EFEKTIVITAS *ROS DEVELOPMENT STUDIO* SEBAGAI SIMULATOR PENDUKUNG PENGEMBANGAN ALGORITMA PLATFORM ROBOTIKA

Kevin Julianto Effendy[1], **Nova Eka Budiyanta**[2]
[1,2]Program Studi Elektro dan Fakultas Teknik
Universitas Katolik Indonesia Atma Jaya
Jakarta, Indonesia
e-mail: [1]kevinjulianto@yahoo.co.id, [2]nova.eka@atmajaya.ac.id*

ABSTRAK

Penelitian ini bertujuan untuk menganalisa perangkat simulasi robotika yang dapat digunakan untuk mendukung pengembangan algoritma pada proses pemodelan robot. Terdapat beberapa perangkat simulasi pemodelan algoritma robot yang popular digunakan oleh pengembang robotika. Penelitian ini berfokus pada analisa dari simulasi yang dibuat pada Robot Operating System Development Studio (ROSDS) dan juga MATLAB menggunakan metode komparasi. Ada beberapa fitur yang dibandingkan dalam penelitian ini seperti aksesibilitas, kecepatan komputasi, persyaratan sistem, *middleware*, simulator dan *motion planning* yang digunakan. Berdasarkan penilaian dari fitur tersebut akan didapatkan hasil penelitian yang menyatakan bahwa ROSDS lebih efektif dibandingkan dengan MATLAB dalam mendukung simulasi pengembangan robotika. Guna mendukung pernyataan tersebut, studi kasus pengembangan simulasi lengan robot dilakukan pada penelitian ini dan berhasil dengan baik.

Kata kunci: RDS, ROS, komparasi, efektivitas, MATLAB

ABSTRACT

This research aims to analyze robotics simulation tools that can be used to support algorithm development in robot modeling process. There are several simulation tools for robot algorithm modeling that are popularly used by robotics developers. This research focuses on analyzing the simulation using Robot Operating System Development Studio (ROSDS) and MATLAB using comparative method. There are several features that used in this research such as accessibility, computing speed, system requirements, middleware, simulator and motion planning. Based on the features valuation, the results of this research will be obtained that claim that ROSDS is more effective than MATLAB. In order to support the statement above, case study of developing robotic arm simulation using ROSDS was carried out in this reasearch and it worked out well

***Keywords**: RDS, ROS, Comparative, Effectivity, MATLAB*

PENDAHULUAN

Robotika merupakan salah satu bidang yang sangat terpengaruh dalam industri. Penggunaan robot dalam bidang industri akan meningkatkan produktivitas dari industri tersebut. Kelebihan dari penggunaan robot dalam bidang industri ini adalah tingkat presisi yang lebih tinggi dan juga lebih presisi jika dibandingkan dengan manusia. Semakin majunya teknologi akan disertai dengan semakin banyak permintaan dari industri untuk robot dengan fungsi yang lebih kompleks. Robot dengan fungsi yang kompleks tentunya memerlukan algoritma yang kompleks. Penelitian dalam pembuatan robot tentunya akan memerlukan biaya yang tidak sedikit terutama jika melakukan penelitian langsung menggunakan perangkat keras. Salah satu cara untuk meminimalisasikan biaya dalam penelitian tersebut adalah dengan menggunakan simulasi untuk penelitian dari robot tersebut.

Simulasi merupakan salah satu alat penting dalam penelitian di bidang robotika. Simulasi menurut KBBI adalah metode pelatihan yang meragakan sesuatu dalam bentuk tiruan yang mirip dengan keadaan yang sesungguhnya [1]. Simulasi pada robot sendiri adalah proses dari mendesain model, menjalankan model tersebut lalu menganalisa keluarannya. Pembuatan simulasi robot pada umumnya akan memerlukan bantuan dari aplikasi seperti MATLAB, Dymola, Modelica untuk di-*install* di komputer [2]. Tidak semua dari aplikasi simulasi tersebut dapat dengan mudah digunakan di komputer dengan OS Windows.

