İ.T.Ü.

Faculty of Computer and Informatics

Computer Engineering

ROBOTICS (BLG 458E)

PROJECT REPORT

Group Name **:** Greenpeace

Group Members **:** Bahadır BAL

: Tamerlan Nusraddinov

: Fahrettin Ay

Instructor **:** Dr. Damien Jade Duff

GARBAGE COLLECTOR ROBOT

ABSTRACT

The earth is all creatures' home that should be clean for the residents. However we are polluting our home irresponsibly. As a result, we all are against with garbage pollution. Therefore, we thought a solution for this problem; using garbage collector robots. Robots are getting more useful and importance in real world. All areas, fields and industries has own robot to profit time, money and yields. There are too many projects that use robotic technology to solve problems. In this project, we designed a robot to collect garbage around it by using automatic grasping from a single-view. Grasping the objects and control the arm of robots are really significant step for collecting the garbage. Thus, the pose and shape of the objects are important to recognize. However, for test section of project, just stick is used for collection. The outline proposed work is as follows: introduction section introduces what is our problem and explained why it is important. In background section introduces what kinds of solutions exist for this problem and how to implement our project and youBot to ROS environment. In design & implementation parts explain how we designed and implement our algorithm for our problem. Evaluation part shows simulation test results and our achievements. With conclusion, all project will explained with last sentences.

Table of Contents

1. INTRODUCTION

Trash on our planet appeared almost simultaneously with the man, but until recently, its existence did not attach much importance. Garbage is something we are willing to pay to get rid of. If we were not willing to pay to get rid of it then it would be a commodity with value. In the middle ages, the garbage was the biggest problem of large cities, but it was not so much as now. In fact, at the future that may be much more serious problem for the world. Because the world is more crowded than the past and the population is increasing. Also, the rate of personal contamination is higher than past. If we do not give weight to this problem, at the future, personal polluting may increase too.

Figure 1-1 Polluting

We can comprehend how the world is getting dirty by looking the table at figure-1. The total solid waste generation and per capita generation between 1960 and 2012 are indicated at the table. The total solid waste generation for the world is about 88 million tons. That value increases until 2005. The maximum amount of waste were generated at 2005, about 253 million tons. Although it decreases from 2005 to 2012, it is not an acceptable value, about 251 million tons. If we look at solid waste generation for per capita in a day, we see another factor why the earth is getting dirty. A person who lived in 1960 would generate 2.68 lbs. garbage in a day where 1 lbs. equals to 0.45 kg. This amount is 4.38 lbs. for who would live in 2012. These values proves that if we do not deal with pollution and go on messing the earth, this will be more than a problem for us.

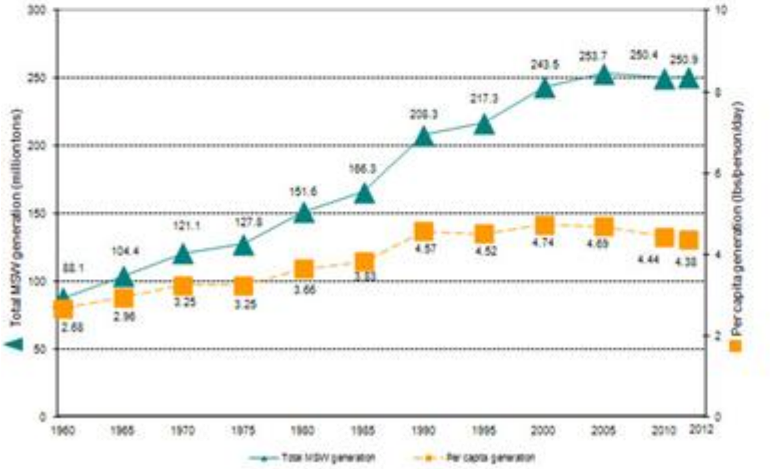


Figure 1-2 MSW Generation Rates between 1960 - 2012

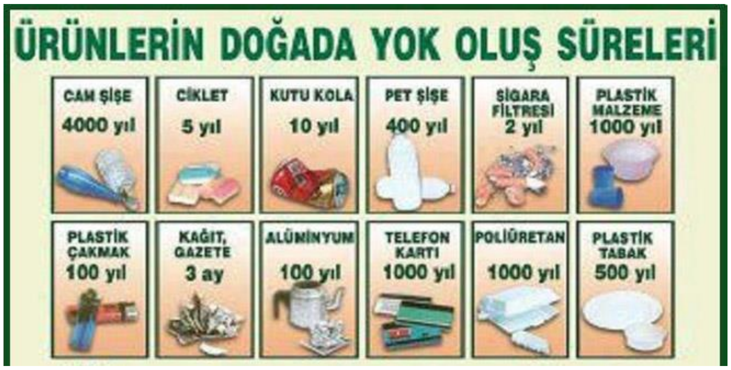


Figure 1-3 some material's extinction period on nature

The world has own magnificent weapon for the pollution, the ground. It can exterminate most kind of solid waste with time. But the time is different for the type of solid waste relatively. To illustrate, a paper is recycled in 3 months by the ground. However, this process takes 4000 years for a glass bottle. Further examples are shown in figure-3. Actually the time for recycling is related with the molecular structure for the solid waste. The organic waste can be recycled easier than chemicals. There is a fact that most of the solid waste for the 21th century earth is chemical based. Thus, the extermination ability of the ground is inadequate for cleaning the world by itself. This means that we should not leave the world alone against the solid pollution that is caused by us.

