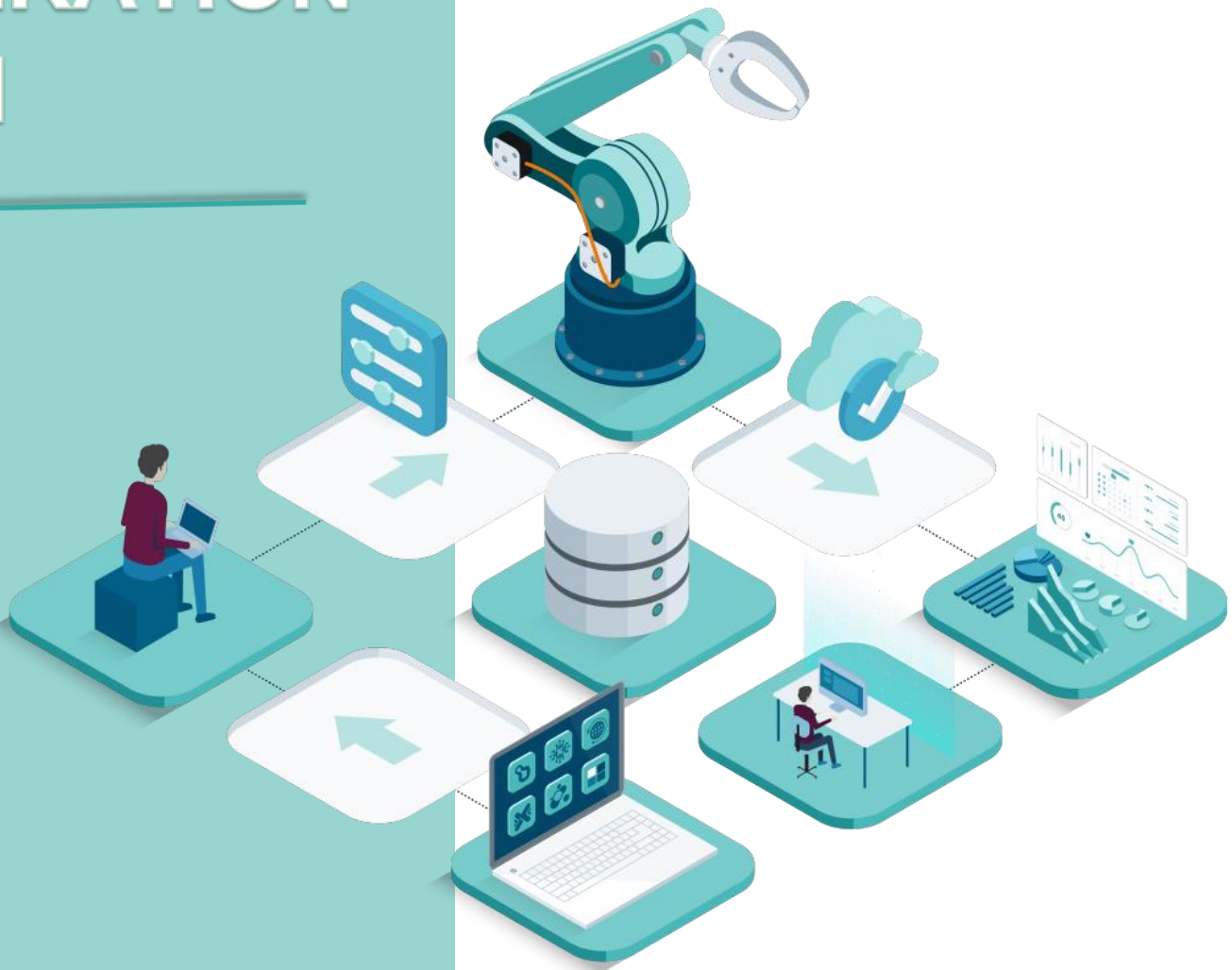


INDUSTRIAL OPERATION 2021



JULY 8

Authored by: Sarah Ali Alshahrani

Industrial engineering & System
Department

Table of Contents

1-Operations.....	3
1.1- Fighting Robot Project dimensions.....	7
1.2- Working Area dimensions.....	8
1.3- Operation Rules.....	9
1.4- Describe the control Board.....	10
1.5- Details of Technical Operations	10
2- Testing.....	20
3- Tolerance	20
4- User Manual.....	21
4.1- Overview.....	21
4.2- Usage.....	21
4.3- Who should read this manual.....	21
4.4- Prerequisites.....	21
4.5- Safety Guidelines.....	21
4.6-Handling & personal Safety.....	23
4-7- General	24
4.8- Working Area.....	24
5- Warranty& Liabilities.....	24

1- Operations

Robot arm & its base structure

Objective:

The main goal of this section to help the reader to rebuild this robot arm easily and with the whole understanding of the robot arm function.

Equipment:

This robot arm is containing of these Parts:

1. Base
2. First arm
3. Second arm
4. Waist
5. End effector

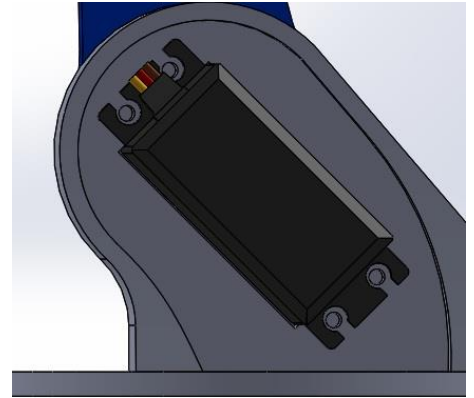
Methods:

A. Building robot arm Using 3D design software:

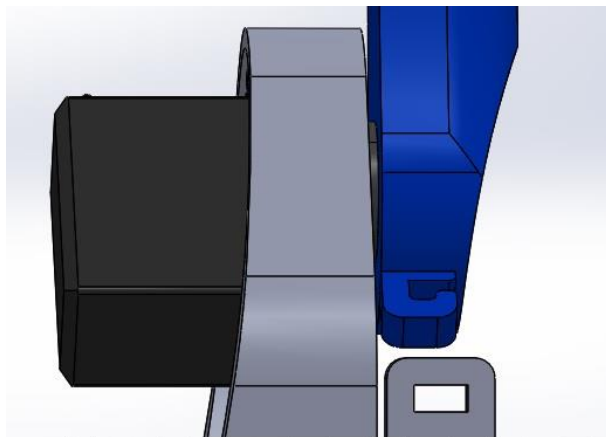
1. Bring the base and the waist and to link them easily you should click on the order constrain then choose mate try to link both of the using the edge of the waist and connect it with the edge of the base.
2. Connect the arm from the side of the joint with the waist using the order constrain then choose mate.
3. Connect the second arm with the first arm from the round side using the order constrain then choose mate.
4. Connect the end effector with second arm from the rectangular side using the order constrain then choose mate.