RDS(*ROS Development Studio*) merupakan salah satu platform simulasi robotika berupa web. Simulasi menggunakan RDS tidak perlu meng-*install* aplikasi apapun, cukup dengan membuka browser peneliti dapat langsung membuat simulasi yang diinginkan di browser tersebut. Proses pembuatan simulasi di dalam RDS dimulai dari mendesain, pemrograman, dan juga pengujian dari simulasi robot tersebut. RDS menggunakan ROS sebagai *middleware* mereka, selain itu RDS juga menggunakan Rviz sebagai alat visualisasi dan juga Gazebo sebagai salah satu dari simulatornya [3].

Penelitian ini bertujuan untuk membandingkan aplikasi simulasi MATLAB dengan ROSDS/RDS (*ROS Development Studio*) untuk mendapatkan aplikasi yang lebih efektif. Efektif yang diinginkan dalam penelitian ini adalah simulasi dapat dibuat lebih sederhana dan juga lebih mudah digunakan dalam pembuatan simulasi dalam bidang robotika.

Berdasarkan dari masalah di atas, pengukuran efektivitas RDS sebagai Simulator Pendukung Pengembangan Algoritma Platform Robotika diusulkan dalam penelitian ini.

TEORI PENDUKUNG

A. ROS

ROS(*Robot Operating System*) dan OS merupakan hal yang berbeda. ROS merupakan *middleware open source* yang berfungsi untuk menghubungkan perangkat keras dengan OS [4]. ROS banyak digunakan oleh peneliti, perusahaan robotika, dan penghobi robotika. ROS terdiri dari kumpulan koleksi *tools*, *libraries*, dan *conventions* yang bertujuan untuk menyederhanakan pembuatan robot di berbagai platform robotika. OS yang umumnya digunakan untuk ROS adalah Linux .

ROS pada awalnya merupakan sebuah proyek besar yang dikembangkan oleh banyak komunitas robotika. Pada pertengahan tahun 2000

STAIR(STanford AI Robot) dan juga program PR (Personal Robots) berhasil membuat prototipe dari ROS. Pada tahun 2007 Willow Garage berhasil mengimplementasikan konsep untuk ROS, didorong dengan bantuan dari banyak peneliti maka terbentuklah inti dari ide ROS dan paket perangkat lunak yang diperlukan [3]. Perangkat lunak tersebut dikembangkan menggunakan lisensi *open-source* sampai akhirnya banyak digunakan oleh komunitas peneliti robotika.

B. Python

Python merupakan salah satu bahasa pemrograman yang populer dan banyak digunakan dalam bidang robotika. Kelebihan menggunakan python yaitu sintaks yang digunakan lebih sederhana dengan mengurangi katakunci yang digunakan untuk membuat sintaks yang benar pada program sehingga memudahkan pengguna untuk memahami program [5]. Selain itu dengan menggunakan python dapat mengakses modul-modul dan juga paket-paket *open source* yang ada. Selain kelebihan python juga memiliki kelemahan, yaitu python tidak pilihan yang baik jika kecepatan menjadi aspek utama pada aplikasi yang dibuat.

C. C++

Bahasa pemrograman C++ menyediakan memori dan komputasi yang menyerupai komputer pada umumnya [6]. C++ bergantung pada manipulasi langsung dari perangkat keras untuk mengirimkan efisiensi tinggi dan programming dengan tingkat yang lebih tinggi yang bergantung kepada tipe *user-defined* yang memberikan model data dan komputasi yang mendekati pandangan manusia terhadap tugas yang dilakukan oleh komputer.

C++ didesain dan diimplementasikan oleh Bjarne Stroustrup di AT&T Bell Laboratorium [6]. Versi inisial C++ bernama "*C with Classes*" pertama digunakan di tahun 1980, mendukung sistem tradisional teknik pemrograman dan abstraksi data. Fasilitas dasar untuk memprogram yang berorientasi pada obyek ditambahkan pada tahun 1983, desain yang berorientasi pada obyek dan teknik memprogram diperkenalkan secara bertahap kedalam komunitas C++.

D. IDE(*Integrated Development Environment*)

IDE merupakan sebuah aplikasi perangkat lunak yang memberikan fasilitas kepada programer komputer untuk pengembangan perangkat lunak. IDE pada umumnya terdiri dari *source code editor*, *build automation tools*, dan debugger digunakan untuk membuat program lain menggunakan alat seperti *editor* dan *compiler*. IDE digunakan oleh para programmer untuk mempermudah proses dari membuat, mengedit dan juga debug dari suatu program[7].