Actually, many people are employed as dustman for collecting garbage. Moreover, some of us try to keep clean in front of their house, garden or street. Also, there are many recycle bin for collecting garbage that can be recycled. However, we all pollute the earth more than cleaning. As a consequence, that pollution threatens to sociologic and biologic health of the people seriously. Recent physiological research has shown that the presence of garbage in the streets, parks and squares untidy can be bad for mental health and lead to depression. Thus, we need more effective solutions for getting rid of the pollution. But how can we obstruct the garbage pollution? Should more dustmen be employed? Should they spend more and more time for cleaning? Or what?

Figure 1-4 dustman

Robotic is a new technology that is getting improved rapidly day by day. We benefits from robots in many areas, such as industry, medicine and military because, they have many advantages. They are never tired, never say *“NO!”*, do not have salary except their charge and repair outcome, and can work more than a human. Thus, their usage area spreads rapidly.

Figure 1-5 dustbot

We can also use robots for collecting garbage from the environment. In order to do this, we choose most suitable designed robot and program it for making it a garbage collector. Garbage is typically comprised of two important aspects: the material it is made from and the shape the material is in. For example, a plastic bottle is made from PET plastic and is in the shape of a bottle. Thus, the robot we choose should be able to pick the garbage up. Also, it should be able to scan the environment in for detecting garbage, to move through to the garbage, and carry it to a container.

2. BACKGROUND

To get our achievement which is collect to garbage from streets, we used KUKU Youbot and its hardware. We provided our algorithm to this robot and collect the achievement results. Almost all of our mile stones to come up with this idea, are done with good results. In the section to follow, we will introduce KUKA youbot, ROS and installation packet.

2.1) KUKA youBot

In this section, we will introduce KUKA youBot and explain why we use this robot.

KUKA youBot user manual explains that “The KUKA youBot is a mobile manipulator that was primarily developed for education and research in mobile manipulation. KUKA youBot (shown in Figure 2) comes with fully open interfaces and allows the developers to access the system on nearly all levels of hardware control. It further comes with an application programming interface (KUKA youBot API), with interfaces and wrappers for recent robotic frameworks, with an open source simulation in Gazebo[1].

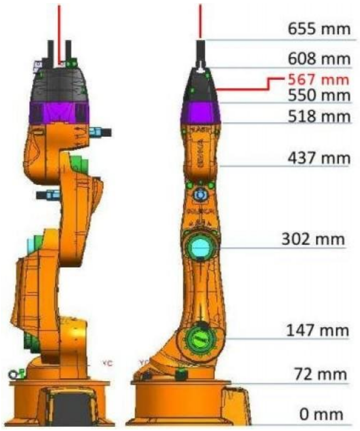


Figure 2-1 KUKA Youbot Arm

As same user manual YouBot has

- An arm with 5 degrees of freedom (0.5 kg payload)
- 2 finger gripper that is attached to the arm
- Omni-directional mobile platform with 4 omni-wheels (20 kg payload)

Arm is suitable to collect garbage around the streets because it moves 3 dimensional direction to get something. Therefore, garbage can be grasped easily by this. But there is a problem that grippers are smaller to hold even coke can. So, in project we implement garbage with stick which is suitable for youbot gripper finger's gap. That means 2 fingers gripper is appropriate for special garbage. Omni-directional mobile platform with 4 omni-wheels help us to find garbage easily around the robot and if robot will stuck somewhere, this feature will be nice to get rid of there. Also, at the same location, it can turn around itself, so take a profit for place.

2.2) ROS (Robot Operating System)

ROS (Robot Operating System) is an open source software framework for robotic development. ROS is to provide a common platform to make the construction of capable robotic applications quicker and easier. Some of the features it provides include hardware abstraction, device drivers, message-passing, and package management [2].