B. Building robot arm in headquarters:

1. You will need to secure the base to the board of the robot base using M3 screws and then you will take the waist and link it with the base.
2. You will take the waist and link it with the first arm but to be able to link them at the beginning we will need to the waist servo. We simply put it in place and secure it to the 3D printed part using self-tapping screws, two screws up and two down.

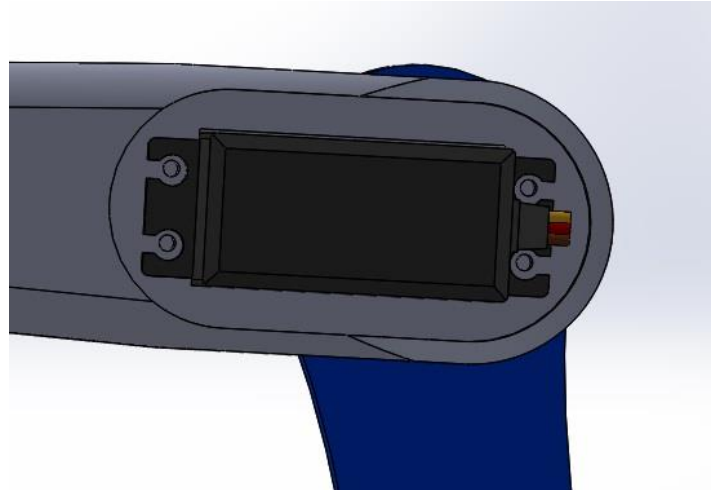


3. From the other side of the waist the round horn is secured to the waist part using the self-tapping screws that come as accessories with the servo, and then the round horn is secured to servo shaft using the appropriate bolts that also come with the servo.

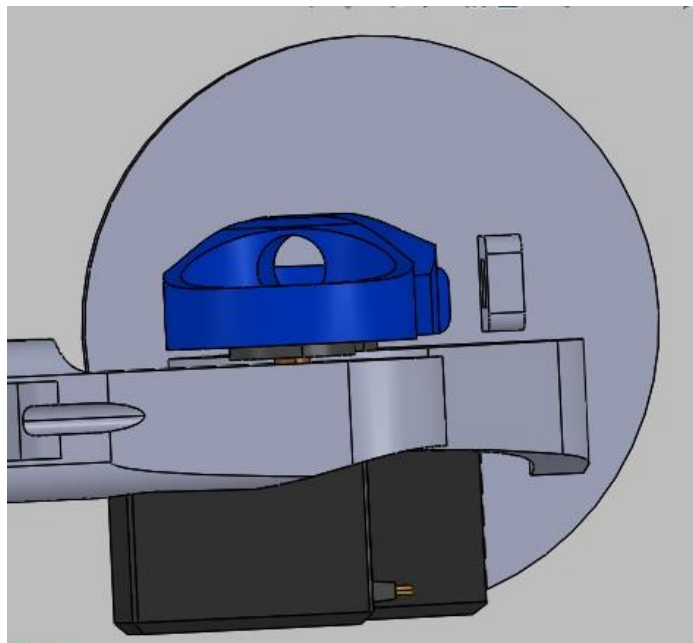


4. Now you can link the first arm with waist since the round horn goes on the next part, and then the two parts are secured to each other using a bolt on the output shaft of the servo. We should note that before securing the parts, we need to make sure that the part has the full range of motion. So, add a rubber band to the waist joint so that it gives a little bit help to the servo, because this servo carries the weight of the rest of the arm as well as the payload.

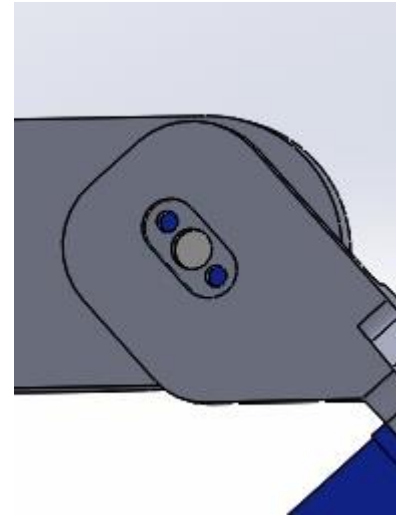
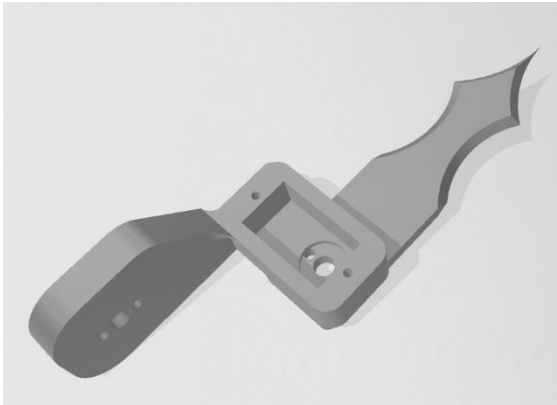
-
5. Now you will take the second arm and link it with the first arm but to be able to link them you will need to connect the second arm servo. you simply put it in place and secure it to the 3D printed part using self-tapping screws, two screws up and two down.



6. From the other side of the second arm the round horn is secured to the second arm part using the self-tapping screws that also come as accessories with the servo, and then the round horn is secured to servo shaft using the appropriate bolts that also come with the servo.
7. Now you can link the first arm with the second arm since the round horn goes on the next part, and then the two parts are secured to each other using a bolt on the output shaft of the servo.



-
8. Now we need to assemble the end effector. First, we attach a custom designed geared link. We pair this link with another geared link on the other side, which is secured using M3 bolt and nuts. all other links are connected using M3 bolts and nuts. Now the robot arm is ready.



