Design and Construction of Desinfiction Robot for Hospitals

Ahmed Mohmed El khatib, Ahmed Mahmoud Mostafa,
Ahmed Mansour Attia El Metwally, Ahmed Yossry Hassan Refaey
And Mohamed Ashraf Said Mostafa
ameco1998@gmail.com,
Supervisor: Prof.. Mohamed Ibrahim
Faculty of Engineering, Ain Shams University, Department of Mechatronics, Egypt,
eng.asu.edu.eg

Abstract- The global COVID-19 pandemic due to the novel coronavirus SARS-CoV-2 has challenged the availability of traditional surface disinfectants. It has also stimulated the production of ultraviolet-disinfection robots by companies and institutions. These robots are increasingly advocated as a simple solution for the immediate disinfection of rooms and spaces of all surfaces in one process and as such they seem attractive to hospital management, The work presented in this paper, demonstrates the construction of a two-wheel differential drive robot that works The robot is driven by two DC motors and equipped by an IMU sensor to sense the rotation angle of the robot, it also contains a pump, tank and spray system as well as UV lamp connected to an inverter from the battery. It also has a PIR system that covers 360 degrees around the robot to detect humans for safety as UV doesn't work in human presence. A PID-controller has been designed with the function to stabilize the two-wheeled robot. To control its motion. The value of the PID-controller gains has been tuned through several experiment. It has a high level controller consisting of Raspberry Pi set up with Ros and low level controller consisting of 2 Arduino Megs. The constructed two-wheel deferential drive robot prototype is capable of mapping the hospital or location and autonomies motion to destination and disinfection and sterilization of surfaces at location.

Keywords-- Robotics, Medical Robots, ROS Operating system,Two-Wheeled Robot. Deferential Drive Robot.

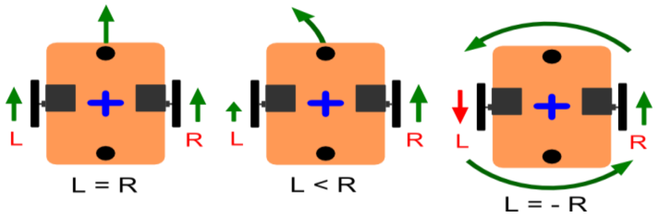
Nomenclature:

IMU *……… Inertial Measurement Unit.*
DOF *............ Degree of Freedom.*
ROS *………… Robot Operating System.*

I. INTRODUCTION

The global COVID-19 pandemic due to the novel coronavirus SARS-CoV-2 has challenged the availability of traditional surface disinfectants. It has also stimulated the production of ultraviolet-disinfection robots by companies and institutions. These robots are increasingly advocated as a simple solution for the immediate disinfection of rooms and spaces of all surfaces in one process and as such they seem attractive to hospital management, also because of automation and apparent cost savings by reducing cleaning staff. Yet, there true potential in the hospital setting needs to be carefully evaluated. Presently, disinfection robots do not replace routine (manual) cleaning but may complement it. The medical Robot we designed is a two-wheeled differential drive robot that has an Arduino mega controlling the two motors through a HC33886 Dual DC Driver and feedback the velocity of each motor from rotary encoders connected on the shaft. The IMU calculates the angle by which the robot takes during motion for more sensitivity and to not be affected by slipping. There is a pump connected to the Arduino through a relay to control the spray system connected to the pump. PIR sensors are arranged to cover 360 degrees around the robot to detect humans to stop the spray and more importantly the UV as it can be harmful to humans. The Raspberry Pi is the high-level control of the system that controls the Arduinos connected to which act as low-level control to control the robot's different functions. The raspberry pi operates with ROS which is Robotics Operating System developed for an easier way to code and operate the robot and has a lot of open-source codes the have codes ready that we can use by adding the parameters of our project and modifying the code to suite our use. The robot uses deferential drive which is added by PID controller as in real life applying the same voltage to each motor doesn't mean that it will rotate at the same rate as each other as each motor is made of components that cause error. We also added a PI/D parameter modifier in the Ros code that modifies the parameter automatically to ensure accuracy.

We used deferential drive as an easy way but depends highly on PID controller parameter to produce same rotation on each motor when given power and to be able to control the rotation speed an each motor with equal ratios in order to move in different directions.



II. System Layout

Fig. 1 The Developed Two-Wheel Robot Layout.