There are three ways that nodes may communicate in a ROS environment:

1. By publishing messages to a topic.

2. By listening to messages published on a topic.

3. By calling a service provided by another node.

2.3) Installation Software on Your Linux

Almost 2 weeks are spent to install youbot with all drivers to our Ubuntu version. Firstly to implement youbot to environment, you have to install ROS as explained above why it is important. We used fuerte environment to implement KUKA youBot. Via this webmail to install ROS

➔ http://wiki.ros.org/fuerte/Installation/Ubuntu

After ROS installment, we have to take as clone github for packet and drivers. Firstly, one document name is ros_stacks should be created, and this documents should download and unzip to this file.

➔ https://github.com/youbot/youbot-ros-pkg/tree/fuerte

➔ https://github.com/ipa320/cob_common/tree/fuerte

After this step, ros-fuerte-pr2-controllers should be installed for ROS. Therefore, in command line next command will be send.

```
➔ $ sudo apt-get install ros-electric-pr2-controllers # required for trajectory_msgs
➔ cd ~/ros_stacks
```

Now, all drivers should be made after this steps. Firstly, we have to show to ROS, ros_stacks is our ROS_PATH. Then, Youbot_Driver, Youbot_Description and our project files should be made by command as “rosmake ….” .

After all steps, our project will implement on ROS and we can run and analyze on Gazebo. [3]

2.4) Literature Background

➔ (Saxena et al., 2008) developed a learning algorithm that predicts the grasp position of an object directly as a function of its image. Their algorithm focuses on the task grasping points that are trained with labelled synthetic images of a different number of objects.

➔ (Kragic & Bjorkman, 2006) developed a vision-guided grasping system. Their approach was based on integrated monocular and binocular cues from five cameras to provide robust 3D object information.

➔ (Wang & Jiang, 2005) developed a framework of automatic grasping of unknown objects by using a laser-range scanner and simulation environment.

➔ (Richtsfeld & Zillich, 2008) published a method to calculate possible grasping points for unknown objects with the help pf the flat top surfaces of the objects based on a laser-range scanner system. However, there exist different approaches for grasping for grasping quasi planar objects.

➔ (Goldfeder et al., 2007) presented a grasp planner which considers the full range of parameters of a real hand and arbitrary object, including physical and material properties as well as environmental obstacles and forces.

3. DESIGN AND IMPLEMENTATION

Our project design is divided into three parts:

1. Move to the garbage location
2. Find the garbage and pick it up
3. Move to the garbage container location and drop down the garbage

Finding the garbage totally automatically was difficult, that is why we made it partially automatically. It means that robot is directed to the near location of the garbage and then when robot is approximately 1 meter close to the location it starts searching by laser sensor. The system was designed like if garbage (stick in our case, will be explained latter) is within sight of 2 meters. In other words, laser sensors maximum range can measure is 4 meters; however, it sees the stick within 2 meters in radius.

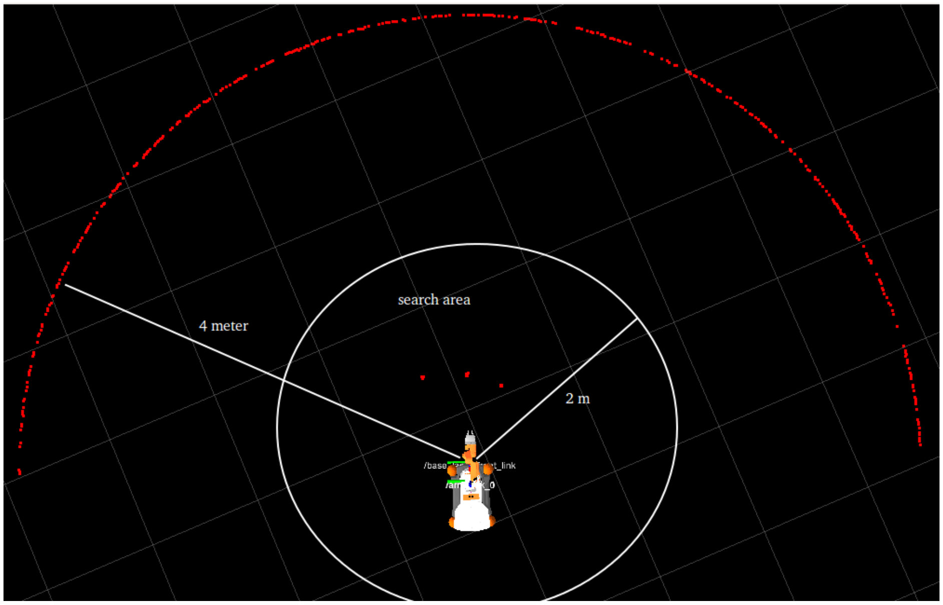


Figure 3-1: Laser scan search range

For robot arm we define 4 arm positions:

- First is when robots arm is getting down and gripper fingers are opened, in addition it does not close the laser sensor.
- Second position the same as first except its grippers fingers are closed.

- Third is when arm is picked up and gripper's fingers are closed.
- Fourth is as third except grippers fingers are opened.