E. Gazebo

Gazebo merupakan sebuah simulator robot yang bersifat *open source* yang dapat digunakan untuk tes algoritma, desain robot, pengujian regresi dan pelatihan sistem kecerdasan buatan (*Artificial Intelligence*) dengan menggunakan skenario yang realistis [8]. Gazebo dapat beroperasi pada OS Linux. Fitur yang terdapat pada Gazebo adalah sebagai berikut:

1. *Dynamic Simulation*
2. *Advanced 3D Graphics*
3. *Sensors and Noise*
4. *Plugins*
5. *Robots Models*

6. *TCP/IP Transport*
7. *Cloud Simulation*
8. *Command Line Tools*

F. MoveIt

MoveIt merupakan sebuah platform robot yang bersifat *open source* yang digunakan untuk melakukan *motion planning* pada robot [9]. Hal yang dapat dilakukan oleh *MoveIt* adalah:

1. *Motion planning*
   Menghasilkan kebebasan *trajectory* tingkat tinggi dalam lingkungan tertentu dan menghindari benda.
2. *Manipulation*
   Menganalisa dan berinteraksi dengan lingkungan yang digunakan.
3. *Inverse Kinematics*
   Menyelesaikan kalkulasi untuk menentukan posisi tiap *joint* pada robot.
4. *Control*
   Dapat menjalankan *joint trajectory* yang berparameter waktu ke perangkat keras melalui *interface* yang umum.
5. *3D Perception*
   Menghubungkan sensor kedalaman dan point cloud dengan *Octomaps*.
6. *Collision Checking*
   Menghindari rintangan menggunakan data geometris, mesh, atau point cloud

G. Efektivitas

Efektivitas merupakan tolak ukur keberhasilan dalam mencapai tujuan yang dapat dinyatakan dengan jelas. [10]. Tujuan yang ingin dicapai dalam penelitian ini adalah dapat membuktikan efektivitas penggunaan ROSDS/RDS sebagai alat pendukung pengembangan algoritma platform robotika dengan MATLAB dengan cara membandingkan studi kasus yang telah dibuat dengan simulasi MATLAB [12].

H. Metode Penelitian Komparasi

Metode penelitian komparasi menurut Lijphart merupakan analisis dari sejumlah kecil kasus, yang memerlukan paling tidak 2 observasi [11]. Metode penelitian komparasi pada penelitian ini membandingkan penelitian menggunakan ROSDS/RDS dengan penelitian yang menggunakan MATLAB dalam hal akesibilitas, *middleware*, bahasa pemrograman, persyaratan sistem, model 3D robot, dan juga simulatornya untuk mendapatkan perangkat mana yang lebih efektif jika digunakan sebagai simulator pendukung dalam pengembangan algoritma platform robotika.

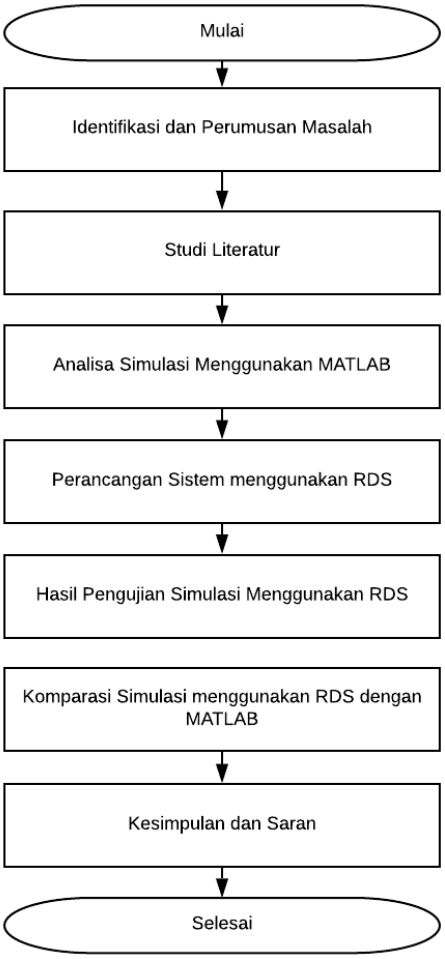
PERANCANGAN SISTEM

A. Perencanaan Pelaksanaan Metode Penelitian

Metode yang digunakan dalam penelitian ini adalah metode komparasi. Dalam penelitian ini metode komparasi digunakan untuk membandingkan simulasi menggunakan RDS dengan simulasi menggunakan aplikasi MATLAB. Hal yang dibandingkan adalah:

1. Aksesibilitas
2. Bahasa pemrograman
3. Persyaratan sistem (*System requirements)*
4. *Middleware*
5. Simulator
6. Motion planning

Diagram alur metode penelitian dapat dilihat pada Gambar 1.



