gerakan-gerakan bisa diteruskan oleh sistem ke mikrokontroler robot menggunakasan komunikasi serial sehingga dapat mengendalikan gerakan robot secara realtime.



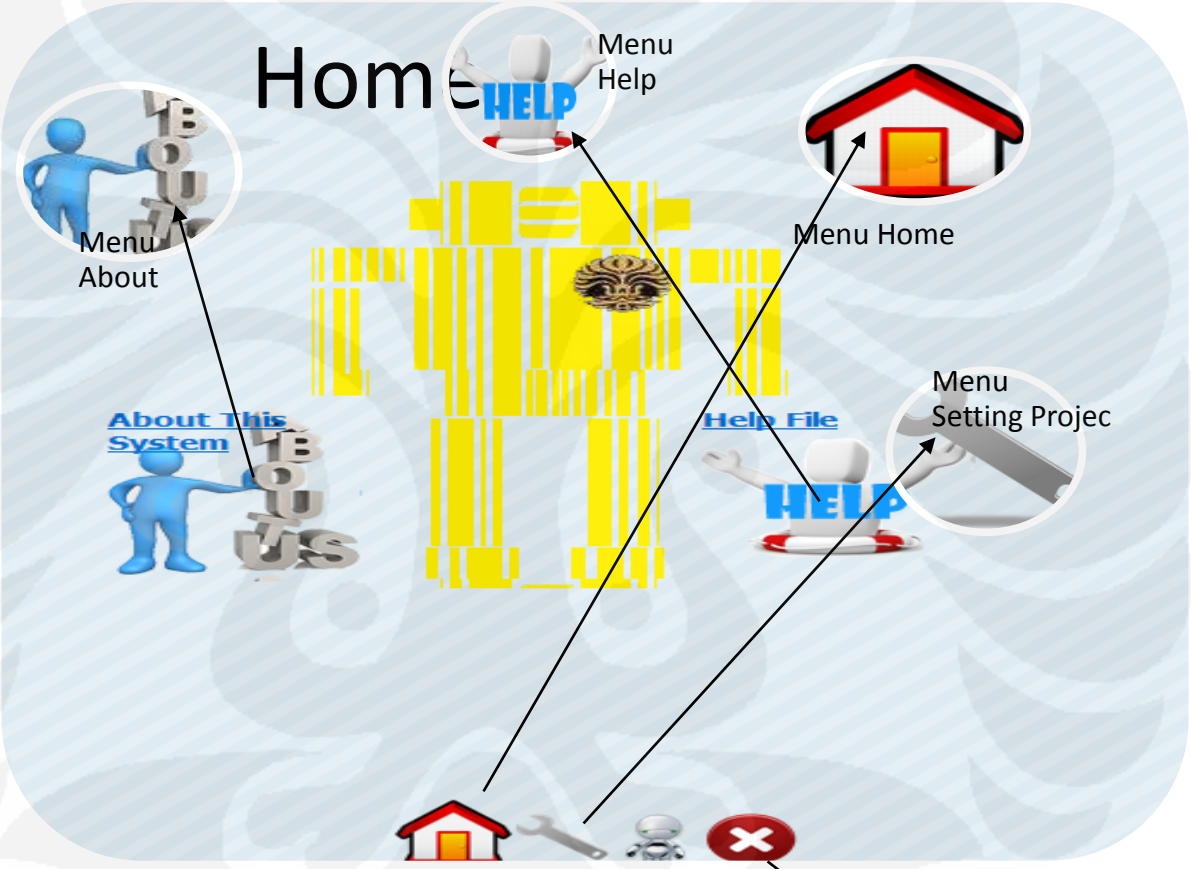
Gambar 3.4 garis besar flowchart sistem



Gambar3.5 flowchart detail system

3.2 Sistem Pengisian Parameter

Perancangan dimulai dengan merancang user interface, dimana pengguna bisa memasukkan parameter-parameter yang nantinya akan dibentuk menjadi visual 3D, di aplikasi tersebut terdapat script-script javascript memakai teknologi „AJAX' yang akan berinteraksi dengan pengguna yaitu menampilkan menu dan juga menerima inputan parameter dari pengguna, tampilan halaman awal / Home pada aplikasi sebagai berikut :

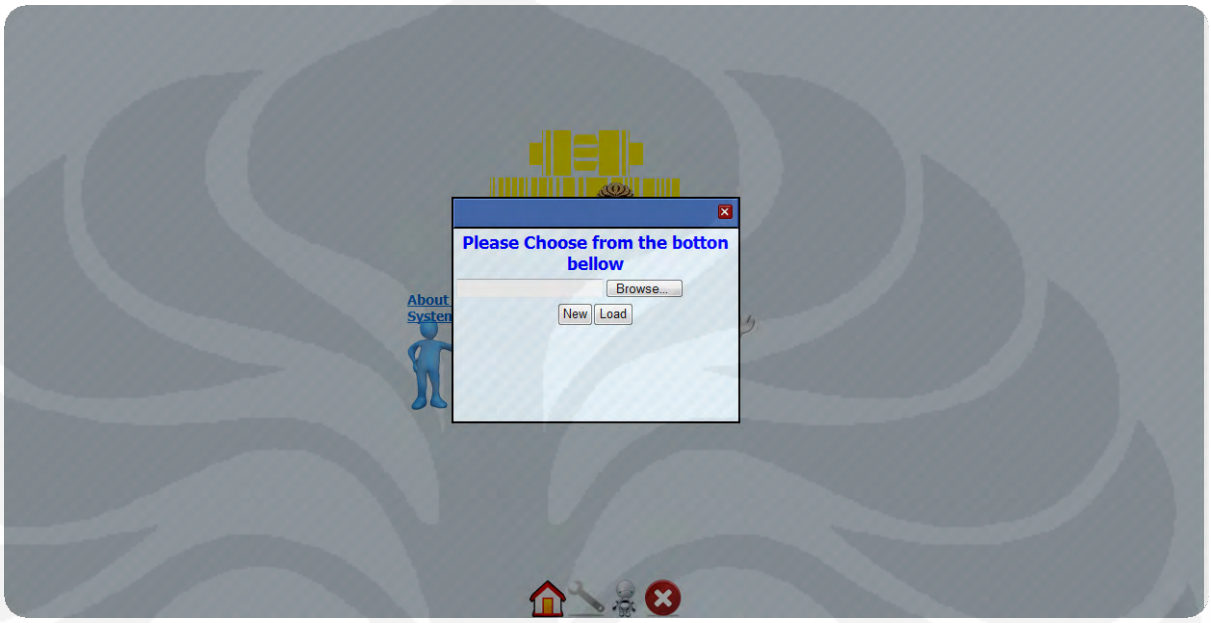


Gambar3.6 Tampilan Halaman awal

Pada gambar 3.5 terdapat beberapa menu yaitu :

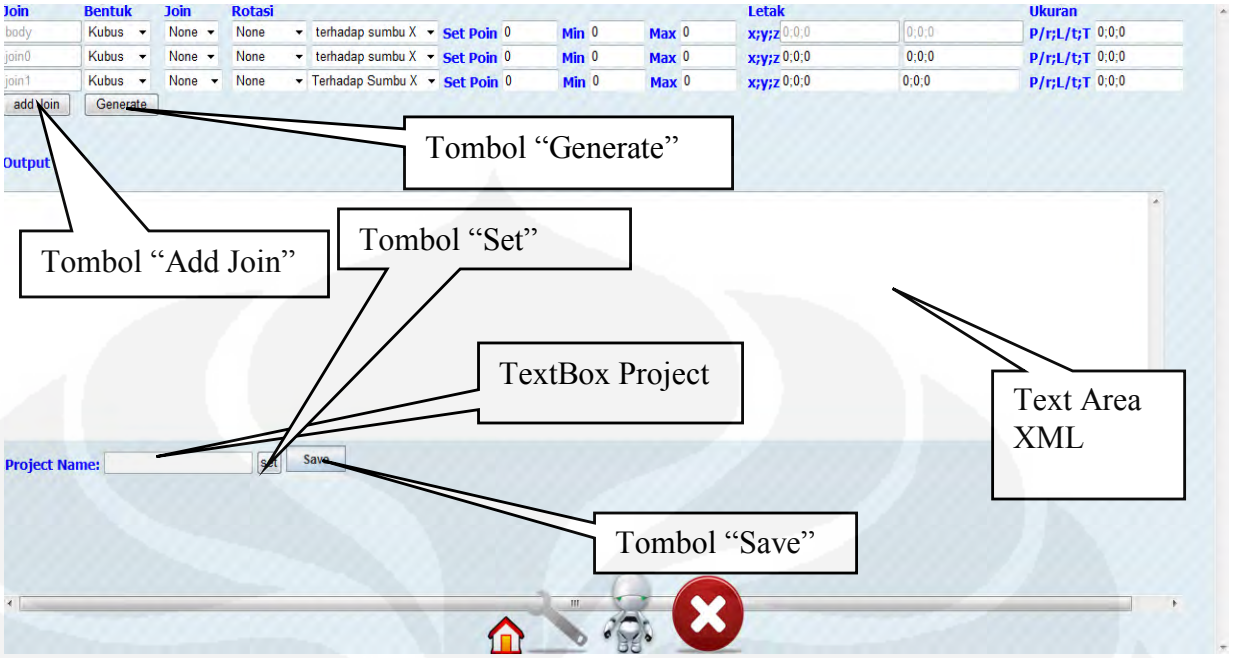
- Home
- Setting Project
- About System
- Help
- Close