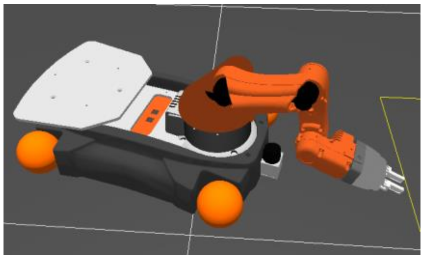


Figure 3-3: First arm position

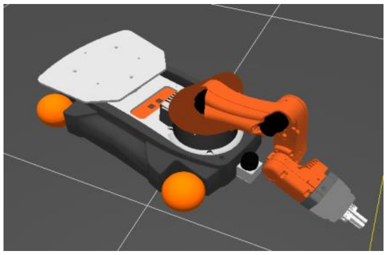


Figure 3-3: Second arm position

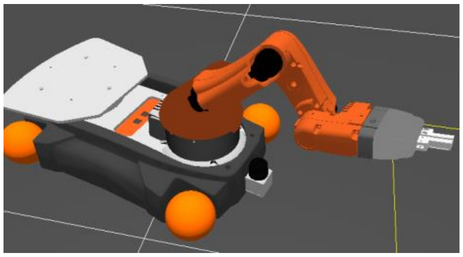


Figure 3-4: Third arm position

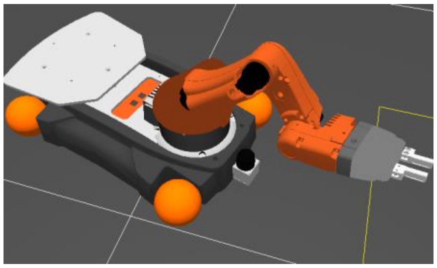


Figure 3-5: Fourth arm position

Initially our project was planned to grasp the coca-can model in Gazebo but we did not consider that the models size is bigger than distance between gripper fingers. That is why we create our own model like a stick.

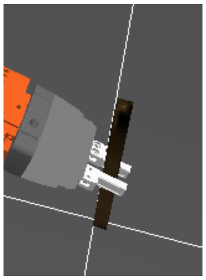


Figure 3-6: Stick as a garbage

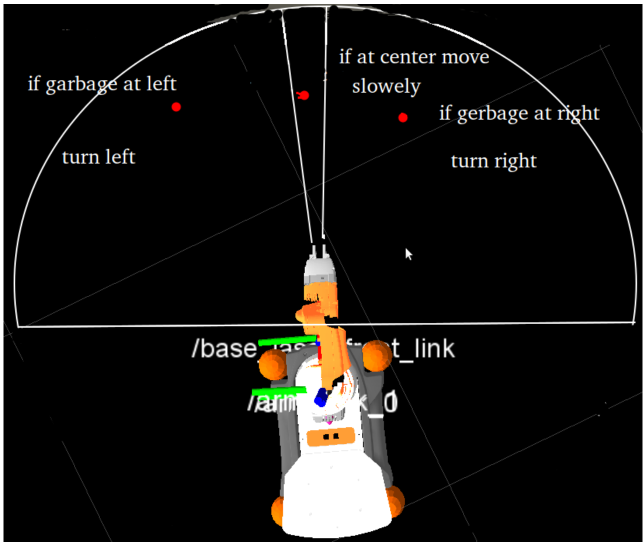


Figure 3-7: Robot adjustment

To find exact location of stick we used laser scan sensor in the following way. From picture we can see that to locate the stick to the center of robot we turn the robot to the right if the is at right and to left if at left. While distance between robot and stick is getting closer velocity of robot is decreasing to increase the accuracy of grasping.

After we get the stick between grippers fingers algorithm sets the arms position to second position. Actually it just grasps it and then it sets arm position to three which is picked up position.

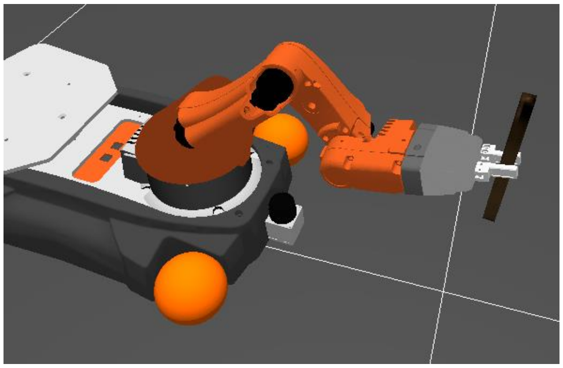


Figure3- 8: Pick up after grasping

After garbage is picked up robot directs to the garbage container location and moves to that direction. While it is close to container it drops the stick (garbage) to down and then robot gets the next garbage location and is directed to that location.