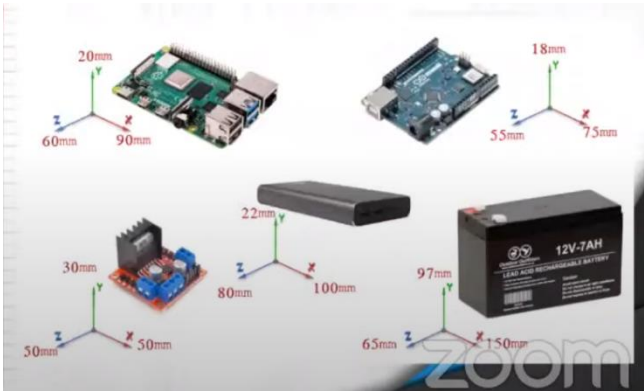
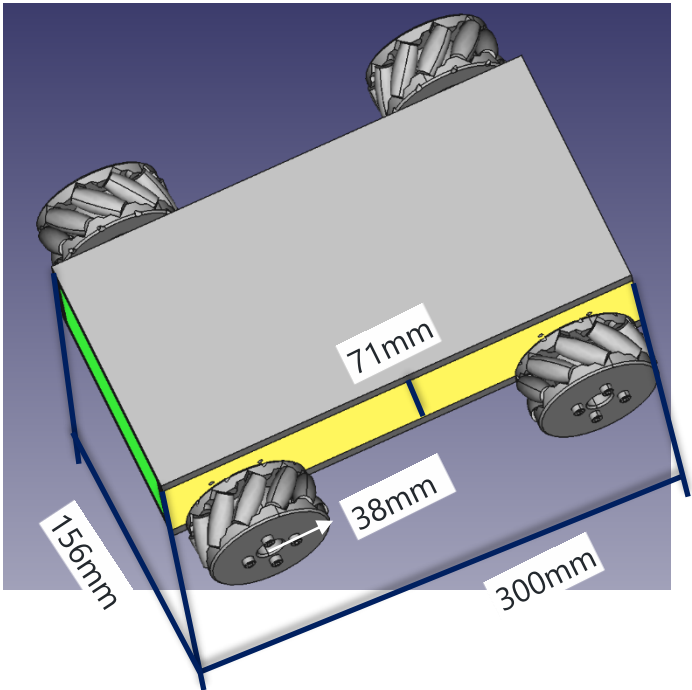
You might need to use a zip-tie to secure all the wires together (depends on the way you want to connect them with the Arduino board).

Fighting Robot Project Dimensions

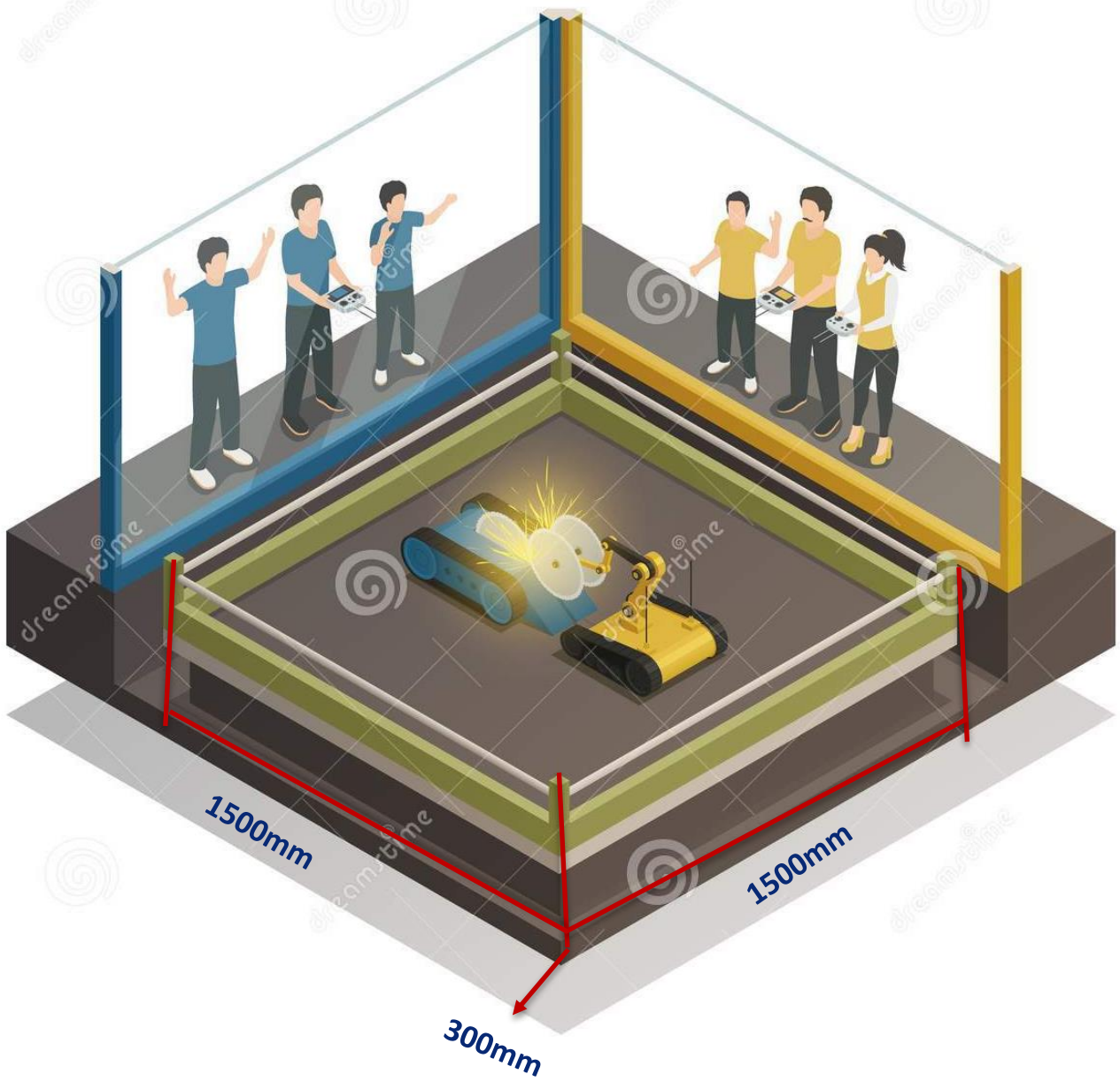


No	Parts	Dimensions L*W*H (mm)
1	Circular base	The dimensions of all the parts together is
2	First arm	
3	Second arm	
4	Waist	180*40*23
5	End effector	
	• Pin	89*20*4
	• Holder	62*38*11

Consider the area of control circuits in the design of the robot base:



Working Area Dimensions



Operation Rules

How Matches Work:

- Robots will begin a match each facing opposite far 500cm directions inside a square area.
- Matches will last a maximum of two minutes.
- If robots become tangled or deadlocked for 20 seconds, officials will stop the clock and return robots to their starting positions.
- All teams will control their robot via web page .

How to Win a Match:

- If the robot pop the balloon of the other, officials will award a win to the this team.
- If a robot drives out of the ring, robots will return to their starting position.
- If a robot becomes disabled and cannot move, officials will award a win to the opposing team.
- If both robots fall out of a ring at the same time, robots will return to their starting position.
- All rulings by officials are final.

Points:

- 2 points for winning a match
- 1 point for a draw

How to Win the Tournament:

The team with the most points at the end of the tournament will be declared the winner.

In the event of a tie, teams will face each other again

Describe the control Board

The brain of this robot platform is a web page contain joy stick which controls each wheel and parts of robotic arm individually.

Details of Technical Operations

IOT :

Robotic-Arm-Control-Panel

Started on a project of creating a robotic arm and controlling it online. On the controlling and programming side of things the IoT specialists decided to choose Python as programming language for this task and MySQL to create the database. They started with the UI first then created the database and after that, connected the two together.