- Klick "Menu Home" akan menampilkan halaman utama seperti pada gambar 3.5,
- Klick "Menu Setting Project" akan menampilkan halaman seperti gambar 3.6 :



Gambar 3.7 Tampilan Halaman Setting

Di halaman setting akan ada dua pilihan tombol "New" dan "Load" jika memilih tombol "New" maka akan membuka halaman "New Setting" untuk membuat project baru dari aplikasi yang akan berisi parameter-parameter join dari robot yang akan dibuat, tampilan halamannya sebagai berikut :



Gambar 3.8 Halaman New Setting

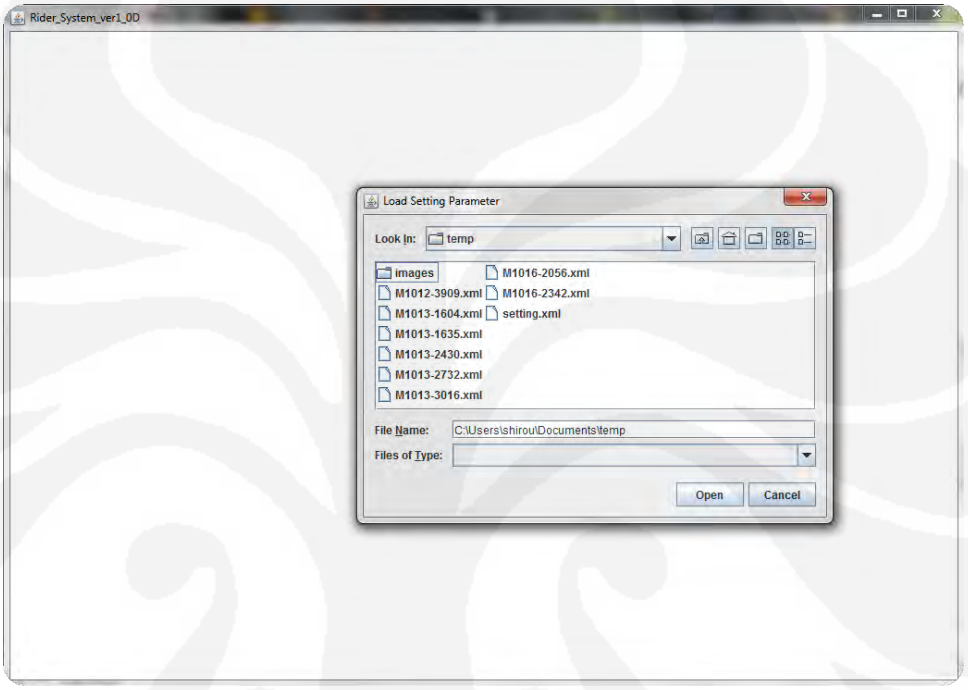
Pada halaman "New Setting" yang seperti dapat dilihat di gambar 3.7 terdapat beberapa tombol yaitu :

1. Tombol "Add Join" , berfungsi untuk menambah join parameter ke dalam project
2. Tombol "Generate", berfungsi untuk menggenerate tag XML dari parameter-parameter yang diinput dan ditampilkan di "Text Area XML"
3. Tombol "Set" untuk mengeset "Nama Project" yang diambil dari "TextBox Project" dan isi setting project yang berasal "Text Area XML" ke aplikasi java
4. Tombol "Save" tombol ini merupakan aplikasi java yang dimana akan menggenerate membuat folder project sesuai yang diset menggunakan tombol "Set" dan juga membuat file "setting.xml" didalam folder tersebut dimana file tersebut berisi parameter-parameter yang sudah dibuat.

Selanjutnya hasil inputan dari parameter-parameter yang digenerate oleh program akan berformat file "XML". Selanjutnya file berformat "XML" tadi bisa dipanggil oleh aplikasi simulasi untuk membentuk visualisasi 3D nya.

1.3 Sistem Visualisasi 3D dan Simulator

Pada saat running sistem visualisasi 3D dan simulator akan meminta file setting berformat "XML", yang didalamnya terdapat settingan parameter-parameter dari join, bentuk dan sendi dari robot, seperti yang terdapat pada gambar dibawah ini :



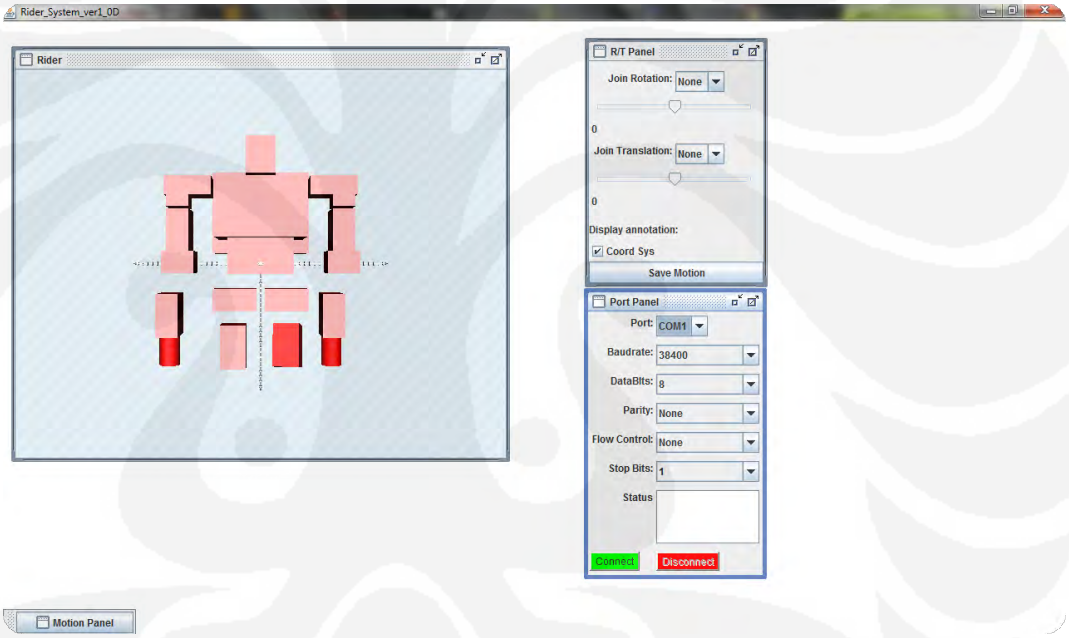
Gambar 3.9 Load setting parameter

Pada gambar diatas aplikasi menanyakan pengguna untuk memasukkan file "setting.xml", yang akan divisualisasikan menjadi gambar 3D dan akan menginisiasi join atau sendi yang bisa dikendalikan. Script untuk meminta pengguna memasukkan file adalah sebagai berikut :

```
void loadFile()
   {
      folderChoose = new JFileChooser();
      folderChoose.setCurrentDirectory(new java.io.File(spath+"/Project"));
      folderChoose.setDialogTitle("Load Setting Parameter");
      folderChoose.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
     folderChoose.setAcceptAllFileFilterUsed(false);
    while(xpath.isEmpty())
    {
              if (folderChoose.showOpenDialog(null) == JFileChooser.APPROVE_OPTION)
                {
                 xpath =  folderChoose.getSelectedFile().toString();
                xdir = folderChoose.getCurrentDirectory().toString();
             }
              else
             {
                System.out.println("No Selection ");
```

```
                int response;
                 response = JOptionPane.showConfirmDialog(null, "Exit this Program");
               System.out.println(response);
                if(response==0)
                {
                  System.exit(1);
                }
               else
                {
                  xpath = "";
                }

            }
         System.out.println(xdir);
        parseXmlFile(xpath);
        System.out.println(xpath);
    }
}
```