Gambar 1. Diagram alur metode penelitian

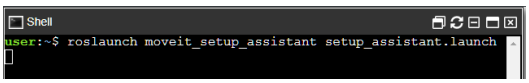
B. Konsep Perancangan

Sistem yang ingin dirancang adalah pengambilan sebuah benda yang posisinya sudah diketahui lalu dipindahkan ke posisi tertentu. Robot yang digunakan pada simulasi ini adalah universal robot. Perancangan projek di RDS ada 2 tipe. Pertama adalah menggunakan Rosject publik yang dapat diakses pada RDS, yang kedua adalah membuat Rosject dari awal dengan mengkloning robot yang ingin digunakan pada Rosject. Ketika perancangan projek selesai akan dilanjutkan dengan mengkonfigurasi paket *MoveIt*. Perancangan terakhir adalah perancangan program yang digunakan untuk mengambil benda.

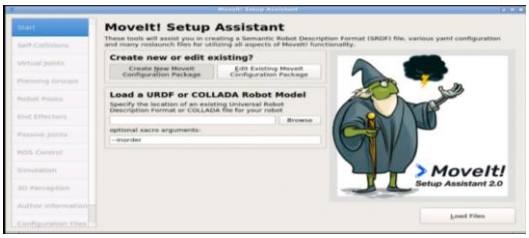
C. Perancangan Projek di RDS dan Mengkonfigurasi *MoveIt*

Perancangan projek pada RDS meliputi pembuatan Rosject lalu dilanjutkan dengan kloning dari universal robot yang akan digunakan atau langsung menggunakan Rosject publik yang ada. Penelitian ini menggunakan Rosject publik yang akan selanjutnya dikonfigurasi ulang. Rosject yang digunakan menggunakan robot universal.

Hal yang selanjutnya perlu dilakukan untuk mengkonfigurasi *MoveIt* adalah membuka rosject yang sudah dipilih lalu masukkan perintah seperti pada gambar 2 kedalam shell lalu buka *graphical tools* untuk dapat mengakses *MoveIt Setup Assistant* seperti pada gambar 3.



Gambar 2. Perintah yang digunakan untuk menampilkan *MoveIt Setup Assistant*



Gambar 3. Tampilan *MoveIt Setup Assistant*

Setelah berhasil menampilkan *MoveIt Setup Assistant* dilanjutkan dengan mengkonfigurasinya. Hal yang pertama dilakukan adalah *load files* URDF yang digunakan dalam simulasi tersebut. Setelah itu masuk kedalam menu *self-collisions* lalu pilih *generate collision matrix*. Jika sudah selesai

dilanjutkan dengan pengaturan *virtual joints*. Setelah pengaturan *virtual joints* dilanjutkan dengan menentukan *planning groups* dan penambahan *joints*. Setelah selesai menambahkan *joints* akan dilanjutkan dengan menambahkan pose untuk robot dan menentukan *end effector* dari robot tersebut. Terakhir akan dilakukan penyimpanan file konfigurasi *MoveIt* tersebut.

Ketika file konfigurasi sudah selesai disimpan akan dilanjutkan dengan pembuatan 2 file yang akan disimpan di dalam folder config yaitu "controllers.yaml", "joint_names.yaml" dan 2 file dengan nama "smart_grasping_sandbox_moveit_controller_manager.launch", "myrobot_planning_execution.launch" yang disimpan pada folder *launch*.