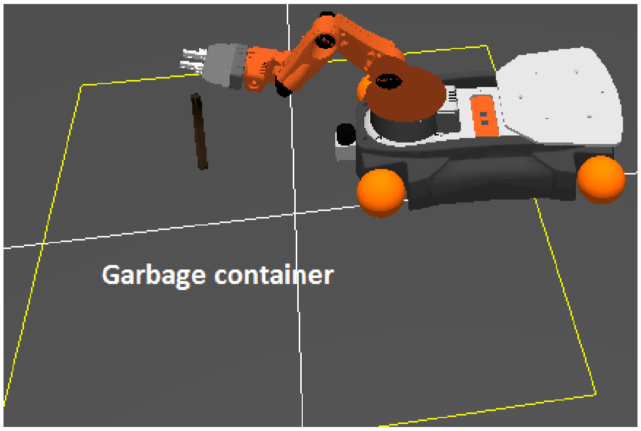


Figure 3-9: Garbage container

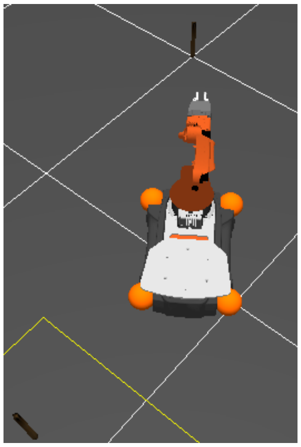


Figure 3-10: Next garbage direction

When robot get closed to stick it again sets its arm to first position and try to find the exact location of garbage.

Technical details

- Arm position and its gripper

To set arm positions in ROS we need to publish a message ***brics_actuator::JointPositions*** to ***"/arm_1/arm_controller/position_command"*** topic. To open or close the gripper's fingers we need to publish a message ***brics_actuator::JointPositions*** to ***"/arm_1/gripper_controller/position_command"***topic.

- How to know whether gripper is closed or not

In order to make this we need **tf** listener which transforms ***"/gripper_finger_link_l"*** and ***"/gripper_finger_link_r"*** frames to ***"/gripper_palm_link"*** frame and then we get coordinates of left and right fingers with respect to ***gripper_palm_link*** which allows us to get information about gripper independent of its location in 3D coordinate. Simply we subtract **x** coordinates of both fingers and take absolute value of it. If this value is 0 it means that gripper is closed, otherwise is opened.

All nodes and topics can be shown with **rxgraph** tool.

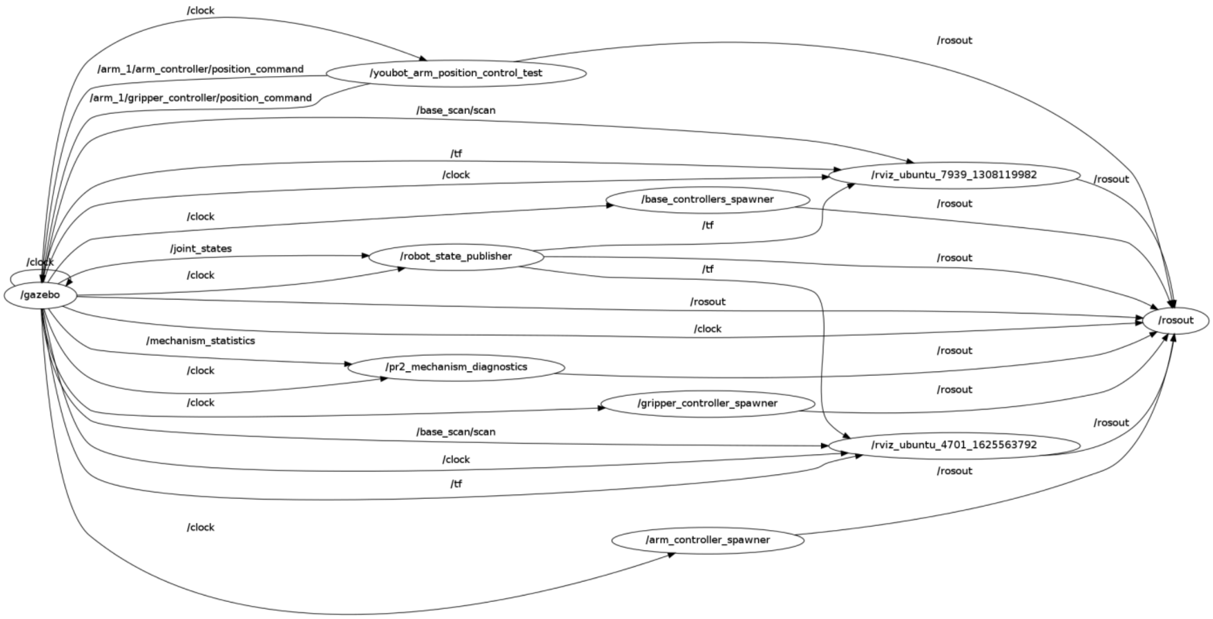


Figure 3-11: Graph of nodes and topics

This tool allows us to see which nodes are using which topics. From this graph we can easily understand all connections between nodes.