The system layout (Fig.1) consists of a holder that is made of two shelves made of wood and connected together by metallic rods to have sufficient rigidity. The two motors are fixed with a steel l-shaped part. There ia a box on the metallic roof containing the buttons for system different functions, emergency stop and to LCDs for battery level and tank level. The system modules are installed on the shelves as follows:

a. The bottom shelf holds the Li-ion batteries and pump.

b. The middle shelf holds the Arduino controller + Raspberry Pi + IMU Sensor.

c. The tank pass through the two shelves.

The bottom of the holder is bolted to the two DC motors, and the wheels (tires) are connected to the two motor shafts. The motors & wheels will be denoted hereafter as the wheel assembly.

Fig. 2 Circuit Components

The system is typically comprising the following modules:

- An The **Arduino Mega 2560** is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards.

- **Raspberry Pi 4 Model B** was released in June 2019 with a 1.5 GHz 64-bit quad core ARM Cortex-A72 processor, on-board 802.11ac Wi-Fi, Bluetooth 5, full gigabit Ethernet (throughput not limited), two USB 2.0 ports, two USB 3.0 ports, and dual-monitor support via a pair of micro HDMI (HDMI Type D) ports for up to 4K resolution. The Pi 4 is also powered via a USB-C port, enabling additional power to be provided to downstream peripherals, when used with an appropriate

PSU. The initial Raspberry Pi 4 board has a design flaw where third-party e-marked USB cables, such as those used on Apple MacBooks, incorrectly identify it and refuse to provide power. Tom's Hardware tested 14 different cables and found that 11 of them turned on and powered the Pi without issue. The design flaw was fixed in revision 1.2 of the board, released in late 2019.

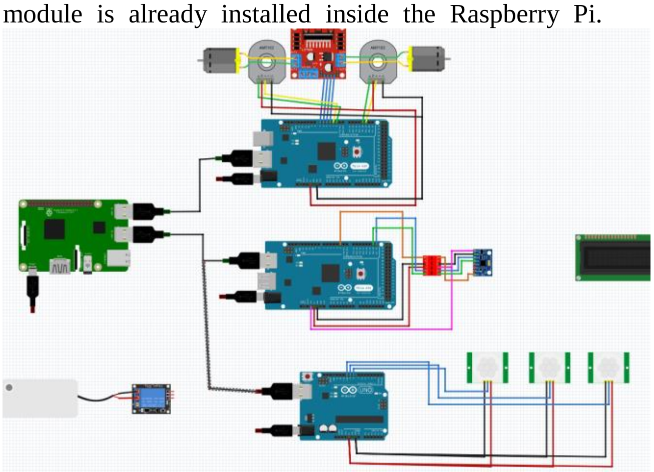
- The **MPU6050** navigation sensor that has 3-axis accelerometers and 3-axis gyros together with an on-board Digital Motion Processor (DMP).
- The accelerometers measure acceleration along the three axes and the gyros measuring angular rate (rotational velocity) around the three axes.
- The angle of rotation of the robot is measured via the gyro readings along the Z-axe.

- A standard **USB** serial communication data bus has been adopted to manage the data transfer between the **Raspberry Pi** and the **Arduino** controller.

- Two geared DC-motor (**SG-7755125000-40k Geared Motor**), each one is directly connected to a single wheel.

- Dual motor driver module (**HC33886 Dual DC Driver**), which is a dual H-Bridge motor driver that controls the speed and direction of the two DC motors at the same time. The Controller determines the appropriate value of motor input volt and passes it to the motor driver in PWM format. The module can drive DC motors that have voltages range between 5 and 30V, with a maximum current up to 5A.

- The robot is controlled using a laptop, and the control commands are transferred to the microcontroller via the **Wi-Fi connection** that establishes the required long-range wireless data communication between the High-level controller and laptop or workstation. The **Wi-Fi** module is already installed inside the Raspberry Pi.



III. System Modeling

- **Mechanical Parameters Identification:**

The system mechanical components have been modeled using **SOLIDWORKS** to estimate the value of mass, location of the center of mass, and the value of the second moments of inertia about the three axes.

Fig. 3 depicts the CAD model of the system on the **SOLIDWORKS**.