controllers.yaml

```yaml
controller_list:
- name: arm_controller
  action_ns: follow_joint_trajectory
  type: FollowJointTrajectory
  joints:
    - shoulder_pan_joint
    - shoulder_lift_joint
    - elbow_joint
    - wrist_1_joint
    - wrist_2_joint
    - wrist_3_joint
- name: hand_controller
  action_ns: follow_joint_trajectory
  type: FollowJointTrajectory
  joints:
    - H1_P1J1
    - H1_P1J2
    - H1_P1J3
    - H1_P2J1
    - H1_P2J2
    - H1_P2J3
    - H1_P3J1
    - H1_P3J2
    - H1_P3J3
```

Gambar 4. File "controllers.yaml"

```
controller_joint_names: [shoulder_pan_joint, shoulder_lift_joint, elbow_joint, wrist_1_joint, wrist_2_joint, wrist_3_joint]
```

Gambar 5. File "joint_names.yaml"

smart_grasping_sandbox_moveit_controller_manager.launch.xml

```xml
<launch>
  <rosparam file="$(find myrobot_moveit_config)/config/ros_controllers.yaml"/>
  <param name="use_controller_manager" value="false"/>
  <param name="trajectory_execution/execution_duration_monitoring" value="false"/>
  <param name="moveit_controller_manager" value="moveit_simple_controller_manager/MoveItSimpleControllerManager"/>
</launch>
```

Gambar 6. File "smart_grasping_sandbox_moveit_controller_manager.launch"

myrobot_planning_execution.launch

```xml
<launch>

    <rosparam command="load" file="$(find myrobot_moveit_config)/config/joint_names.yaml"/>

    <include file="$(find myrobot_moveit_config)/launch/planning_context.launch">
        <arg name="load_robot_description" value="true"/>
    </include>

    <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher">
        <param name="/use_gui" value="false"/>
        <rosparam param="/source_list">[/joint_states]</rosparam>
    </node>

    <include file="$(find myrobot_moveit_config)/launch/move_group.launch">
        <arg name="publish_monitored_planning_scene" value="true" />
    </include>

    <include file="$(find myrobot_moveit_config)/launch/moveit_rviz.launch">
        <arg name="config" value="true"/>
    </include>

</launch>
```

Gambar 7. File "myrobot_planning_execution.launch"

D. Perancangan Program untuk Mengambil Benda

Hal yang perlu dilakukan agar robot dapat mengambil benda adalah membuat *MoveIt Commander* dengan bahasa python. Pertama yang harus dilakukan adalah membuat paket yang berisikan program agar robot dapat menggenggam bola. Paket tersebut diberi nama "*my_grasping*". File program akan dimasukkan ke dalam src

grasp_demo.py

```python
#! /usr/bin/env python
import sys
import rospy
import moveit_commander
import geometry_msgs.msg

moveit_commander.roscpp_initialize(sys.argv)
rospy.init_node('move_group_python_interface_tutorial', anonymous=True)
robot = moveit_commander.RobotCommander()

arm_group = moveit_commander.MoveGroupCommander("arm")
arm_group.set_named_target("start")
plan1 = arm_group.go()

hand_group = moveit_commander.MoveGroupCommander("hand")
hand_group.set_named_target("open")
plan2 = hand_group.go()

pose_target = geometry_msgs.msg.Pose()
pose_target.orientation.w = 0.5
pose_target.orientation.x = -0.5
pose_target.orientation.y = 0.5
pose_target.orientation.z = -0.5
pose_target.position.x = 0.15
pose_target.position.y = 0.0
pose_target.position.z = 1.25
arm_group.set_pose_target(pose_target)
plan1 = arm_group.go()

pose_target.position.z = 1.125
arm_group.set_pose_target(pose_target)
plan1 = arm_group.go()
```

Gambar 8. Program "grasp_demo.py"

pada *my_grasping* dengan nama file "*grasp_demo.py"*. File tersebut berisi program seperti pada gambar 8 dan gambar 9.

```
pose_target.position.z = 1.125
arm_group.set_pose_target(pose_target)
plan1 = arm_group.go()

hand_group.set_named_target("close")
plan2 = hand_group.go()

pose_target.position.z = 1.5
arm_group.set_pose_target(pose_target)
plan1 = arm_group.go()

pose_target = geometry_msgs.msg.Pose()
pose_target.orientation.w = 0.5
pose_target.orientation.x = -0.5
pose_target.orientation.y = 0.5
pose_target.orientation.z = -0.5
pose_target.position.x = 0.6
pose_target.position.y = -0.2
pose_target.position.z = 1.5
arm_group.set_pose_target(pose_target)
plan1 = arm_group.go()

hand_group = moveit_commander.MoveGroupCommander("hand")
hand_group.set_named_target("open")
plan2 = hand_group.go()

rospy.sleep(5)
moveit_commander.roscpp_shutdown()
```