- Check if garbage is in front of and very close to robot

This problem is easy to check by laser scan sensor. After subscribing the **/laser_scan** topic we get a message which include the vector that contains the ranges. Maximum range is 4 meters. While robot is getting close to the garbage it always checks if garbage is in front or not. If it is sure that garbage is in front and between the gripper it grasps it.

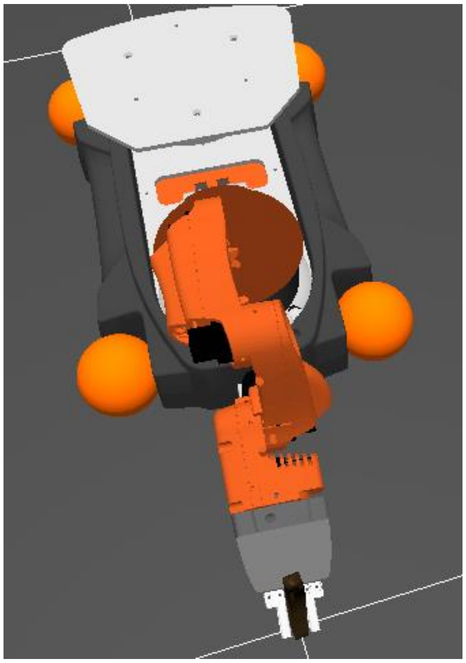


Figure 3-12: Garbage is in front of robot and between gripper's fingers

- Code design

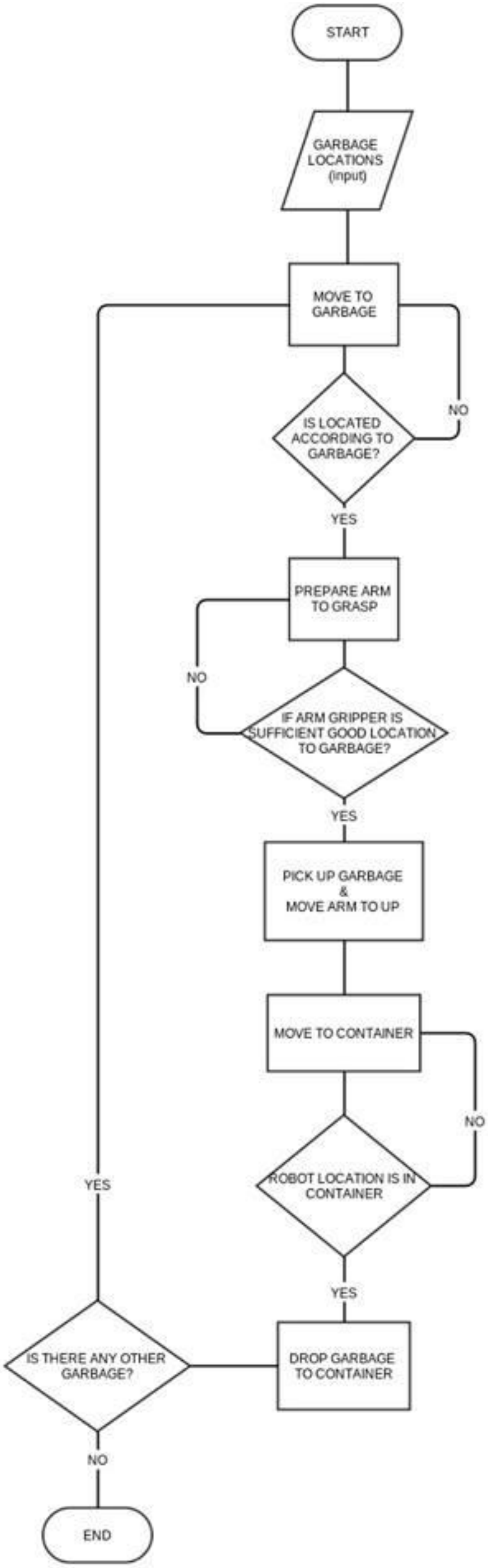
All code was written in object oriented programming way. All variables are defined in private and methods in public.