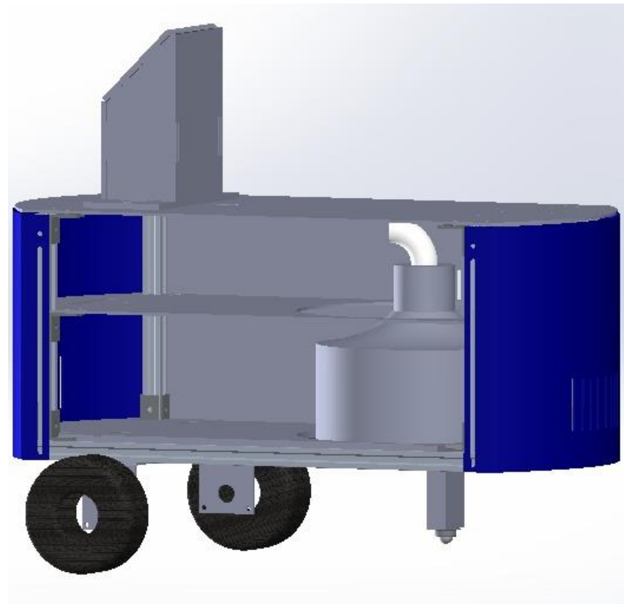
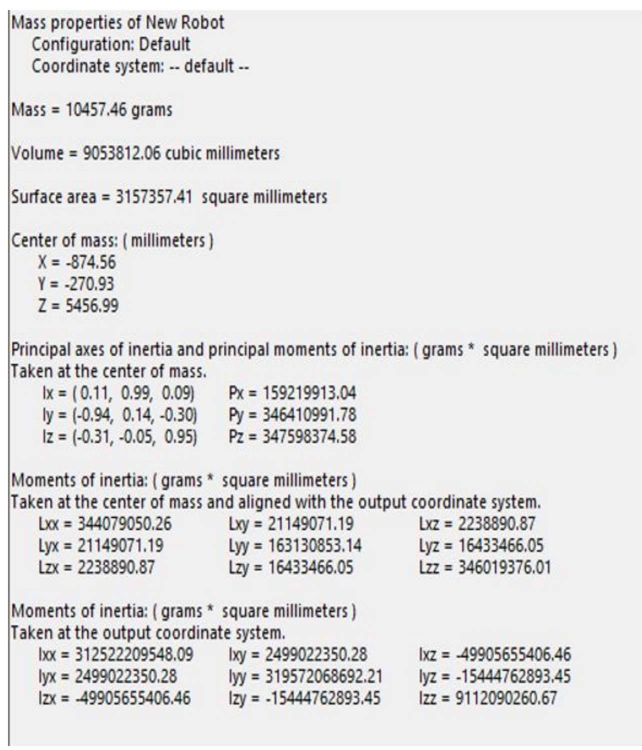


Fig. 3 The CAD model



Mass properties of New Robot
Configuration: Default
Coordinate system: -- default --

Mass = 10457.46 grams

Volume = 9053812.06 cubic millimeters

Surface area = 3157357.41 square millimeters

Center of mass: (millimeters)
X = -874.56
Y = -270.93
Z = 5456.99

Principal axes of inertia and principal moments of inertia: (grams * square millimeters)
Taken at the center of mass.
Ix = (0.11, 0.99, 0.09) Px = 159219913.04
Iy = (-0.94, 0.14, -0.30) Py = 346410991.78
Iz = (-0.31, -0.05, 0.95) Pz = 347598374.58

Moments of inertia: (grams * square millimeters)
Taken at the center of mass and aligned with the output coordinate system.
Lxx = 344079050.26 Lxy = 21149071.19 Lxz = 2238890.87
Lyx = 21149071.19 Lyy = 163130853.14 Lyz = 16433466.05
Lzx = 2238890.87 Lzy = 16433466.05 Lzz = 346019376.01

Moments of inertia: (grams * square millimeters)
Taken at the output coordinate system.
Ixx = 312522209548.09 Ixy = 2499022350.28 Ixz = -49905655406.46
Iyx = 2499022350.28 Iyy = 319572068692.21 Iyz = -15444762893.45
Izx = -49905655406.46 Izy = -15444762893.45 Izz = 9112090260.67

Fig. 4 depicts Mass Properties.