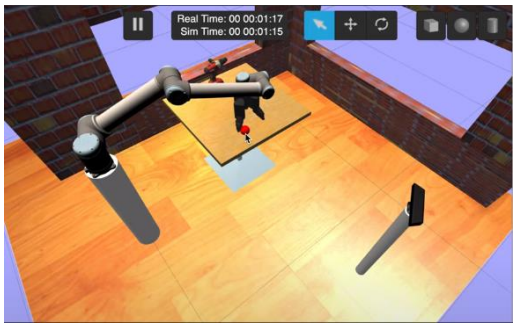
Gambar 9. Program "my_grasping"

E. Hasil Pengujian Simulasi pada ROSDS

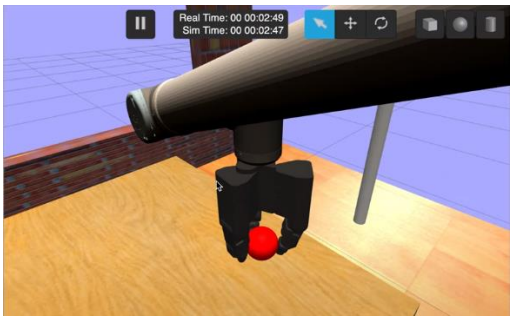
Tahap tahap yang perlu dilakukan untuk menjalankan simulasi pada ROSDS adalah:

1. Buka Rosject
2. Mulai simulasi
3. Buka shell dan masukkan perintah berikut untuk menjalankan konfigurasi *MoveIt*:
   roslaunch myrobot_moveit_config myrobot_planning_execution.launch
4. Masukkan perintah berikut untuk menjalankan program pengambilan benda:
   rosrun my_grasping grasp_demo.py

Setelah memasukan perintah tersebut pertama robot akan bergerak ke posisi "start". Kedua dilanjutkan dengan menggerakkan *gripper* ke posisi "*open"*. Ketiga dilanjutkan dengan perpindahan posisi *end effector* hingga berada di atas bola(gambar 10). Jika *end effector* sudah berada diatas bola dilanjutkan dengan bergeraknya *end effector* tersebut mendekati posisi bola. Selanjutnya *gripper* akan menutup yaitu posisi "close"(gambar 11), lalu *end effector* akan bergerak naik mengangkat bola dan memindahkan ke posisi yang ditentukan pada program. Terakhir *gripper* akan bergerak ke posisi "open" sehingga bola akan jatuh pada posisi yang ditentukan.



Gambar 10. *End effector* berada diatas bola



Gambar 11. *Gripper* berada pada posisi "close"

F. HASIL PENGUJIAN

Hasil penelitian komparasi antara simulasi menggunakan RDS yang dibuat dengan penelitian terdahulu yang menggunakan MATLAB adalah:

Tabel 1. Komparasi simulasi menggunakan RDS dengan simulasi menggunakan MATLAB [12]

ASPEK	RDS	MATLAB
Aksesibilitas	Dapat diakses walaupun	Hanya dapat diakses jika memiliki

	menggunakan perangkat yang berbeda, hanya membutuhkan akses internet.	MATLAB dan juga file simulasinya.
Bahasa pemrograman	python, C++.	bahasa MATLAB.
Persyaratan sistem	OS(Operating System): Windows 7, Windows 8, Windows 8.1, Windows 10 atau yang lebih baru. Prosesor: Intel Pentium 4 atau yang lebih baru. (spesifikasi browser Google Chrome) RAM: 128MB	OS(Operating System): Windows 10(ver, 1803 atau yang lebih tinggi), Windows 7 Service Pack 1, Windows Server 2019, Windows Server 2016. Prosesor: Intel atau AMD x86-64(recommended: Intel atau AMD dengan 4 logic core dan AVX2 instruksi set) RAM: 4GB(recommended: 8GB) Memori: 3.5GB hanya untuk MATLAB saja, 5-8 GB untuk instalasinya. (recommended: menggunakan SSD untuk instalasi) Total instalasi MATLAB secara keseluruhan bisa mencapai 32 GB
Middleware	ROS	tidak ada(bisa menggunakan ROS,LapMap)
Simulator	Gazebo	Simulink / SimMechanics
Motion planning	*MoveIt*	Jacobian