User-Interface

They started by importing the tkinter library in Python to create and customize UI. First created my window and chose the title, the size and the color. Then, started on creating the six sliders , will be using in this UI while also choosing their colors and sizes and also their place on the UI, while also assigning a label which contains the sliders name on its left side. They repeated this step for each slider until they were all prepared and ready Then, added the buttons "Save" and "Run" but currently they have no function, this will be added later when connect it to the database. Due to the limitations of python programming, they couldn't customize UI in an fashionable way so the UI we created is basic.

Database

First started creating database by launching MySQL and creating a server inside of it. Then connected SQL using the sql.connector function and defined the host, username and password. After that, created the database after defining the connection with SQL as cursor and named the database robot. Then commented the creation command and went back to the connection and added the database and its name there. Added a command to show me the database to ensure its existence after that, commented the command so it won't overlap So now cursor also has the database defined in it and we can create a table and place it inside the database which we did and named the table motor and added 3 columns inside of it "name", "degree", "Status". lastly in order to ensure our work is complete they gave the "SHOW TABLES" Command to ensure the

table was correctly created. "NOTE: because they took the picture, they finished all three subtasks they had to include the array from the next task in the picture that was uploaded within the database file as couldn't return the database's table to its original form after creation without some complications."

Connecting UI to DB

Now time to connect them both. Going back to the User interface code and added 2 new functions, The first was called Values which contains an array that will take the values of the sliders and store them in the database according to their names, they also connected the database using the same method in the Database code, this function will only run if the "Save" button was pressed. The second function was added to the "Run" button command which will turn the default "OFF" status for each slider to "ON" upon being clicked. They uploaded a picture that shows the changed values of the database after running the code.

Robotic-arm-Base

Just like the previous task we are using Python for this task where we are going to create the base's UI and also its database.

Base-UI

After importing the tkinter library from we choose the size of the window, background color and title. After this they created the button which are supposed to direct the base movement in all 4 directions and a fifth button to stop it from moving. Currently the button has no function because we haven't connected it with the database.

Base-Database

Connecting to SQL using the `sql.connector` function and defined the host, username and password. After that they created my database after defining the connection with MySQL as cursor and I named the database BaseDB. They commented the creation command and went back to the connection and added the database and its name there. Adding a command to show me the database to ensure its existence after that they commented the command so it won't overlap. So now cursor also has the database defined in it and we can create a table and place it inside the database which we did and named the table Base and added 2 columns inside of it "ID" for the number of the command that we are giving, and "Status" to indicate the direction the base is moving in. Lastly in order to ensure our work is complete they gave the "SHOW TABLES" Command to ensure the table was correctly created. including a picture of the database inside the file but since they haven't connected it with the UI it is empty.

AI

We used the codes in smart methods github account

```
$ cd ~/catkin_ws/src
```

```
$ sudo apt install git
```

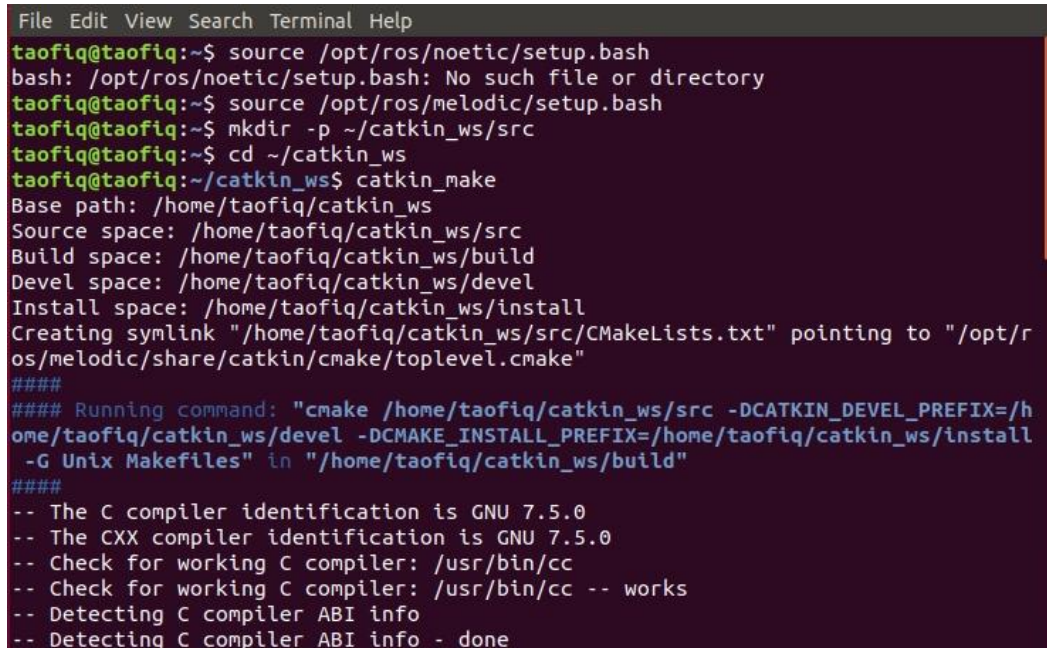
```
$ git clone https://github.com/smart-methods/arduino\_robot\_arm
```

Creating a workspace for catkin by using this codes

```
$ mkdir -p ~/catkin_ws/src
```

```
$ cd ~/catkin_ws/
```

```
$ catkin_make
```

A terminal window with a dark background and light-colored text. The terminal shows the execution of catkin_make in a workspace. It starts with the user sourcing ROS setup files (noetic and melodic), creating the workspace directory, and then running catkin_make. The output shows the base, source, build, devel, and install spaces, the creation of a symlink for CMakeLists.txt, and the execution of cmake. The cmake output shows the detection of the C compiler (GNU 7.5.0) and the successful configuration of the build system.

```
File Edit View Search Terminal Help
taofiq@taofiq:~$ source /opt/ros/noetic/setup.bash
bash: /opt/ros/noetic/setup.bash: No such file or directory
taofiq@taofiq:~$ source /opt/ros/melodic/setup.bash
taofiq@taofiq:~$ mkdir -p ~/catkin_ws/src
taofiq@taofiq:~$ cd ~/catkin_ws
taofiq@taofiq:~/catkin_ws$ catkin_make
Base path: /home/taofiq/catkin_ws
Source space: /home/taofiq/catkin_ws/src
Build space: /home/taofiq/catkin_ws/build
Devel space: /home/taofiq/catkin_ws/devel
Install space: /home/taofiq/catkin_ws/install
Creating symlink "/home/taofiq/catkin_ws/src/CMakeLists.txt" pointing to "/opt/r
os/melodic/share/catkin/cmake/toplevel.cmake"
####
#### Running command: "cmake /home/taofiq/catkin_ws/src -DCATKIN_DEVEL_PREFIX=/h
ome/taofiq/catkin_ws/devel -DCMAKE_INSTALL_PREFIX=/home/taofiq/catkin_ws/install
-G Unix Makefiles" in "/home/taofiq/catkin_ws/build"
####
-- The C compiler identification is GNU 7.5.0
-- The CXX compiler identification is GNU 7.5.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
```