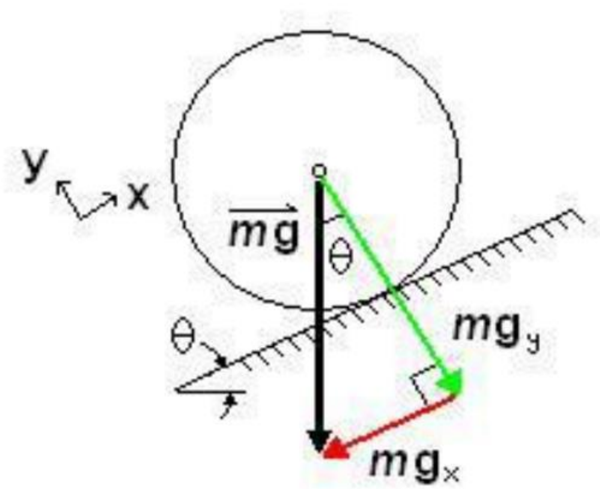
- **Calculations:**

To calculate the required torque, power, current and battery pack required by a wheeled mobile robot, there are several principles that must be understood: concept of vectors; 2D Force balance; Power; Current and Voltage.

In order to roll on a horizontal surface, a wheeled robot's motors must produce enough torque to overcome any imperfections in the surface or wheels, as well as friction in the motor itself. Therefore theoretically, a robot (small or large) does not require much torque to move purely horizontally. Obviously, there will be more friction and resistance in a large robot than in a small robot, though it is still exponentially less than when a robot encounters an incline.

In order for a robot to roll up an incline at a constant velocity (no acceleration or deceleration) it must produce enough torque to "counteract" the effect of gravity, which would otherwise cause it to roll down the incline. On an inclined surface (at an angle theta) however, only one component of its weight (mg_x parallel to the surface) causes the robot to move downwards. The other component, mg_y is balanced by the normal force the surface exerts on the wheels.*:*

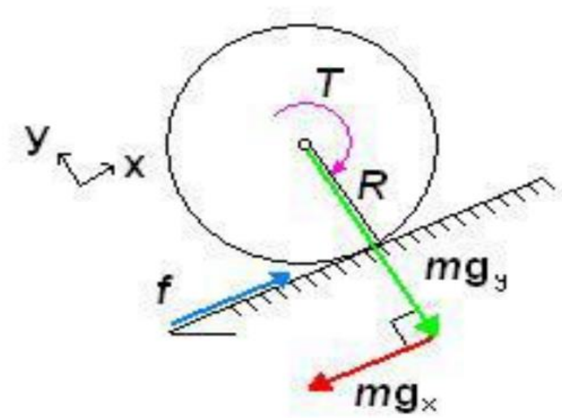
$$mg_x = \boldsymbol{mg} * \boldsymbol{sin}(\boldsymbol{\theta})$$
$$mg_y = \boldsymbol{mg} * \boldsymbol{cos}(\boldsymbol{\theta})$$



In order for the robot not to slide down the incline, there must be friction between the wheel and the surface. The motor in a heavy truck may be able to produce 250 horsepower and significant torque, but we have all seen (in person or in video) large trucks simply spinning their wheels as they fall backwards on an icy street. It is friction (f) that "produces" the torque.

The torque (T) required is:

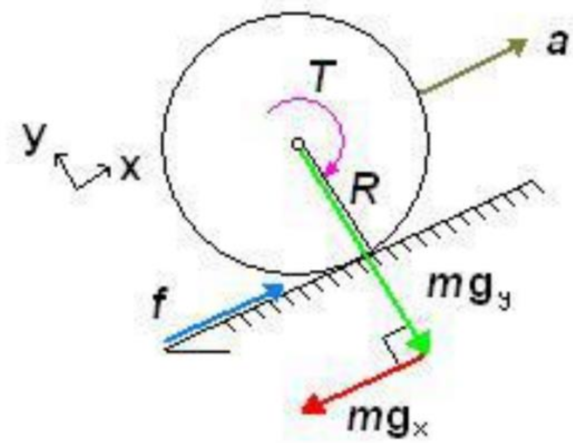
$$\boldsymbol{T} = \boldsymbol{f} * \boldsymbol{R}$$



To select the proper motor, we must consider the "worst case scenario", where the robot is not only on an incline, but accelerating up it.