- Algorithm and its Explanation

```
if(!obj_coordinates.empty() && !isCloseToObj){
        cout << "[MOVE TO GARBAGE] "<<endl;
        waypoint_x=obj_coordinates.front().first; // garbage coordinates are taken
        waypoint_y=obj_coordinates.front().second;
        getTargetDirection();
        if(dist < 1 ){ // if distance < 1 switch to adjustRobotPosition() function
                if(isInFront(msg)){
                        isCloseToObj=true;
                        vel_msg.linear.x = 0.0;

                        vel_msg.angular.z = 0.0;
                        pub.publish(vel_msg);
                        setArmPos(1);
                        cout << "[SWITCH] "<<endl;
                }else{
                        isCloseToObj=false;
                        vel_msg.linear.x = 0.0;
                        vel_msg.angular.z = 1.0;
                        pub.publish(vel_msg);
                }
        }
}else if (isCloseToObj && !objectTaken){
        cout << "[ADJUST ROBOT POSITION] "<<endl;
        adjustRobotPosition(msg);// garbage will be between the gripper's fingers
        if(objectTaken){
                vel_msg.linear.x = 0.0;

                vel_msg.angular.z = 0.0;
        }
}else if(objectTaken){
        cout << "[MOVE TO GARBAGE CONTAINER] "<<endl;
        waypoint_x=0;
        waypoint_y=0;
        getTargetDirection();
        if(dist < 0.3 ){
                isCloseToObj=false;
                vel_msg.linear.x = 0.0;

                vel_msg.angular.z = 0.0;
                pub.publish(vel_msg);
                setArmPos(4); // drops down
                objectTaken=false;
                obj_coordinates.pop_front(); //pops front to get next coord.
                cout << "[CAME TO GARBAGE CONTAINER] "<<endl;

        }
}else{
        cout << "[WORK IS DONE] "<<endl;
        if(!isDone){
                setArmPos(5);
                isDone=true;
        }
}
```

First we put the coordinates into the **deque** and define **bool** variables **isCloseToObj=false, objectTaken=false.** After that we check if that deque is not empty and not close to the object then it takes from front of deque coordinates and directs the robot to that location. If distance is smaller than 1 meter it checks if the object in the front of the robot. If it is it stops, sets arm position to 1 and makes **isCloseToObj=true**, otherwise robot turns around and searches the object. Then it gets into the other functions. If robot close to object and object not taken robot is adjusting itself so that the garbage will be between the gripper's fingers. If object is taken it sets the **objectTaken=true** then stops and picks it up. Getting into another third part of our algorithm it checks if object is taken if so it moves to the garbage container and drops the garbage down and sets the variables **isCloseToObj=false, objectTaken=false.** This loop repeats until deque will be empty.

- Flow chart


START
GARBAGE LOCATIONS (input)
MOVE TO GARBAGE
IS LOCATED ACCORDING TO GARBAGE?
NO
YES
PREPARE ARM TO GRASP
NO
IF ARM GRIPPER IS SUFFICIENT GOOD LOCATION TO GARBAGE?
YES
PICK UP GARBAGE & MOVE ARM TO UP
MOVE TO CONTAINER
NO
ROBOT LOCATION IS IN CONTAINER
YES
YES
IS THERE ANY OTHER GARBAGE?
DROP GARBAGE TO CONTAINER
NO
END

4. EVALUATION

In this section, we will introduce how many percent our project made success and what kind of problems we encountered. When we implement our algorithm to Youbot and run it as explained above, we had just sticks around the robot. Therefore, we could make tests just on sticks because of fingers' gap of gripper. As graph is mentioned below, we got 3 steps of simulation test time. Each same sticks number are tried 20 times and get results. So,

1) Firstly, we run project on 3 sticks,
    a. Just percentage of hold 1 stick for simulation is 10%.
    b. Percentage of hold 2 stick for simulation is 15%.
    c. Percentage of hold 3 stick for simulation is 75%.
2) Secondly, we increased sticks number from 3 to 4.
    a. Just percentage of hold 1 stick for simulation is 5%.
    b. Percentage of hold 2 stick for simulation is 10%.
    c. Percentage of hold 3 stick for simulation is 15%.
    d. Percentage of hold 4 stick for simulation is 70%.
3) Thirdly and final, we tried 5 sticks at same time to collect them;
    a. Just percentage of hold 1 stick for simulation is 5%.
    b. Percentage of hold 2 stick for simulation is 6%.
    c. Percentage of hold 2 stick for simulation is 9%.
    d. Percentage of hold 2 stick for simulation is 25%.
    e. Percentage of hold 2 stick for simulation is 55%.