Setelah user memilih file yang benar maka akan menampilkan tampilan seperti gambar 3.10 sebagai berikut :

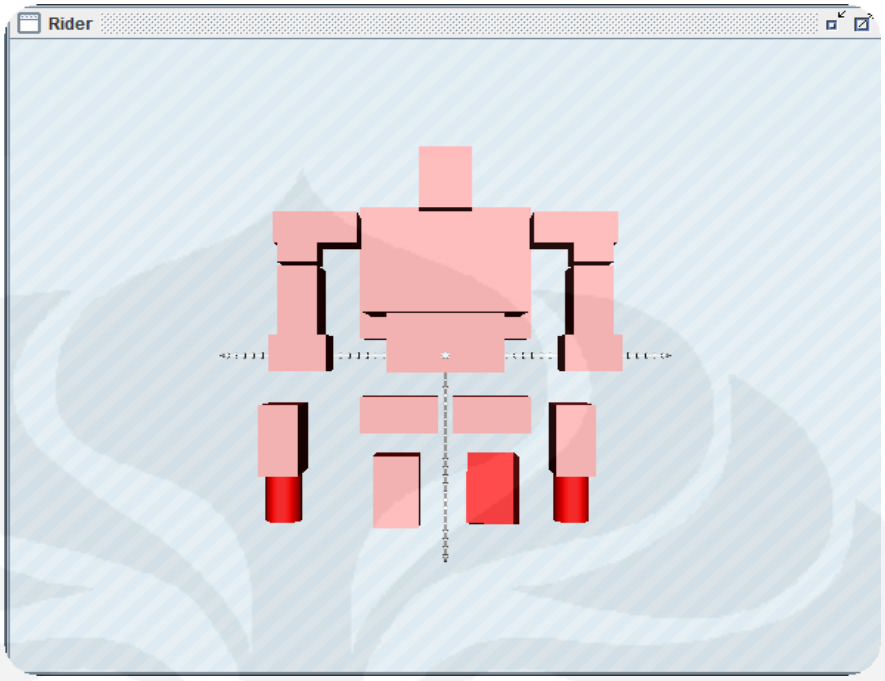


Gambar 3.10 Setelah meload parameter.

Di gambar 3.10 terdapat beberapa panel diantaranya :

1. Panel 3D Robot

   Didalam panel ini terdapat gambar visual 3D hasil dari parameter yang diambil dari file “setting.xml”.



Gambar 3.11 Panel Robot

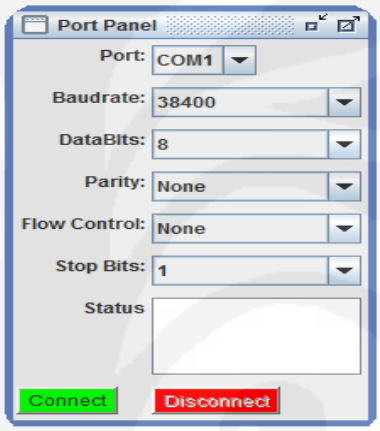
Script untuk membuat visualisasi 3D dari file “XML” yang diinput adalah sebagai berikut :

```
TransformGroup createRobot()
  {

    loadFile();
    Element docEle = dom.getDocumentElement();
    String join_id,join_ke,rotasi_translasi,rotasi_translasi_terhadap;
    int icx = 0;
    NodeList nl = docEle.getElementsByTagName("Join");
    if(nl != null && nl.getLength() > 0)
     {
        for(int i = 0 ; i < nl.getLength();i++) {
        Element el = (Element)nl.item(i);
        NodeList el2 = el.getElementsByTagName("Move");
        Element moveEl = (Element)el2.item(0);
        rotasi_translasi = moveEl.getAttribute("rotasi_translasi");
        rotasi_translasi_terhadap = moveEl.getAttribute("rotasi_translasi_terhadap");
        join_id = el.getAttribute("join_id");
        join_ke = el.getAttribute("join_ke");
        if (join_id.toLowerCase().equals("body"))
        {
            joinBody = createRobotBodyTransform(el);
            System.out.println("shin"+join_id);
        }
        else
        {
            System.out.println("Name: "+join_id);
            join[i-1] = createRobotTransform(el,i-1);
            if(!join_ke.toLowerCase().equals("n"))
             {
                int j = Integer.parseInt(join_ke.substring(4));
                join[j].addChild(join[i-1]);
                System.out.println("shin"+j);
            }
```

```
                else if(join_ke.toLowerCase().equals("n"))
                {
                    joinBody.addChild(join[i-1]);

                }
            }
        }

        return joinBody;
    }
```

2. Panel setting Port

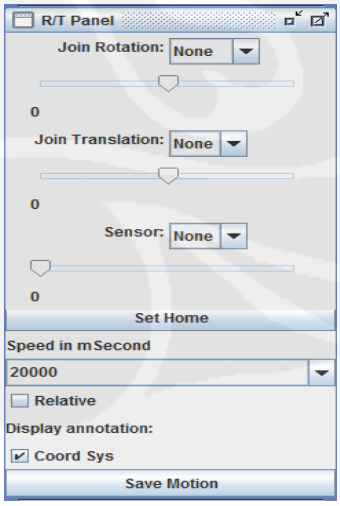
   Di panel ini berisi settingan port serial yang tersedia dan akan dihubungkan dengan microcontroller robot



Gambar 3.12 Panel Port

3. Panel Rotation & Translation

   Pada panel ini terdapat join-join apa saja dari robot yang bisa digerakkan seperti rotasi dan translasi, pada panel ini bisa dibuat untuk membuat motion yang akan digunakan di panel motion.

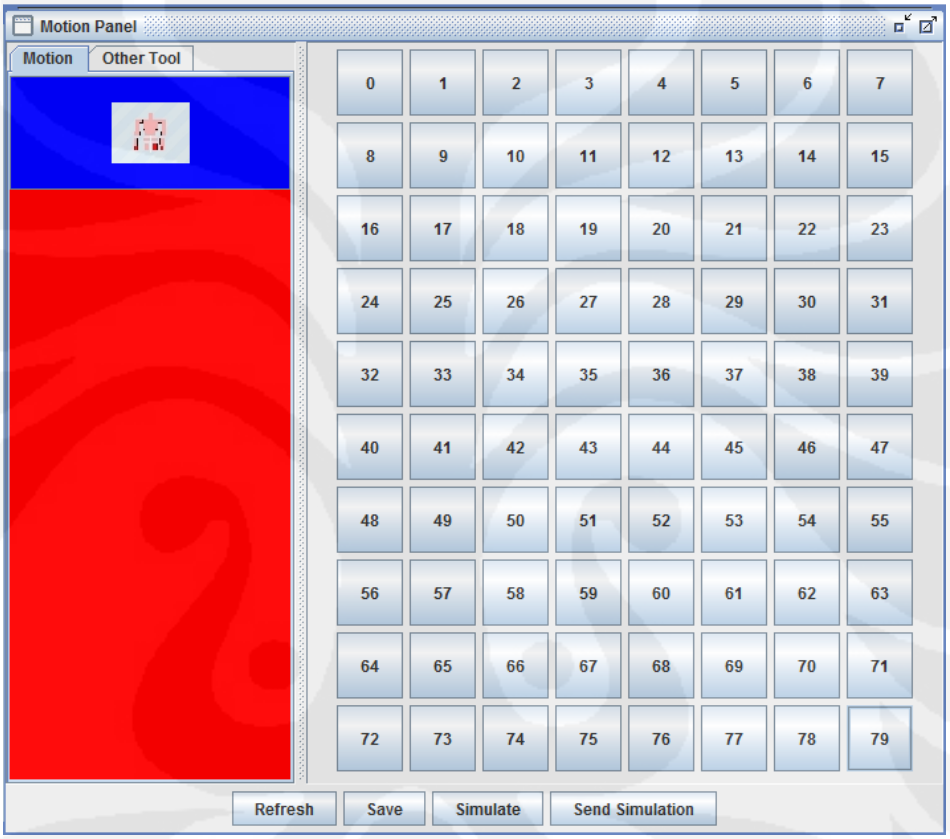


Gambar 3.13 Panel Rotation & Translation

Proses pembuatan motion berupa tag-tag XML yang berisi join yang dikendalikan, nilai perubahan, jenis perubahan sendi robot absolut atau relative, dan *speed* perubahan nilai sendi robot