Note now that all forces (F) are along the x and y axes. We balance the forces in the x- direction:

$$\sum \boldsymbol{F_x} = \boldsymbol{M} * \boldsymbol{a} = \boldsymbol{f} - \boldsymbol{m} * \boldsymbol{g_x}$$



Inserting the equation for torque above, and the equation for mg_x, we obtain:

$$\boldsymbol{M} * \boldsymbol{a} = \frac{\boldsymbol{T}}{\boldsymbol{R}} - \boldsymbol{m} * \boldsymbol{g} * \boldsymbol{sin}(\boldsymbol{\theta})$$

Rearrange the equation to isolate T:

$$\boldsymbol{T} = \boldsymbol{R} * \boldsymbol{M} * (\boldsymbol{a} + \boldsymbol{gsin}(\boldsymbol{\theta}))$$

This torque value represents the total torque required to accelerate the robot up an incline. However, this value must be divided by the total number (N) of drive wheels to obtain the torque needed for each drive motor. Note that we do not consider the total number of passive wheels as they have no effect on the torque required to move the object aside from adding weight.

$$T = \frac{\boldsymbol{R} * \boldsymbol{M} * (\boldsymbol{a} + \boldsymbol{gsin}(\boldsymbol{\theta}))}{N}$$

The final point to consider is the efficiency (e) in the motor, gearing and wheel (slip).

$$T = \left(\frac{100}{e}\right) * \frac{\boldsymbol{R} * \boldsymbol{M} * (\boldsymbol{a} + \boldsymbol{gsin}(\boldsymbol{\theta}))}{N}$$

This increases the torque required and compensates for inefficiencies. Total power (P) per motor can be calculated using the following relation:

$$P = T * \omega$$

T is known from above and the angular velocity (w) is specified by the builder. It is best to select the maximum angular velocity to be able to find the corresponding maximum power. Knowing the maximum power and the supply voltage (V) which the builder chooses, we can find an idea of the maximum current (I) requirements:

$$P = I * V$$

The two equations above are used to produce the following relation:

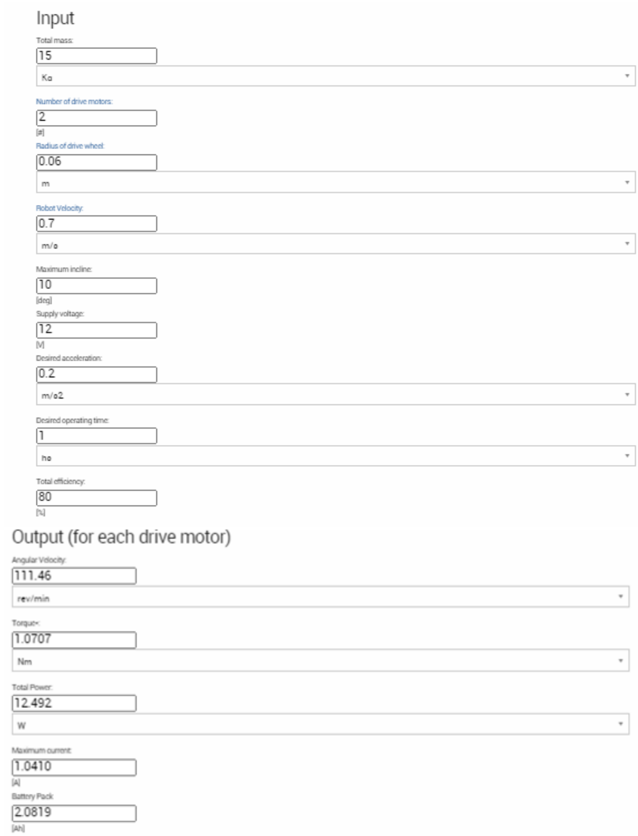
$$I = \frac{T * \omega}{V}$$

Finally, the capacity (c) of battery pack required can be estimated using the equation:

$$C = I * t$$

You may wonder why such a large value is needed. This is because when choosing a battery pack, the rated amp hours are not an accurate indicate of the maximum current the pack can produce for extended periods of time. Also, the total charge is rarely retained over time.

Using the calculations above we made our own motor sizing tool; our motor sizing tool was build using C#. The Drive Motor Sizing Tool is intended to give an idea of the type of drive motor required for your specific robot by taking known values and calculating values required when searching for a motor. DC motors are generally used for continuous rotation drive systems, though can be used for partial (angle to angle)
rotation as well. They come in an almost infinite variety of speeds and torques to suite any need. Without a gear down, DC motors turn very fast (thousands of revolutions per minute (rpm)) but have little torque. To get feedback of the angle or the speed of the motor, consider a motor with an encoder option.