Hal pertama yang dibandingkan antara RDS dengan MATLAB adalah aksesibilitasnya. RDS dapat diakses walaupun menggunakan perangkat yang berbeda sedangkan MATLAB hanya dapat diakses jika memiliki applikasi MATLAB dan juga file simulasinya. Hal kedua adalah bahasa pemrograman yang digunakan pada RDS dan MATLAB. Bahasa pemrograman utama yang digunakan pada RDS sama dengan bahasa pemrograman utama yang digunakan pada ROS yaitu C++ dan juga Python. MATLAB memiliki bahasa pemrogramannya sendiri, selain bahasa pemrogramannya sendiri MATLAB juga dapat memanggil *library* yang menggunakan bahasa pemrograman lain seperti C++, Java, Python, dan Fortran. Hal ketiga adalah persyaratan sistem yang diperlukan untuk dapat menggunakan RDS ataupun MATLAB. RDS hanya memerlukan spesifikasi untuk dapat instalasi browser, penelitian ini menggunakan Google Chrome sebagai browser yang digunakan. MATLAB membutuhkan spesifikasi sistem yang lebih tinggi terutama di bagian memori yang dibutuhkan untuk melakukan instalasi penuh MATLAB dan juga RAM yang dibutuhkan. Hal keempat adalah *middleware* yang digunakan pada RDS adalah ROS yang dapat dibilang sangat optimal jika digunakan untuk mengembangkan algoritma untuk robotika. Simulasi yang MATLAB tidak menggunakan *middleware* tetapi dapat menambahkan *middleware* seperti ROS, LabMap jika diperlukan. Hal kelima adalah simulator yang digunakan pada RDS adalah

Gazebo sedangkan pada MATLAB dapat menggunakan Simulink/SimMechanics tetapi memerlukan biaya yang cukup besar agar dapat menggunakan Simulink ataupun SimMechanics. Hal keenam adalah *motion planning* yang digunakan pada studi kasus menggunakan RDS adalah *MoveIt* sendangkan pada simulasi MATLAB menggunakan algoritma Jacobian.

Dari hasil komparasi diatas dari sisi komputasinya menggunakan RDS akan lebih efektif jika menggunakan komputer dengan kapasitas ram dan memori yang terbatas, selain itu penggunaan Simulink/SimMechanics memerlukan biaya yang cukup besar untuk dapat mengaksesnya. Motion planning yang digunakan pada RDS juga lebih sederhana jika dibandingkan dengan MATLAB dikarenakan motion planning pada MATLAB melibatkan proses kalkulasi sedangkan dengan pada RDS kalkulasi dilakukan oleh M*oveIt* sehingga mempermudah pembuatan simulasi. Penilaian komparasi RDS dengan MATLAB dapat dilihat pada Tabel 2.

Tabel 2. Tabel komparasi penilaian antara RDS dengan MATLAB

Fitur	**RDS**	**MATLAB**
Motion planning	9	7
Persyaratan sistem	9	6
Simulator	9	7
Middleware	10	6
Kecepatan komputasi	7	9
Aksesibilitas	7	6
TOTAL	51	41
Persentase	85%	68,33%