4. Panel Motion

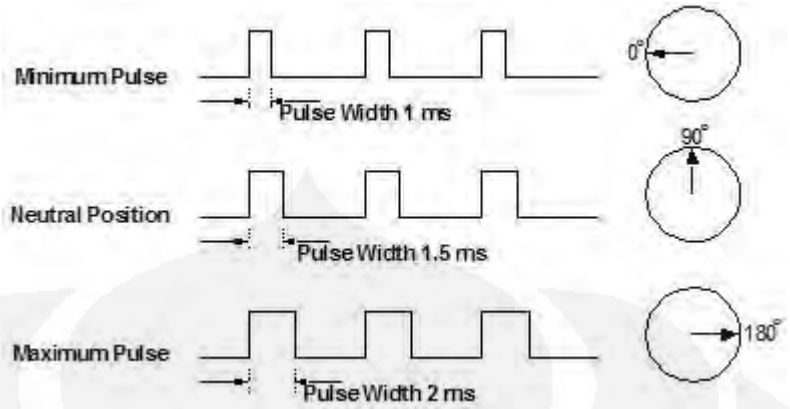
   Panel ini berisi *motion* atau gerakan yang telah dibuat dan juga akan bisa membuat set gerakan yang akan disimulasikan ataupun di kirim ke robot



Gambar 3.14 Panel Motion

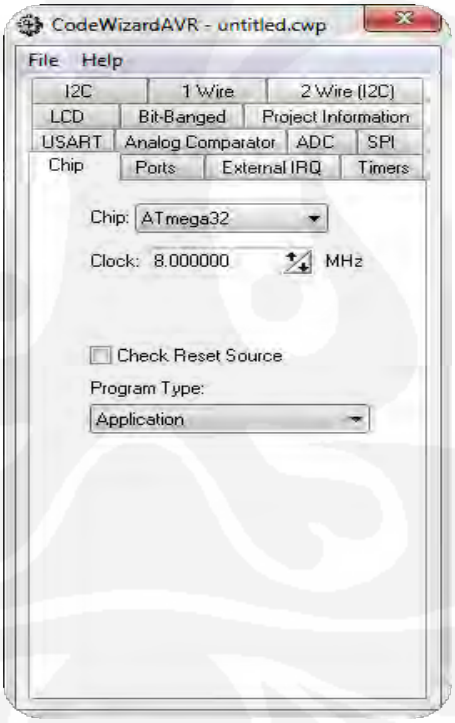
3.4 Sistem Pengendali Motor Servo

Untuk menggerakkan motor servo ke kanan atau ke kiri, tergantung dari lebar *positive pulse* yang diberikan. Untuk membuat servo pada posisi center, diperlukan pulsa 1.5ms. Untuk memutar servo ke kanan, diperlukan pulsa <1.5ms, dan pulsa > 1.5ms untuk berputar ke kiri dengan frekuensi 50Hz. Berikut gambar ilustrasinya :



Gambar 3.15 ilustrasi perputaran servo

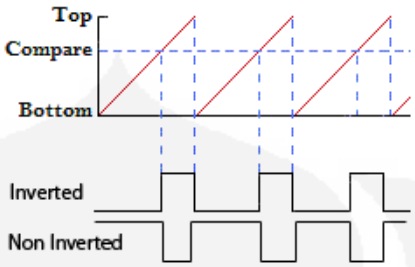
Dalam membuat program untuk microcontroller dapat digunakan bahasa pemrograman Assembler, C, ataupun Basic. Dalam tulisan ini digunakan aplikasi "Code Vision" berbasis bahasa pemrograman C, dimana dengan fasilitas wizardnya dapat mempermudah penulisan parameter awal yang diperlukan. Berikut ini adalaha tampilan wizardnya :



Gambar 3.16 Wizard codevision

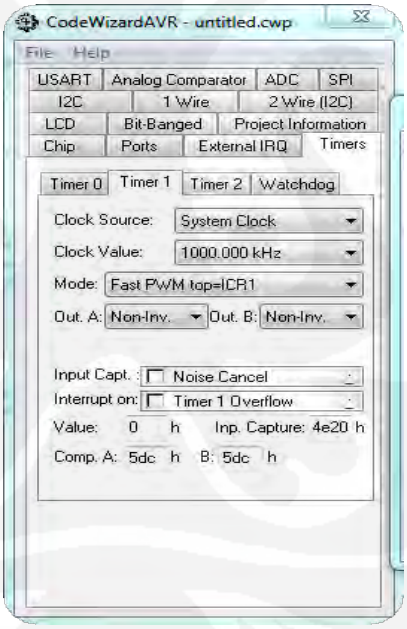
Melalui wizard ini dapat diatur parameter PWM(Pulse Width Modulation) untuk mengendalikan motor servo, dimana diperlukan sinyal PWM. Untuk mudahnya dapat dipilih mode Fast PWM sebagaimana yang terlihat pada gambar

dibawah. Nilai topnya ditentukan oleh ICR sebagai periode sinyal PWM, adapun lebar positif pulsenya ditentukan oleh OCR.



Gambar 3.17 Fast PWM

Untuk mengendalikan motor servo pada posisi tengah diperlukan lebar pulse sebesar 1,5 ms, untuk menghasilkan sinyal ini pada mikrokontroler dengan clock 8MHz diperlukan ICR sebesar 20000 atau 4E20(dalam hexa), nilai ICR akan memberikan frekuensi PWM sebesar 50Hz. Adapun untuk menghasilkan lebar pulse positive sebesar 1,5 ms diperlukan nilai OCR sebesar 1500 atau 5DC(dalam hexa), sebagaimana yang tampak pada gambar berikut :

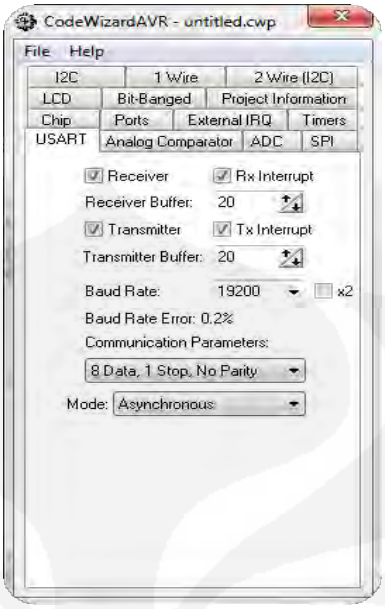


Gambar 3.18 Wizard set PWM

Pilih timer1 di tab “Timers”, timer1 dipilih karena mempunyai kapasitas cacahan 16bit dengan nilai maksimum 65535, selanjutnya pilih mode “Fast PWM” pada gambar diatas dipilih “Fast PWM top=ICR1A”, untuk menghasilkan PWM di kai IC / pin OCIA, maka nilai OCR dimasukkan di Comp.A, dengan mode out A sebagai Non-Inv. Wizard ini akan menghasilkan script seperti berikut:

```
DDRD=0x20; // dimana pin port D.5 adalah pin OCIA
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 1000,000 kHz
// Mode: Fast PWM top=ICR1
// OC1A output: Non-Inv.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x82;
TCCR1B=0x1A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x4E;
ICR1L=0x20;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
```

Kemudian untuk dapat berkomunikasi antara PC dengan mikrokontroler, menggunakan komunikasi serial, pada mikrokontroler dibuat program untuk menerima input dari port serialnya sehingga setiap ada data yang masuk melalui port serial mikrokontroler. Settingan untuk menerima input dari port serial, tampilan wizardnya sebagai berikut :



Gambar 3.19 Wizard setting port serial

Pilih tab "USART" untuk melakukan settingan port serial, lalu diaktifkan Receiver jika ingin menerima input dari port serial, aktifkan Transmitter jika ingin melakukan pengiriman data ke port serial, interrupt baik untuk transmitter dan receiver juga diaktifkan untuk menjamin tidak ada data yang tidak terproses. Kemudian pilih Baudrate yang akan digunakan, Baudrate ini disesuaikan dengan port serial yang disetting di pengendali pada PC. Wizard setting port serial (USART) akan menghasilkan script di bawah ini :