Using the gained outputs from the motor sizing calculations, we can pick up the motors knowing these specs.:

- o Torque: 1.86 Nm
- o RPM: 125 rpm
- o Voltage: 12 V
- o Max Current: 4 A

- **Differential Drive Kinematics**

Many mobile robots use a drive mechanism known as differential drive. It consists of 2 drive wheels mounted on a common axis, and each wheel can independently being driven either forward or backward. While we can vary the velocity of each wheel, for the robot to perform rolling motion, the robot must rotate about a point that lies along their common left and right wheel axis. The point that the robot rotates about is known as the ICC.

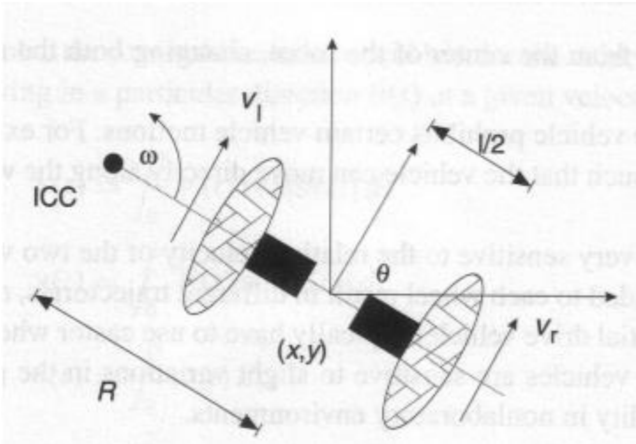


Figure 4 Differential Drive kinematics (from Dudek and Jenkin, Computational Principles of Mobile

By varying the velocities of the two wheels, we can vary the trajectories that the robot takes. Because the rate of rotation ω about the ICC must be the same for both wheels, we can write the following equations:

$$\omega\,(R + l/2) = V_r$$
$$\omega\,(R - l/2) = V_l$$

where l is the distance between the centers of the two wheels, V_r , V_l are the right and left wheel velocities along the ground, and R is the signed distance from the ICC to the midpoint between the wheels. At any instance in time, we can solve for R and ω:

$$R = \frac{1}{2} * \frac{V_r + V_l}{V_r - V_l}; \ \omega = \frac{V_r - V_l}{l}$$

There are three interesting cases with these kinds of drives.

1. If $V_l = V_r$, then we have forward linear motion in a straight line. R becomes infinite, and there is effectively no rotation - ω is zero.

2. If $V_l = -V_r$, then R = 0, and we have rotation about the midpoint of the wheel axis - we rotate in place.

3. If $V_l = 0$, then we have rotation about the left wheel. In this case R = $\frac{1}{2}$. Same is true if $V_r = 0$.

Note that a differential drive robot cannot move in the direction along the axis - this is a singularity. Differential drive vehicles are very sensitive to slight changes in velocity in each of the wheels. Small errors in the relative velocities between the wheels can affect the robot trajectory. They are also very sensitive to small variations in the ground plane and may need extra wheels (castor wheels) for support.

- **Forward kinematics**

Assume the robot is at some position (x, y), headed in a direction making an angle θ with the X axis. We assume the robot is centered at a point midway along the wheel axle. By manipulating the control parameters V_l , V_r, we can get the robot to move to different positions and orientations. (Note: V_l , V_r) are wheel velocities along the ground). Knowing velocities V_l , V_r and using equation 3, we can find the ICC location:

$$ICC = [x - R\sin(\theta),\, y + R\cos(\theta)]$$

and at time t + δt the robot's pose will be:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{bmatrix}$$

This equation simply describes the motion of a robot rotating a distance R about its ICC with an angular velocity of ω.

IV. System Engineering

PID-Controller gains have been tuned experimentally, and finally the values of the gains have been added to the Arduino code of the motor as well as adding a function in the ROS coding as to tune the parameter automatically during motion in order to decrease error..

The wiring of the motor arduino is demonstrated in Fig.6.

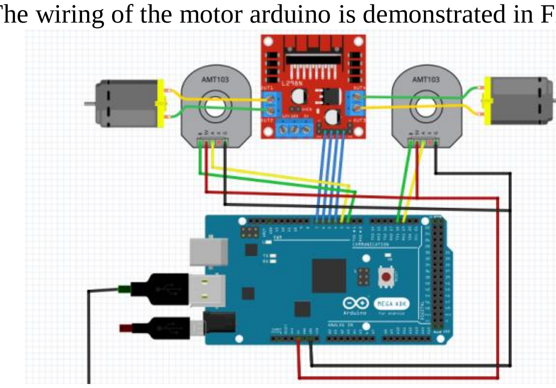


Fig. 6 Motor Connection Diagram

- The wiring of the IMU module is depicted in Fig.7.