Then we make sure that these functions exist or install them again using the following codes:

```
$ cd ~/catkin_ws
```

```
$ rosdep install --from-paths src --ignore-src -r -y
```

```
$ sudo apt-get install ros-melodic-moveit
```

\$ sudo apt-get install ros-melodic-joint-state-publisher ros-melodic-joint-state-publisher-gui

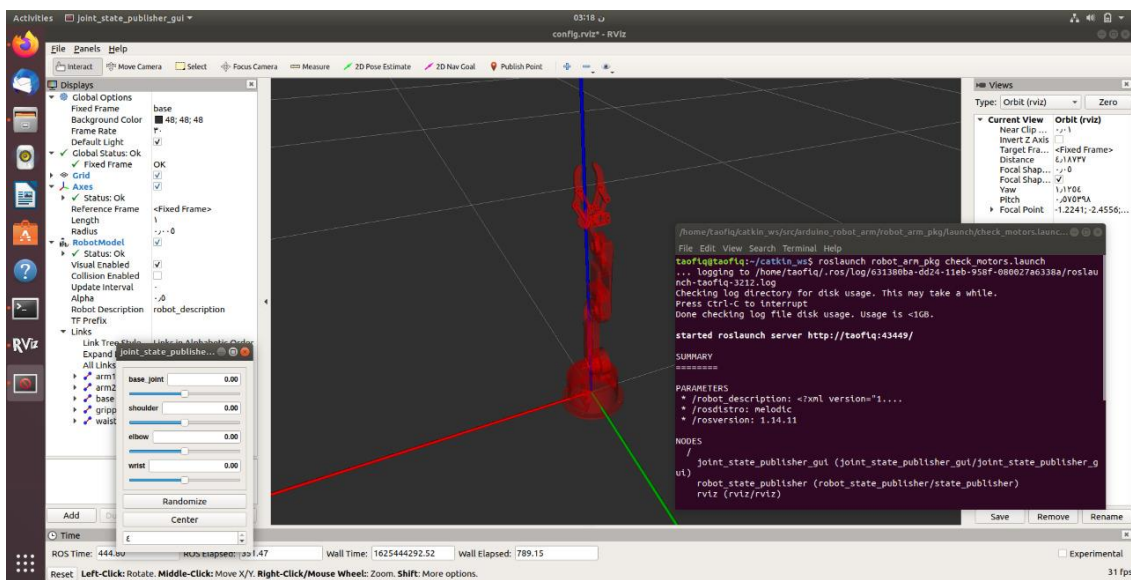
\$ sudo apt-get install ros-melodic-gazebo-ros-control joint-state-publisher

\$ sudo apt-get install ros-melodic-ros-controllers ros-melodic-ros-control

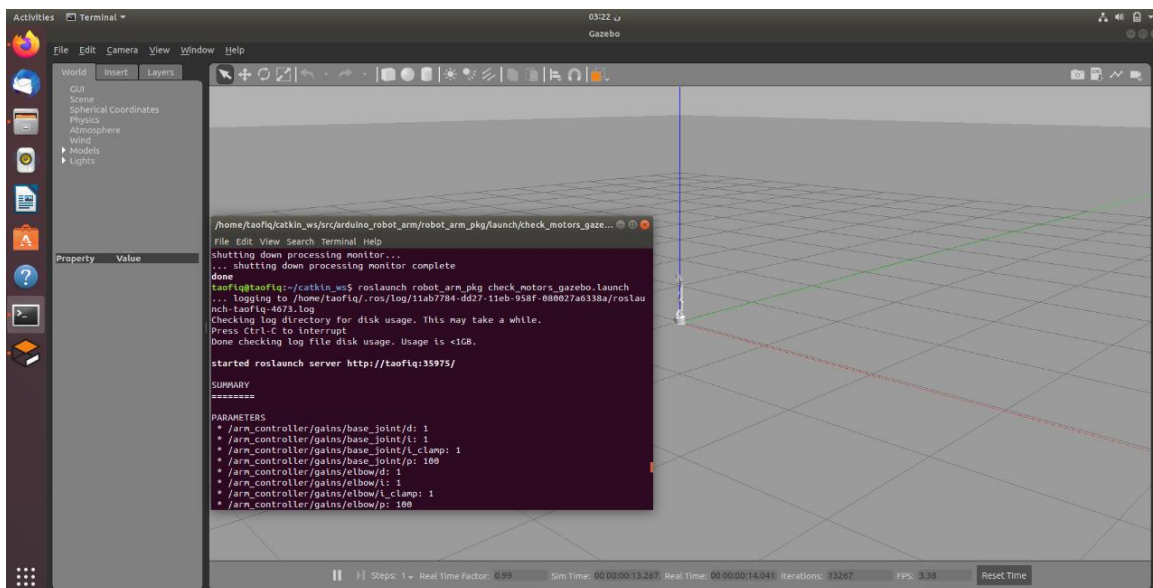
Third:

lunch the backage by using these codes:

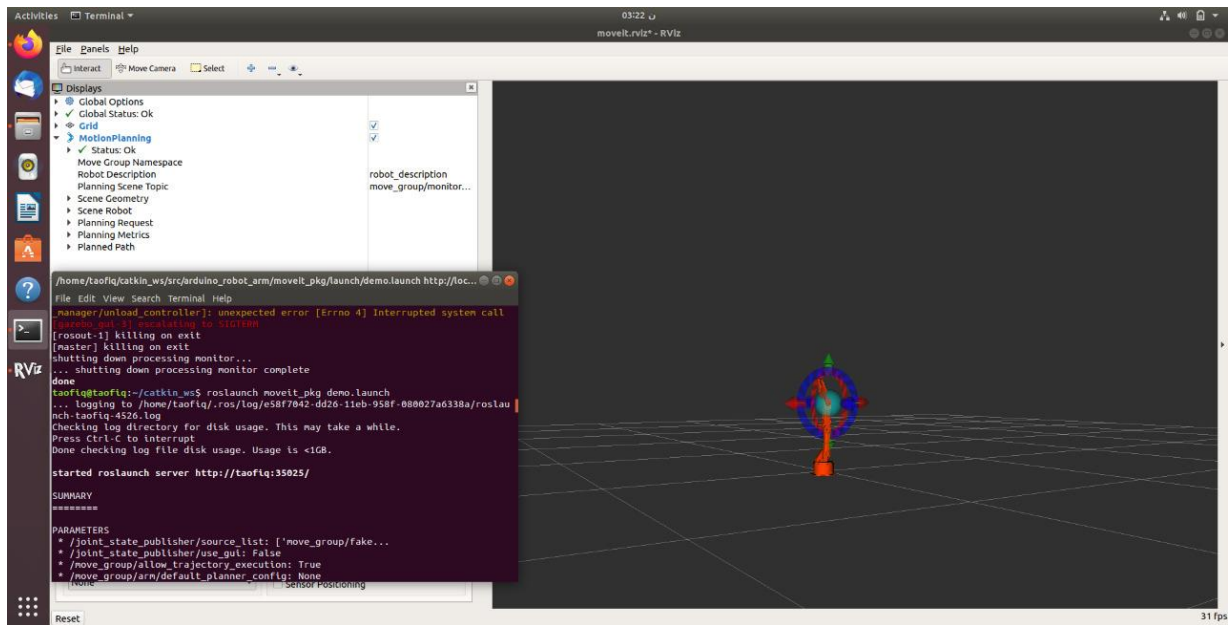
\$ roslaunch robot_arm_pkg check_motors.launch