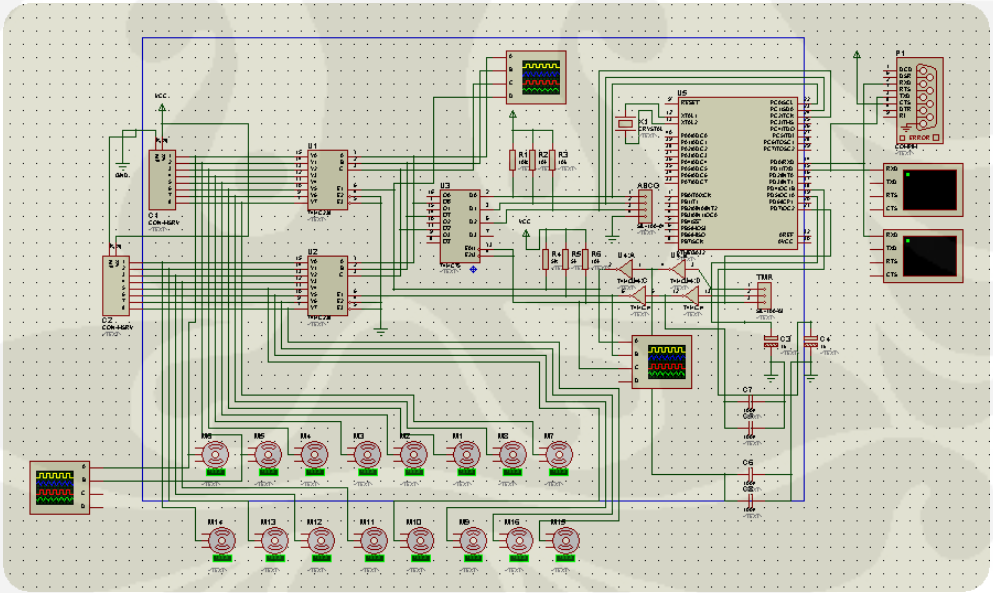
```
// USART Receiver buffer
#define RX_BUFFER_SIZE 20
char rx_buffer[RX_BUFFER_SIZE];
#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif
// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;
// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
```

```
{
char status,data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer[rx_wr_index]=data;
  if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
  if (++rx_counter == RX_BUFFER_SIZE)
    {
    rx_counter=0;
    rx_buffer_overflow=1;
    };
  };
}
#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif
```

```
// USART Transmitter buffer
#define TX_BUFFER_SIZE 20
char tx_buffer[TX_BUFFER_SIZE];
#if TX_BUFFER_SIZE<256
unsigned char tx_wr_index,tx_rd_index,tx_counter;
#else
unsigned int tx_wr_index,tx_rd_index,tx_counter;
#endif
// USART Transmitter interrupt service routine
interrupt [USART_TXC] void usart_tx_isr(void)
{
if (tx_counter)
  {
  --tx_counter;
  UDR=tx_buffer[tx_rd_index];
  if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
  };
}
#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
while (tx_counter == TX_BUFFER_SIZE);
#asm("cli")
if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
  {
  tx_buffer[tx_wr_index]=c;
  if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
  ++tx_counter;
  }
```

```
            else
              UDR=c;
            #asm("sei")
}

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 19200
UCSRA=0x00;
UCSRB=0xD8;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x19;
```

Program mikrokontroler ini, didesain untuk mengecek setiap kali apakah ada data yang dikirim oleh PC. Untuk menguji program yang dibuat apakah sudah sesuai dengan yang diinginkan dapat dilakukan pada aplikasi "Proteus" dengan skematik sebagai berikut :



Gambar 3.20 Skematik rangkaian

Pada gambar skematik diatas digunakan mikrokontroler ATMega32, compimp untuk komunikasi port serial dengan PC, 16 motor servo, Oscilator untuk melihat sinyal, virtual terminal untuk melihat data yang masuk dan keluar dari port serial.

BAB 4
PENGUJIAN SISTEM

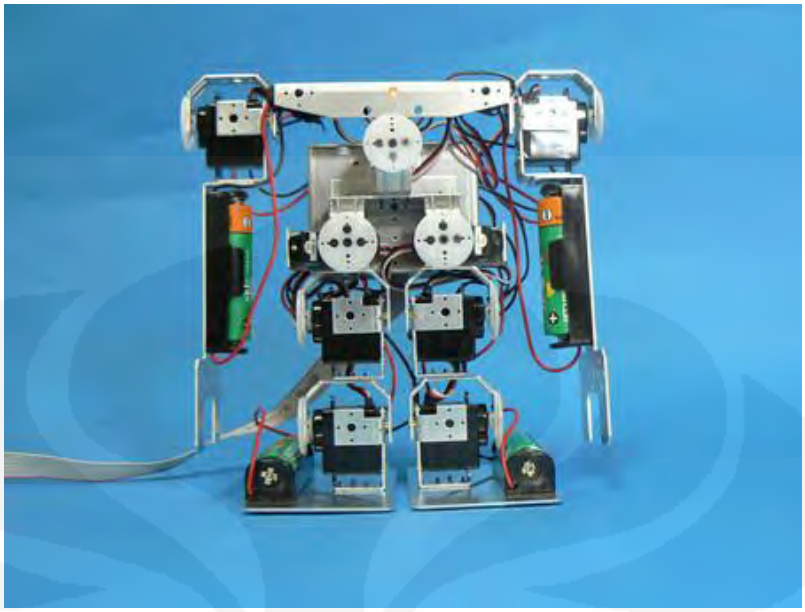
Pada tahap pengujian ini yang diujikan adalah pengisian parameter, visualisasi 3D, simulasi 3D dan komunikasi serial ke mikrokontroler. Pengujian ini dilakukan untuk mengklarifikasi apakah parameter yang ada di XML bisa di visualisassikan ke gambar 3D, apakah keluaran data dari serial PC bisa diterima oleh mikrokontroler, dan apakah data dari serial tadi bisa diterjemahkan oleh mikrokontroler untuk menggerakkan motor servo.

4.1 Pegujian Pengisian Parameter Model Robot

Pengisian parameter diperlukan untuk membuat file XML yang didalamnya terdapat parameter-parameter untuk divisualisasikan ke bentuk gambar 3D. Tampilan untuk pengisian parameter adalah sebagai berikut :

Gambar 4.1 Sistem pengisian parameter

Saat dilakukan pengisian parameter, parameter yang dimasukkan harus sesuai dengan format yang ada. Dalam pengujian ini menggunakan robot “puchi robo “ sebagai visualisasi 3D nya, bentuk fisik dari “puchi robo” tersebut seperti gambar di bawah ini :



Gambar 4.2 bentuk fisik “puchi robo”

Setelah diklik tombol generate akan menghasilkan tag-tag XML yang ditunjukkan di lampiran 1, adapun contoh potongan XMLnya sebagai berikut :

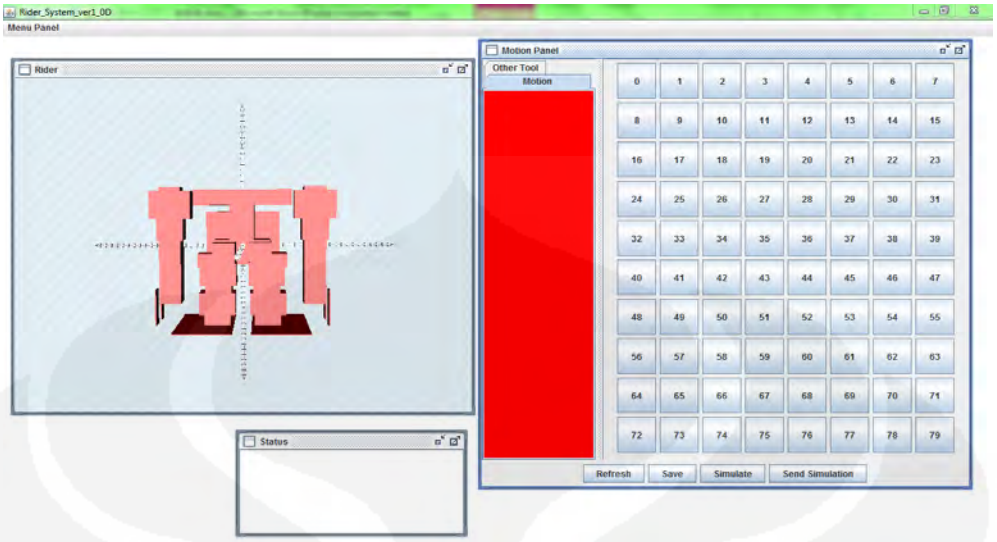
```
<Robot RiderSystem='Rider System'>
<Join join_id='body' bentuk='k' join_ke='n'>
<Move rotasi_translasi='n' rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;0;0
</Letak_xyz>
<letak_Miring>0;0;0
</letak_Miring>
<Ukuran>0;0;0
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join0' bentuk='k' join_ke='n'>
<Move rotasi_translasi='n' rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;1.35;-2.1
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>6.3;4.95;2.75
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join1' bentuk='k' join_ke='join0'>
<Move rotasi_translasi='r' rotasi_translasi_terhadap='z'>
<setP>0
</setP>
<min>-90
</min>
<max>90
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;7.5;5
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>8.1;1.35;2.1
</Ukuran>
</Letak_Ukuran>
</Join>
….
….
<Join join_id='join27' bentuk='k' join_ke='join26'>
<Move rotasi_translasi='n' rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>-2;-2;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>4.7;0.1;8.2
</Ukuran>
</Letak_Ukuran>
</Join>
</Robot>
```