Fitur pertama yang dibandingkan adalah *motion planning* yang digunakan pada RDS dan juga MATLAB. RDS mendapatkan nilai yang lebih tinggi karena pembuatan *motion planning* pada RDS menggunakan *MoveIt* sehingga memudahkan proses pembuatannya jika dibandingkan dengan MATLAB yang menggunakan perhitungan Jacobian. Penilaian pada fitur kedua ini diambil berdasarkan penggunakan komputer dengan spesifikasi rendah sehingga membuat RDS lebih optimal. Fitur ketiga adalah simulator yang digunakan pada kedua aplikasi tersebut. Penilaian pada MATLAB lebih rendah dikarenakan untuk dapat mengakses simulator MATLAB (Simulink/SimMechanics) membutuhkan biaya yang besar. Fitur keempat adalah *middleware* yang digunakan pada RDS memudahkan pembuatan simulasi sedangkat pada MATLAB tidak menggunakan *middleware*. Fitur kelima adalah kecepatan komputasi pada kedua aplikasi. MATLAB memiliki nilai yang lebih tinggi jika dibandingkan dengan RDS karena MATLAB sendiri merupakan aplikasi yang digunakan untuk melakukan perhitungan sehingga membuat proses komputasi pada MATLAB lebih tinggi. Fitur keenam adalah aksesibilitas pada kedua aplikasi tersebut. Penilaian pada RDS agak rendah dikarenakan RDS memerlukan akses internet untuk dapat menjalankan simulasi sedangkan pada MATLAB harus menggunakan perangkat yang memiliki aplikasi MATLAB dan juga file simulasi untuk dapat menjalankan simulasinya.

SIMPULAN DAN SARAN

Berdasarkan perancangan dan pengujian yang telah dilakukan dalam penelitian ini dapat ditarik kesimpulan sebagai berikut:

1. Berdasarkan tabel komparasi penilaian antara RDS dan MATLAB dapat dinyatakan RDS lebih efektif jika dibandingkan dengan MATLAB dengan penilaian

efektivitas RDS 85% dan MATLAB 68,33%.
2. Simulasi menggunakan Simulink ataupun SimMechanics memerlukan biaya yang besar.
3. Pergerakan robot dari posisi awal sampai akhir terbagi menjadi 6 yaitu pergerakan ke posisi "start", membuka *gripper*, *end effector* bergerak hingga berada di atas bola, *end effector* bergerak mendekati benda, menutup *gripper*, lalu bola akan dipindahkan ke posisi sesuai dengan program.

Selain simpulan, saran yang dapat diberikan adalah sebagai berikut:

1. Akses untuk membuka file simulasi pada RDS terkadang memakan waktu yang cukup lama.
2. Menambahkan pendeteksian benda pada simulasi.

DAFTAR PUSTAKA

[1] Pengertian simulasi menurut KBBI, (https://kbbi.web.id/simulasi, diakses 20 Oktober 2020)

[2] Podishetty Naveen Kumar, Y. Shivraj Narayan. "Simulation in Robotics". *Proceedings of the National Conference on "Recent Advances in Manufacturing Engineering & Technology"*. January 10-11, 2011.

[3] Abdul Jalil. "Robot Operating System (ROS) dan Gazebo Sebagai Media Pembelajaran Robot Interaktif". ILKOM Jurnal Ilmiah Volume 10 Nomor 3. Desember, 2018.

[4] Abdul Jalil. "Pemanfaatan Middleware Robot Operating System (ROS) Dalam Menjawab Tantangan Revolusi Industri 4.0". ILKOM Jurnal Ilmiah Volume 11 Nomor 1. April, 2019 .

[5] A Bogdanchikov, M Zhaparov, R Suliyev. "Python to learn programming". *Journal of Physics: Conference Seriers 423 (2013) 012027. 2013*.

[6] Bjarne Stroustrup. "An Overview of the C++ Programming Language". *The Handbook of Object Technology*. 1999.

[7] Mrs. Kavitha S., Ms. Sindhu S. " COMPARISON OF INTEGRATED DEVELOPMENT ENVIRONTMENT (IDE) DEBUGGING TOOLS: ECLIPSE VS NETBEANS". *International Research Journal of Engineering and technology (IRJET) Volume: 02 Issue: 04*. Juli, 2015.

[8] GAZEBO Robot simulation made easy, (http://gazebosim.org/, diakses 3 November 2020)

[9] MoveIt Moving robots into the future, (https://moveit.ros.org/, diakses 3 November 2020)

[10] J. S. McCORMICK. "Effectiveness and efficiency". *Journal of the Royal College of General Practitioners*, Mei 1981.

[11] David Coller. "The Comparative Method". *Political Science: The State of the Discipline II.* Januari 1993.

[12] Dinh Tho Long, To Van Binh, Road Van Hoa, Le Van Anh, Nguyen Van Toan. "Robotic Arm Simulation by using Matlab and Robotics Toolbox fot Industry Application". *SSRG International Journal of Electronics and Communication Engineering* Volume 7 Issue 10. Oktober, 2020.