\$ roslaunch robot_arm_pkg check_motors_gazebo.launch



\$ roslaunch moveit_pkg demo.launch



Moving the Robot

Part 1

First Install ROS 1 on Remote PC and used this codes

```
$ sudo apt update  
$ sudo apt upgrade  
$ wget https://raw.githubusercontent.com/ROBOTIS-GIT/robotis_tools/master/install_ros_melodic.sh  
$ chmod 755 ./install_ros_melodic.sh  
$ bash ./install_ros_melodic.sh
```

Second I Install Dependent ROS 1 Packages by this code

```
$ sudo apt-get install ros-melodic-joy ros-melodic-teleop-twist-joy \  
ros-melodic-teleop-twist-keyboard ros-melodic-laser-proc \  
ros-melodic-rgbd-launch ros-melodic-depthimage-to-laserscan \  
ros-melodic-rosserial-arduino ros-melodic-rosserial-python \  
ros-melodic-rosserial-server ros-melodic-rosserial-client \  
ros-melodic-rosserial-msgs ros-melodic-amcl ros-melodic-map-server \  
ros-melodic-move-base ros-melodic-urdf ros-melodic-xacro \  
ros-melodic-compressed-image-transport ros-melodic-rqt* \  
ros-melodic-gmapping ros-melodic-navigation ros-melodic-interactive-markers
```

Third they Install TurtleBot3 Packages by this codes

```
$ sudo apt-get install ros-melodic-dynamixel-sdk  
$ sudo apt-get install ros-melodic-turtlebot3-msgs  
$ sudo apt-get install ros-melodic-turtlebot3
```

Part 2

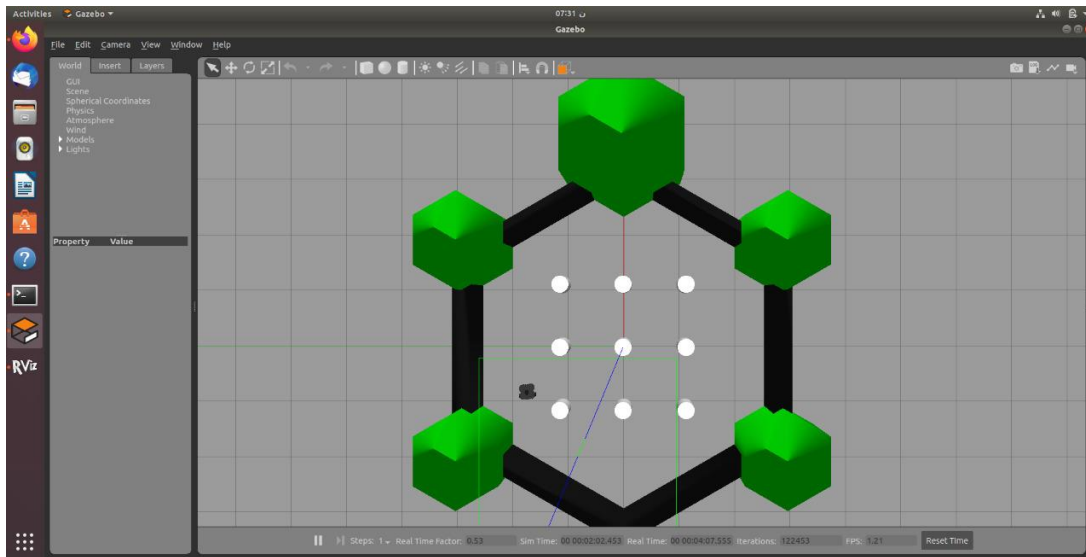
First ,Install Simulation Package by this codes

```
$ cd ~/catkin_ws/src/  
$ git clone -b melodic-devel https://github.com/ROBOTIS-GIT/turtlebot3\_simulations.git  
$ cd ~/catkin_ws && catkin_make\
```

Second , Launch Simulation World (TurtleBot3 World)

```
$ export TURTLEBOT3_MODEL=burger  
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

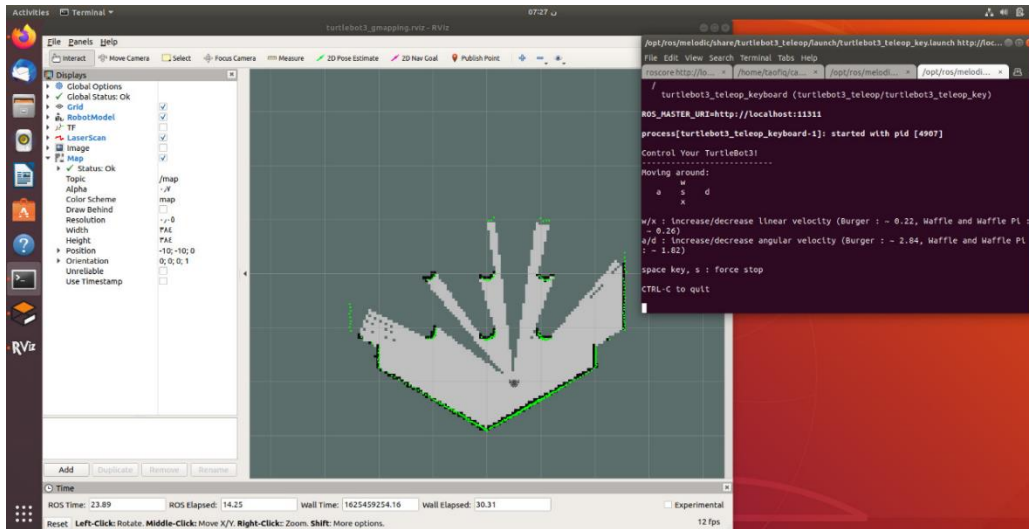
Third ,Run SLAM Node by codes