Pada contoh tag XML diatas root dari node XMLnya adalah “Robot”, lalu ada node “join”. Diambil contoh node “join” dengan id join_id=’join1’, didalam node join terdapat attribute bentuk=’k’ menandakan join ini berbentuk kubus, kemudian attribute join_ke=’join0’ menandakan join ini menempel pada join0. Kemudian didalam node join terdapat subnode bernama “Move” yang didalam subnode ini terdapat attribute rotasi_translasi='r' yang menandakan join ini bisa dirotasi dan attribute rotasi_translasi_terhadap='z' menandakan rotasi atau translasi join ini terhadap sumbu z, didalam subnode ini terdapat childnode yaitu “setP” yang nilai dari child ini menetukan set poin dari join, “min” nilai childnode ini yang menentukan set minimum dari rotasi atau translasi yang dilakukan terhadap join, lalu “max” nilai childnode ini yang menentukan set maksimum dari rotasi atau translasi yang dilakukan terhadap join. Selain subnode “Move” terdapat pula subnode “Letak_Ukuran” yang didalam subnode ini terdapat childnode “Letak_xyz” yang nilai dari childnode ini berisi letak sumbux, sumbu y, dan sumbu z dari join ini, terdapat pula childnode “letak_Miring” yang menentukan sudut kemiringan dari join penulisan nilai diawali dengan sumbu lalu diikuti nilai sudutnya pada contoh join ini x0;y0;z0 berarti join ini sudut miring terhadap sumbu x,y, dan z masing-masing 0^o selain dua childnode tadi terdapat pula childnode “Ukuran” yang nilai ini berisi ukuran dari join yang dibuat pada contoh “4.7;0.1;8.2” karena join berbentuk kubus maka nilai ini berarti ukuran panjang, lebar, dan tinggi dari join.

Kemudian ketika tombol save diklik maka aplikasi akan membuat folder dan mengisikan di dalam folder tersebut file “setting.xml” dimana file tersebut berisi dari parameter-parameter yang tadi telah dibuat oleh sistem. Selanjutnya file “setting.xml” bisa divisualisasikan ke dalam gambar 3D.

4.2 Pegujian Visualisasi dan Simulasi 3D

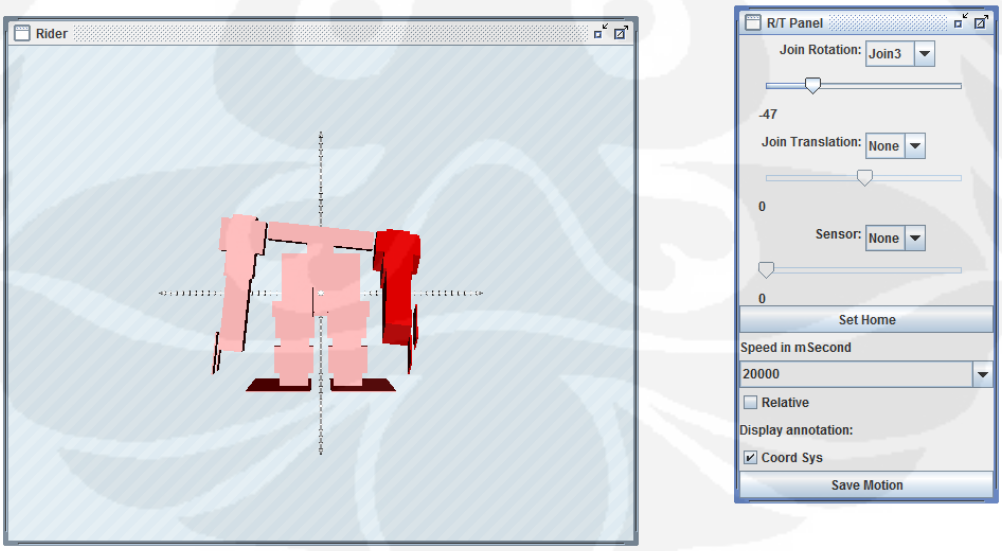
Untuk menjalankan aplikasi visualisasi dan simulasi 3D, aplikasi java akan meload file “setting.xml” dan akan menerjemahkan parameter yang ada di file “setting.xml” tersebut menjadi visual 3D, seperti pada gambar sebagai berikut:



Gambar 4.3 Gambar Aplikasi Visual dan simulasi 3D

4.3 Pegujian Pengisian Parameter Motion

Untuk membentuk satu gerakan dari robot, sendi robot yang digerakkan dapat diset pada panel "R/T panel" dengan memilih sendi robot pada combobox join lalu menggerakkan slider sesuai dengan nilai yang dipilih, memilih kecepatan perubahan gerakan robot, serta memilih perubahan nilai sendi tersebut absolut atau relative. Dapat dilihat pada gambar berikut :



Gambar 4.4 Set sendi untuk motion

Ketika tombol "save motion" diklik maka program akan membuat file XML yang berisi perubahan-perubahan dari sendi robot berupa sudut jika

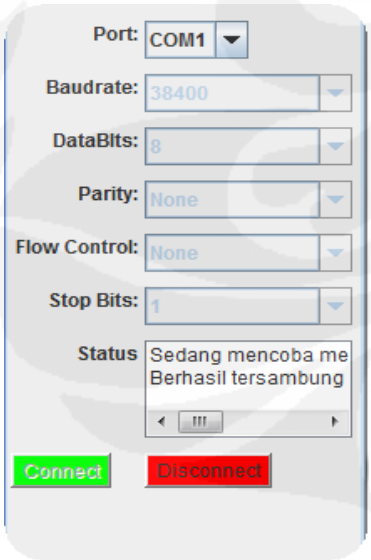
perubahannya rotasi, berupa panjang dalam cm jika perubahannya translasi, contoh file XML nya sebagai berikut :

```
<Motion>
<join Join_id="1" Speed="20000">-0</join>
<join Join_id="3" Speed="20000">-90</join>
<join Join_id="5" Speed="20000">-90</join>
<join Join_id="16" Speed="20000">0</join>
<join Join_id="17" Speed="20000">0</join>
<join Join_id="19" Speed="20000">0</join>
<join Join_id="21" Speed="20000">0</join>
<join Join_id="23" Speed="20000">0</join>
<join Join_id="26" Speed="20000">0</join>
</Motion>
```

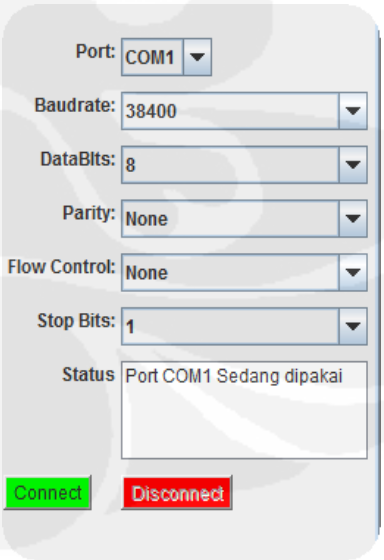
Pada contoh tag-tag xml diatas dapat dilihat sendi yang berubah adalah join dengan join_id=3 dan join_id=5, terlihat join_id=3 diset ke sudut -90, dan join_id=5 diset ke sudut -90, motion ini diset dengan kecepatan 20000µS atau sama dengan 20mS.