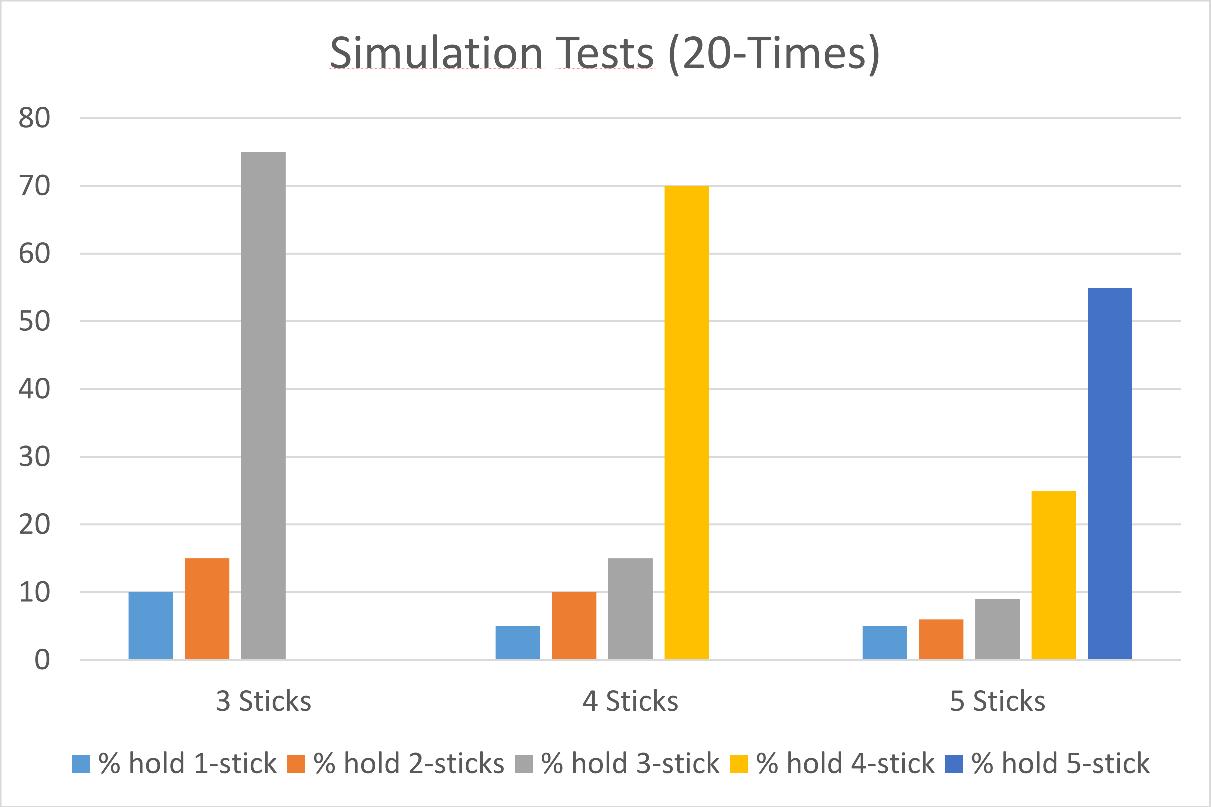


Figure 4 - Simulation Results

As result of also graphs, we got success results. However, sometimes we didn't get all sticks because of some exceptions. We will explain some of them;

- ➔ Sometimes Robot is spinning around when try to find garbage
- ➔ Gripper is stuck near to garbage because it cannot get out in loop if it holds or not.
- ➔ Sometimes, stick's friction will not enough to hold stick and while carrying to container, stick is sliding and dropped on way.
- ➔ While working on gazebo in long term, sometimes real odometry and gazebo frame is not shown good and we need this time to reset all of them.

5. CONCLUSION

Eventually, increasing solid waste pollution that will may cause destructive effects living creatures' habitat is one of the serious problem for the people. This problem can be solved or at least reduced by robotic technology. In order to show this, a garbage collector robot was implemented by us. The technique that was used simple to understand, but hard to implement. However, it was succeed by group work. Also, we utilized from very useful tools.

The project was developed in Robotic Operation System on Ubuntu. Also, Gazebo environment was used for the simulation. KUKA youBot was preferred as a robot because of its attractive properties. It can move with its omni – wheels, it has an arm with 5 degrees of freedom, it can pick up the garbage with its 2 – finger gripper. Also, external laser sensor was integrated it by us for making it talented about detecting garbage from the environment. Consequently, a garbage collector robot was ready for working in environment.

However, our garbage collector robot cannot work on rude platform. Also, it cannot pick up garbage that is not like a stick. Additionally, the weight of the garbage is another constraint for our robot. Therefore, for the future work, we will advance the project for resolving these constraints for the robot. After this, there may be *dustbots* all over the world that help people to get rid of the garbage at the near future.

6. REFERENCES

[1] *Locomotec, "KUKA youBot User Manual," 6 October 2012. [Online]. Available: http://youbot-store.com/downloads/KUKA-youBot_UserManual.pdf. [Accessed 14 May 2014].*

[2] "Grasping unknown objects based on 3d model reconstruction. Proceedings of International Conference on Advanced Intelligent Mechatronics/ ASME", Wang, B., Jiang, L. (2005)

[3] http://www.youbot-store.com/youbot-developers/software/frameworks/ros-wrapper-for-kuka-youbot-api