```
$ export TURTLEBOT3_MODEL=burger  
$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
```

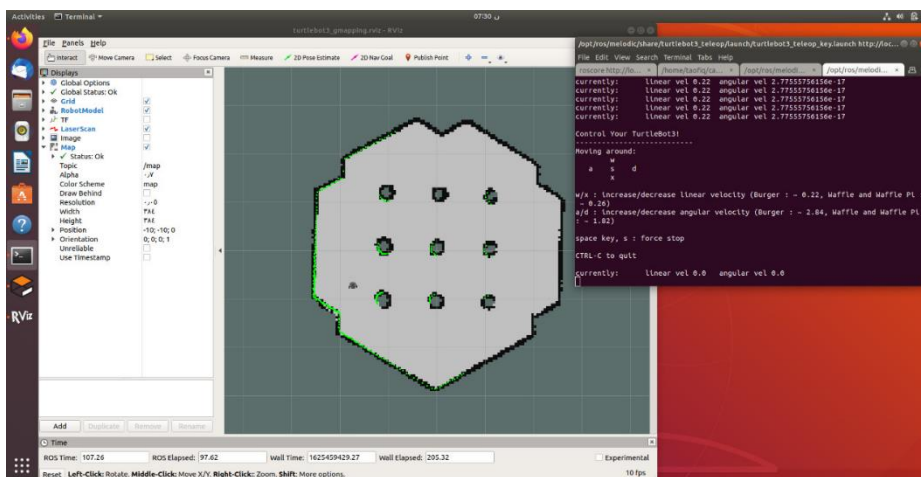
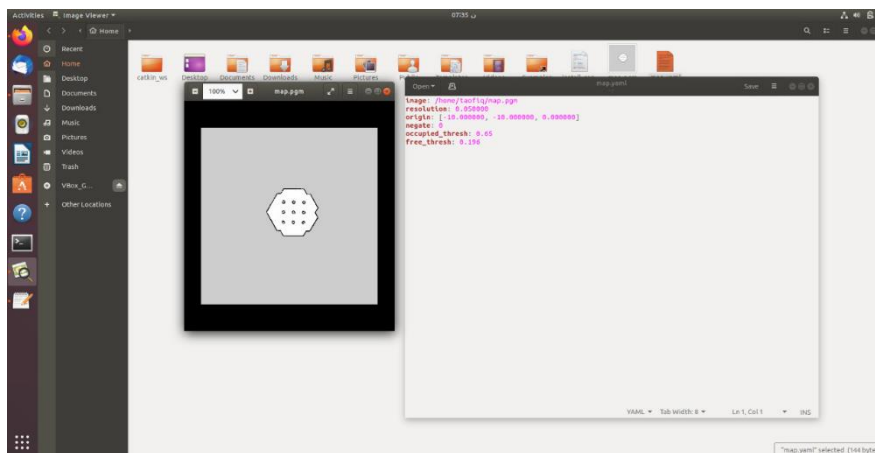
Fourth ,Run Teleoperation Node


```
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```



Finally, Save the Map by this code and get this results

```
$ rosrn map_server map_saver -f ~/map
```



Electronic

We will control the four NEMA 17 stepper motors using four DRV8825 stepper drivers, or also we could use the A4988 stepper drivers. For powering the steppers and the whole robot we will use 12V power supply, and in this case, we will use a 3S Li-Po battery which provides around 12V. We also included a simple voltage divider which will be used for monitoring the battery voltage and an LED connection for indicating when the battery voltage will drop below 11V. We also included a dedicated 5V voltage regulator which can provide around 3A of current. controlling servo motors for robot arm project

this is the electronic part of the robot arm project using 5 servo motors to control the joints of the robot arm using potentiometer in 90 degrees range. See the circuit connections of the servo motors with potentiometer in this file "circuit Robot Arm Servo Motors with potentiometer.png "

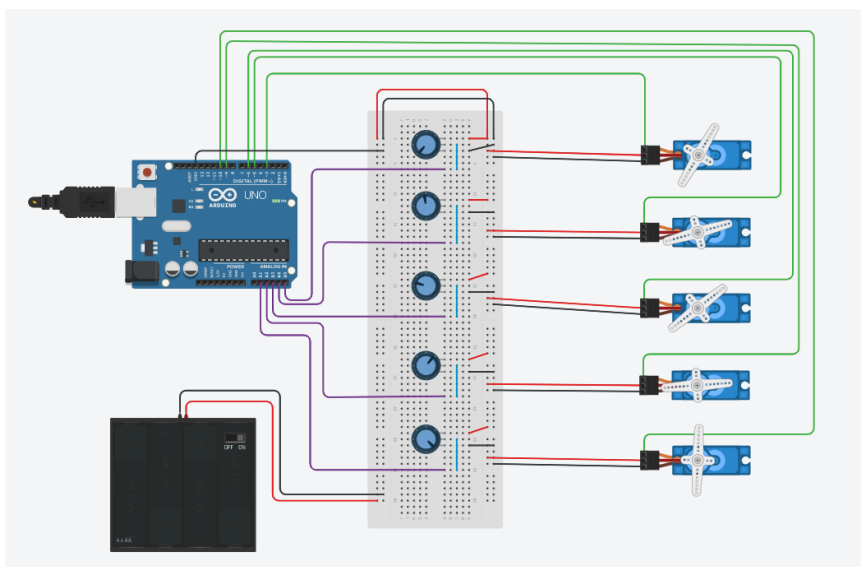
in the potentiometer circuit:

1- I connect all the servo motors to the power +battery and -battery and the GND of Arduino to -battery, and connect the signal wires of the servo to the Arduino PWM pins (3,5,6,9,10,11) .

2- connecting the power +battery and -battery to the terminal 1 and 2 of the Potentiometer, then connect the wiper of the potentiometer in analog pins (5,4,3,2,1).

in the code page you will find a description for the steps.

after connecting the circuit and run the code you can control the servo with the potentiometer from 90-degree angle to 0-degree angle as in the next picture:



the auto sweep 90-degree circuit its easier, see the connection in the file "circuit 90-degree Robot Arm Servo Motors.png"