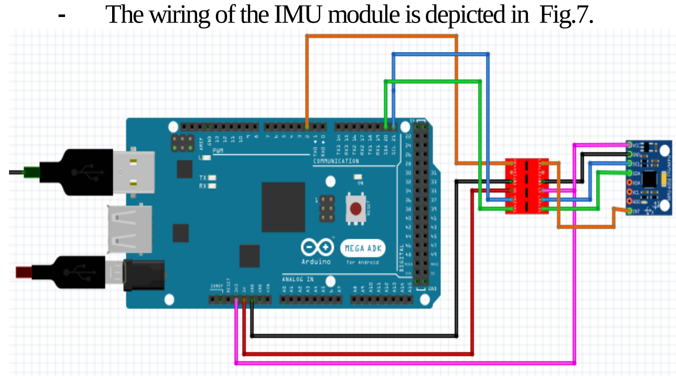


Fig. 7 IMU Connection Diagram

V. SYSTEM MODES OF OPERATION

- The system has three modes of operation:

(1) **Mapping mode**, for which the robot moves with the help of a controller and a person in the place to be mapped as will activating this mode.

(2) **Spray mode**, for which the robot moves autonomously to the target location while it sprays the surface disinfection as long as the PIR sensors don't read human presence

(3) **Spray and UV mode**, for which the robot moves autonomously to the target location and activates while it sprays the surface disinfection as long as the PIR sensors don't read human presence at target room.

(4) VI. CONCLUSIONS AND RECOMMENDATIONS

The current COVID-19 pandemic boosts innovation on many publics, societal and medical levels and disinfection practices are not an exception. Disinfection robots are a promising tool for surface decontamination in the hospital already today, but with even greater potential tomorrow. Further design adjustments of hospitals and devices are needed to overcome the issue of shadowing and free the movement of robots in the hospital environment. One-size does not fit all, and apart from communication between robot and the environment, more work must also be invested in defining efficient wavelength and exposure time to allow sufficient energy to be applied on each surface, as a function of the intended pathogen to be inactivated. Finally, a fit-for-purpose hospital environment would allow disinfection robots to function independently.

Presently, disinfection robots do not replace routine (manual) cleaning but may complement it. They might in the future provide validated, reproducible and documented disinfection processes. Further technical developments and clinical trials in a variety of hospitals are warranted to overcome the current limitations and to find ways to integrate this novel technology into the hospitals of to-day and the future.

However, there were different challenges in autonomous mobile robot systems which made it a bit difficult to apply results proposed in research to get reliable products that can be used in industry, gladly we managed to find our way through these challenges. The first challenge was cost, and gladly we managed to move beyond that with our cheap robot. The second challenge was the complexity in using ROS platform and reaching an output using it.

VII. REFERENCES

[1] Andrew k. Stimac, Standup & Stabilization of the Inverted Pendulum, BSc Thesis, Massachusetts Institute of Technology, USA, June 1999.

[2] F. Grasser, A. D'Arrigo, S. Colombi and A. C. Rufer, JOE: A mobile, inverted pendulum, IEEE Transactions on Industrial Electronics, 49(1), 107-113, 2002.

[3] Peter Miller, Building a Two Wheeled Balancing Robot, BSc Thesis, University of Southern Queensland, USA, October 2008.

[4] Bikash Kumar Bhawmick, "Design and Implementation of A Self-Balancing Robot", Research Gate On-line, University of Dhaka, December 2017.

[5] Baskaran Shanmugavel, "Development of Self Balancing Robot", Research Gate On-line, Bannari Amman Institute of Technology, July 2018.

[6] Fabian Kung, "A Tutorial on Modelling and Control of Two-Wheeled Self-Balancing Robot with Stepper Motor", Research Gate On-line, Multimedia University, July 2019.

[7] Manikandan S., "A new Design and Control of a Two-Wheel Self- Balancing Robot using the Arduino Microcontroller", Research Gate On-line, Ganesh College of Engineering and Technology, March 2018.

[8] H. A. Kadir, Modelling and control of a balancing robot using digital state space approach, Master Thesis, University Technology Malaysia, 2005.

[9] R. Chan, K. A. Stol and R. C. Halkyard, Review of modelling and control of two-wheeled robots, Annual Reviews in Control, 37, 89-103, 2013.