4.4 Pegujian Komunikasi ke Serial

Pada gambar 4.3 dibagian panel "Port Panel" terdapat settingan untuk komunikasi serial seperti pengaturan port yang akan digunakan, baudrate, databits, parity, flow control, dan stop bits. Berdasarkan parameter yang ditentukan, aplikasi akan meminta request ke port serial setelah tombol "Connect" di klik, maka di textbox "status" akan menampilkan status keberhasilan koneksi, seperti yang terlihat pada gambar dibawah ini :



Gambar 4.5.a Komunikasi berhasil

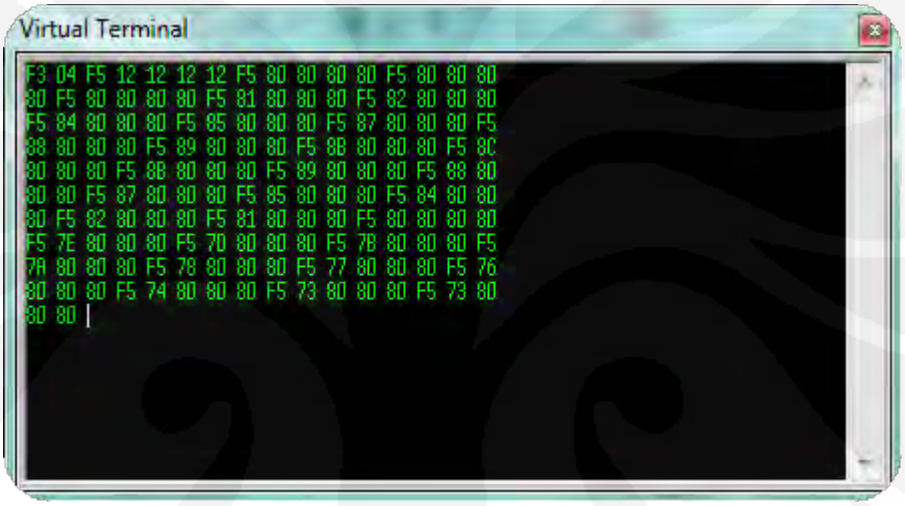


Gambar 4.5.b Komunikasi gagal

Pada gambar 4.3.a diperlihatkan komunikasi dengan port serial telah berhasil, sedangkan pada gambar 4.2.b komunikasi dengan port serial tidak berhasil karena com port nya sedang sibuk atau diakses oleh aplikasi lain.

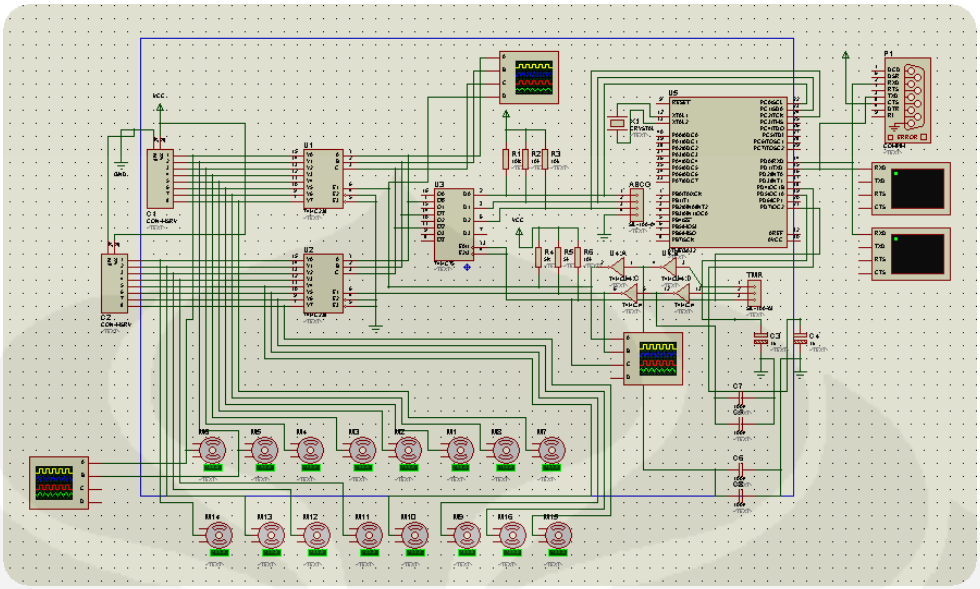
4.5 Pengujian Pengiriman Data ke Port Serial

Pengujian ini dilakukan untuk memverifikasi apakah data yang dikeluarkan oleh aplikasi java terkirim ke port serial, hasilnya terlihat dari virtual trerminal dengan format hexadesimal. Berikut adalah gambar hasil data yang dikeluarkan aplikasi yang dibaca oleh aplikasi "virtual terminal" :



Gambar 4.6 Data serial yang dibaca virtual terminal di isis

Dari gambar tampak data yang masuk diawali dengan "F3" lalu "04" lalu "F5" "12" "12" dan seterusnya, "F3" ini adalah inisiasi jumlah join, sedangkan "04" adalah jumlah join yang bisa dikendalikan, lalu "F5" menandakan header dari data, "12" adalah data dari sudut yang akan diset ke motor servo. Kemudian akan di test data yang terkirim ke port serial tersebut apakah bisa dibaca oleh mikrokontroler, dalam hal ini menggunakan simulasi rangkaian di program "Isis". Berikut adalah gambar rangkaiaan yang akan disimulasikan :



Gambar 4.7 Simulasi menggunakan Isis

Dari pengujian sistem diatas karena terdapat perbedaan kecepatan processor di PC dengan mikrokontroller dalam memproses dan mengkalkulasi data, jadi perlu diatur delay pengiriman data dari program Java di PC.

4.6 Pengujian Simulasi Motion Robot

Untuk melakukan simulasi motion dari robot terdapat pada panel “Motion panel” yang terlihat pada gambar 4.3. pada panel “Motion panel” drag tombol yang berada disebelah kiri ke urutan motion yang berada disebelah kanan dari panel ”Motion panel” ketika tombol “Save” diklik maka program akan membuat file motion dalam bentuk XML yang berisi urutan motion, contoh filenya sebagai berikut :

```
<Motion>
<StepMotion>A10304316</StepMotion>
<StepMotion>A10291454</StepMotion>
<StepMotion>A10291346</StepMotion>
<StepMotion>A10304316</StepMotion>
<StepMotion>A10292757</StepMotion>
<StepMotion>A10292742</StepMotion>
<StepMotion>A10292238</StepMotion>
</Motion>
```

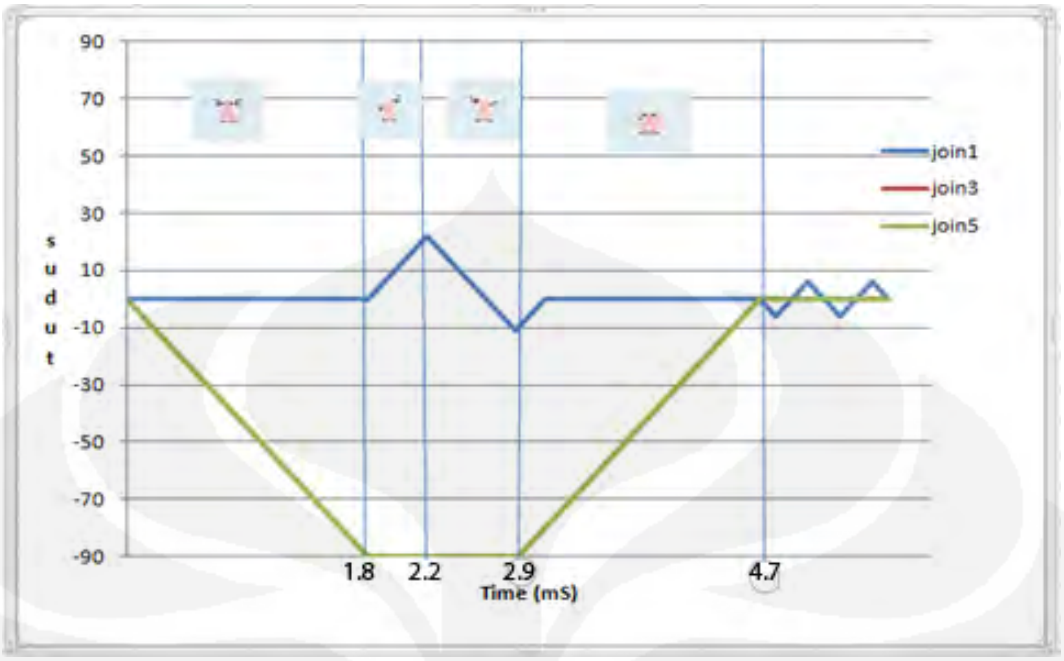
Untuk melakukan simulasi gerakan dapat dilakukan dengan klik tombol “Simulation” atau tombol “Send Simulation”, bedanya dari dua tombol ini adalah send simulation selain melakukan simulasi secara 3D di program java perintah ini juga akan mengirim data-data sudut ke port serial.