1- They connect all the servo motors to the power +battery and -battery and the GND of Arduino to -battery, and connect the signal wires of the servo to the Arduino PWM pins (3,5,6,9 ,10,11)

```
63 lines (41 sloc) | 1.47 KB

1
2
3  #include <Servo.h> //include servo library
4
5  Servo servo1; // create servo object to control a servo
6  Servo servo2;
7  Servo servo3;
8  Servo servo4;
9  Servo servo5;
10
11  int ptpin1 = 5; // analog pin used to connect the potentiometer
12  int val1; // variable to read the value from the analog pin
13
14  int ptpin2 = 4;
15  int val2;
16
17  int ptpin3 = 3;
18  int val3;
19
20  int ptpin4 = 2;
21  int val4;
22
23  int ptpin5 = 1;
24  int val5;
25
26
27  void setup() {
28
29      servo1.attach(3); // attaches the servo on pin with a pwm
30
31      servo2.attach(5);
32
33      servo3.attach(6);
34
35      servo4.attach(9);
36
37      servo5.attach(10);
38
39  }
```

```
41 void loop() {
42   val1 = analogRead(ptpin1);           // reads the value of the potentiometer (value between 0 and 1023)
43   val1 = map(val1, 0, 1023, 90, 0);     // scale it to use it with the servo (angle value at 90 and 180 becuse we want its movment in 90 degrees range, so any 90 degrees range
44   servo1.write(val1);                  // sets the servo position according to the scaled value
45
46   val2 = analogRead(ptpin2);
47   val2 = map(val2, 0, 1023, 90, 0);
48   servo2.write(val2);
49
50   val3 = analogRead(ptpin3);
51   val3 = map(val3, 0, 1023, 90, 0);
52   servo3.write(val3);
53
54   val4 = analogRead(ptpin4);
55   val4 = map(val4, 0, 1023, 90, 0);
56   servo4.write(val4);
57
58   val5 = analogRead(ptpin5);
59   val5 = map(val5, 0, 1023, 90, 0);
60   servo5.write(val5);
61   delay(10);                          // waits for the servo to get there
62
63 }
```

2- Test

After finishing our fighting robot, we will do some tests (Functional & Non-functional) practically to enhance, develop and produce our project with full quality and accuracy.

3- Tolerance

In the testing process we will discover and predict the failures in each part starting from Internet of Thing (IOT) until electronic parts.

4- User Manual

Overview

This manual describes the right use of fighting robot.

Who should read this manual?

This manual is intended for fighting robot users, proposal engineers, mechanical designers, offline programmers, robot technicians, service technicians, PLC programmers, Robot programmers, and Robot System integrators.

Prerequisites

Windows Operating System Requirements

- Windows 7 or later. Including both 32-bit and 64-bit versions
- Microsoft .Net Framework 4.5

Mac Operating System Requirements

- Mac OS X 10.8 or later

Ubuntu16.04 LTS

- Mac OS X 10.8 or later

The Robot Operation System (ROS)

Safety Guideline

Please read the following information: failure to comply with provided information may lead to voiding the warranty.

This document covers safety, proper handling, and regulatory information for use of your Robotic Arm.

General Precautions

Caution: To avoid injury, damage to the robot or equipment, please follow the provided guidelines.

- Please read through the directions before starting.

-
- The robot is not intended for use by children under 15 years old, or any person with reduced physical, sensory or mental capabilities, or lack of experience and knowledge, unless:
 - Supervised by a person responsible for the child's safety and who has read and understood these instructions.
 - Keep the robot away from children under 3 years old at all times to prevent injury or damage.
 - Keep components or small parts away from children.
 - Keep away from pets and animals of any kind, animals may behave erratically in the presence of the robot.
 - First time users should take extra care when handling the robot to minimize injury or damage.
 - If the robot is operating abnormally, there is an unusual sound, smell or smoke is detected:
 - Turn the robot OFF immediately.
 - Unplug the robot.
 - Ensure the robot does not tip over or fall down.
 - Remove the battery (remove 1 screw on the back of the battery casing).
 - To prevent the spread of fire, keep candles or other open flames away from the robot at all times.
 - At all times, keep in mind safety first to prevent injury to individuals using or around the robot.
 - Always follow installation and service instructions closely. Keep manuals for future reference.
 - Review and follow all safety information provided throughout this manual. This guide does not cover all possible safety issues or conditions. Always use common sense and good judgment.
 - Warning: Conversion or modifications to this product not expressly approved by the party responsible for compliance could void the user's authority to operate the product.
 - The battery will become hot a little during charging, which is normal phenomenon. • Please take care of this unit and its accessories, keep them clean. Please do not let this unit or accessories exposed to fire/burning cigarette, etc... Try to keep the robot and its accessories dry; please do not let this unit exposed to water or moisture.
 - Please do not break, throw or trample the robot.
 - Avoid installation in extremely hot, rainy or water splashing, or being placed in high temperature or moist environment.
 - Please use the accessories we match for this robot.
 - Never disassemble or modify the smart servo in any way, otherwise, warranty of the product will be lost. For non-human faults or breakdown, please contact authorized distributors.
 - Please unplug the charging cord from the charger after charging completed to avoid over-charging.
 - Keep robot away from face and body when moving.
 - Do not use any tools other than those provided in the kit.

Battery Safety Warning:

DO NOT throw the battery in fire.

DO NOT short circuit the contacts.

DO NOT disassemble the battery.

DO NOT throw the battery in municipal waste.

The symbol of the crossed out wheeled bin indicates that the battery should not be placed in municipal waste.

Handling & Personal Safety

Buttons

- Before using, take a moment to locate the Power On/Off and directional buttons and familiarize yourself with the functions.

Handling

- Handle the robot with care at all times.
- Pick up the robot by the bar in the event of the following situations:
 - If the robot comes close to danger, exposure to water, or falling off a surface.
 - Before the robot knocks over objects.
 - If the robot's operation seems erratic.
- Procedure for handling in a potentially hazardous event:
 1. Pick up the robot by bar even if it is still moving.
 2. Press the Power On/Off button to turn off.
 3. . Put the robot in the starting default posture.

Refer to the User Guide for powering on instructions.

- In case of a power emergency, do not touch the robot.



CAUTION:

Pinching Hazard

- Avoid carrying and touching the robot when it is moving, walking or getting up to avoid getting pinched.
- Keep hand and fingers out of the joint areas to avoid getting caught in between.
- To avoid injury, do not place your hands in any joint to prevent damage or personal injury.

General

IMPORTANT:

- The robot requires a clean work space to move around and perform activities.
- Do not insert any foreign objects into any of the component or internal cavities.
- Generally, do not allow or cause the robot to fall down.
- Do not exert strong force against the joints or actuators (smart servos).
- Do not grab and pull by the head or arms/legs to prevent exerting force on the joints.

Working Area

- Start from the default posture before turning on the robot.
- When turned off, it is best to keep the robot in the default posture without the cable plugged unless charging.
- The working surface must be dry and level; thick carpets or rugs are not recommended for operational stability.
- The robot is designed for indoor use only.
- Keep the robot away from radiators, heat sources and direct sunlight.
- Do not leave the robot unattended on the floor.

5- WARRANTY & LIABILITIES

Your Fighting Robot is fully warranted against defective parts or assembly for 90 days after it is shipped to you from the factory. Accessories are warranted for 90 days. This warranty does not include damage from abuse or inappropriate operation.

The following are specifically not covered by warranty: (1) failure due to abuse and neglect and/or improper operating environment (including, but not limited to, improper power supply, temperature, humidity, and environmental conditions); (2) items such as batteries, software.