Pada urutan motion yang terdapat di tag XML diatas, dijelaskan detailnya di dalam tabel 4.1 sebagai berikut :

Tabel 4.1. Step Motion gerakan robot

Motion Step	Nama motion	Jenis perubahan	join	Perubahan sudut	Speed(ms)
M01	A10304316	Absolut	Join1	0	20
			Join3	-90	20
			Join5	-90	20
M02	A10291454	Absolut	Join1	22	20
			Join3	-90	20
			Join5	-90	20
M03	A10291346	Absolut	Join1	-11	20
			Join3	-90	20
			Join5	-90	20
M04	A10304316	Absolut	Join1	0	20
			Join3	0	20
			Join5	0	20
M05	A10292757	Absolut	Join1	-11	20
			Join3	0	20
			Join5	0	20
M06	A10292742	Absolut	Join1	22	20
			Join3	0	20
			Join5	0	20

Dari tabel 4.1 setiap perubahan 1^{o} dari sendi dilakukan dengan kecepatan 20ms, maka didapat hasil grafik perubahan sudut dari sendi robot yang dikendalikan sebagai berikut :



Gambar 4.8 grafik perubahan sudut sendi

Dari grafik diatas setiap perubahan gerakan akan dilakukan setiap 20ms, step motion yang pertama sistem menset join1 ke sudut 0^o sedangkan join3 dan join5 di set ke sudut -90^o, step motion berikutnya join1 diset ke sudut 25^o sedangkan join3 dan join5 tetap di sudut -90^o, selanjutnya join1 diset ke sudut -10^o join3 dan join5 masih tetap di sudut -90^o,kemudian step motion ke 4 join1, join3, dan join5 masing-masing diset ke sudut 0^o.

BAB 5

KESIMPULAN

Dari hasil pengujian yang dilakukan pada bab 4. Dapat ditarik kesimpulan sebagai berikut :

- Pengisian parameter harus sesuai dengan aturan yang telah ditentukan, jika tidak akan terjadi error dalam visualisasi dan simulasi 3D
- Data yang dikirim dari aplikasi Java yang dibuat dapat diterima oleh mikrokontroler
- Dari java bisa dilakukan pengendalian robot secara realtime yang ditampilkan di visual 3D
- Karena kecepatan processor di PC dengan di Mikrokontroler berbeda maka diatur delay di program java untuk menyesuaikan dengan mikrokontroler dalam mengolah data yang diterima, hal ini dilakukan untuk menghindari data yang hilang dan belum sempat diolah oleh mikrokontroler.
- Sudut yang dihasilkan oleh mikrokontroler untuk motor servo terdapat perbedaan sedikit dari sudut yang ditampilkan oleh Java

DAFTAR REFERENSI

[1] Java. Wikipedia Indonesia

<http://id.wikipedia.org/wiki/Java>. Februari 2010

[2] Java Applet. Wikipedia Indonesia

<http://id.wikipedia.org/wiki/Java_applet>. Februari 2010

[3] Java 3D API. Sun Microsystem.

<http://java.sun.com/javase/technologies/desktop/java3d/>. Februari 2010

[4] Mikrokontroller, Pengertian Mikrokontroler.

<http://curhatandhandi.blogspot.com/2009/07/microcontroller.html>. Februari 2010

[5] Junaedi ,Moh**,** P engantar XML. Ilmu Komputer.

< http://ilmukomputer.org/2008/11/25/pengantar-xml/ > . Maret 2010.

[6] Interfacing Port Paralel Komputer, Serial. *Toko elektronika*
<http://www.toko-elektronika.com/tutorial/paralel.html>. maret 2010

[7] Motor Servo. Indoskripsi.

<http://one.indoskripsi.com/node/7903 (motor servo)>. Januari 2010

[8] Motor Stepper dan Servo.Toko elektronika

< http://www.toko-elektronika.com/tutorial/Stepper.html>. April 2010

[9] Adi, Agung N, Mekatronika, Graha Ilmu, 2010

LAMPIRAN

1. Tag-tag XML pengisian parameter (setting.xml)

```
<Robot RiderSystem='Rider System'>
<Join join_id='body' bentuk='k' join_ke='n'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;0;0
</Letak_xyz>
<letak_Miring>0;0;0
</letak_Miring>
<Ukuran>0;0;0
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join0' bentuk='k' join_ke='n'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;1.35;-2.1
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>6.3;4.95;2.75
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join1' bentuk='k' join_ke='join0'>
<Move rotasi_translasi='r'
rotasi_translasi_terhadap='z'>
<setP>0
</setP>
<min>-90
</min>
<max>90
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;7.5;5
<Join join_id='join14' bentuk='k'
join_ke='join13'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>3;-2.5;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>1.8;2.2;3.4
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join15' bentuk='k'
join_ke='join13'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>-3;-2.5;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>1.8;2.2;3.4
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join16' bentuk='k'
join_ke='join14'>
<Move rotasi_translasi='r'
rotasi_translasi_terhadap='z'>
<setP>0
</setP>
<min>-90
</min>
<max>90
</max>
</Move>
```

```
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>8.1;1.35;2.1
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join2' bentuk='k' join_ke='join0'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>-90
</min>
<max>-90
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;3.8;5.3
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>2;3.9;3.4
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join3' bentuk='k' join_ke='join1'>
<Move rotasi_translasi='r'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>-90
</min>
<max>90
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>12;-1.3;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>3.4;2.5;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join4' bentuk='k' join_ke='join3'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;-4;0
</Letak_xyz>
<Letak_Ukuran>
<Letak_xyz>2;-1.1;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>2.4;1.2;2.1
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join17' bentuk='k'
join_ke='join15'>
<Move rotasi_translasi='r'
rotasi_translasi_terhadap='z'>
<setP>0
</setP>
<min>-90
</min>
<max>90
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>-2;-1.1;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>2.4;1.2;2.1
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join18' bentuk='k'
join_ke='join16'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>-1;-2.5;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>3;2;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join19' bentuk='k'
join_ke='join18'>
<Move rotasi_translasi='r'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>-90
</min>
<max>90
```

```
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>2.2;2.4;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join5' bentuk='k' join_ke='join1'>
<Move rotasi_translasi='r'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>-90
</min>
<max>90
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>-12;-1.3;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>3.4;2.5;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join6' bentuk='k' join_ke='join5'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;-4;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>2.2;2.4;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join7' bentuk='k' join_ke='join4'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;-3.3;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;-2.6;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>2.2;2;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join20' bentuk='k'
join_ke='join17'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>1;-2.5;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>3;2;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join21' bentuk='k'
join_ke='join20'>
<Move rotasi_translasi='r'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>-90
</min>
<max>90
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;-2.6;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>2.2;2;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join22' bentuk='k'
join_ke='join19'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
```

```
</letak_Miring>
<Ukuran>1.8;10.5;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join8' bentuk='k' join_ke='join6'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;-3.3;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>1.8;10.5;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join9' bentuk='k' join_ke='join7'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>2.2;-11;2.5
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>0.1;3.2;1.2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join10' bentuk='k' join_ke='join7'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>2.2;-11;-2.5
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0.2;-2.7;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>3;2;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join23' bentuk='k'
join_ke='join22'>
<Move rotasi_translasi='r'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>-90
</min>
<max>90
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;-2;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>2.2;2;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join24' bentuk='k'
join_ke='join23'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>2;-2;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>4.7;0.1;8.2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join25' bentuk='k'
join_ke='join21'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
```

```
<Ukuran>0.1;3.2;1.2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join11' bentuk='k' join_ke='join8'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>-2.2;-11;2.5
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>0.1;3.2;1.2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join12' bentuk='k' join_ke='join8'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>-2.2;-11;-2.5
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>0.1;3.2;1.2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join13' bentuk='k' join_ke='join2'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;-4;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>4.6;0.1;2.9
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>-0.2;-2.7;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>3;2;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join26' bentuk='k'
join_ke='join25'>
<Move rotasi_translasi='r'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>-90
</min>
<max>90
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>0;-2;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>2.2;2;2
</Ukuran>
</Letak_Ukuran>
</Join>
<Join join_id='join27' bentuk='k'
join_ke='join26'>
<Move rotasi_translasi='n'
rotasi_translasi_terhadap='x'>
<setP>0
</setP>
<min>0
</min>
<max>0
</max>
</Move>
<Letak_Ukuran>
<Letak_xyz>-2;-2;0
</Letak_xyz>
<letak_Miring>x0;y0;z0
</letak_Miring>
<Ukuran>4.7;0.1;8.2
</Ukuran>
</Letak_Ukuran>
</Join>
</Robot>
```

</Ukuran> </Letak_Ukuran> </Join>	