Advanced Autonomous Industrial Robot: Holonomic Navigation Drive System and Vision-Based Material Handling

Farhan Daemi[1], Padina Farrokhian[2]

[1,2] Imam Khomeini International University, Tehran, Iran
fdmxfarhan@yahoo.com
padina3386@gmail.com

Abstract. In this paper, we explore various methods of designing and implementing an industrial robot with omnidirectional navigation and vision-based object detection capabilities for automated material handling. The robot features a holonomic drive system, enabling it to move in any direction without changing its orientation, thus offering high maneuverability in crowded industrial environments. Our investigation includes experimenting with different vision systems that utilize combinations of cameras and image processing algorithms to detect and recognize objects in the robot's environment. Additionally, we test various configurations of the robot's arm for fully automated object handling without human intervention. The integration of navigation and object detection systems is assessed to determine the most effective approach for the robot to locate, retrieve, and transport objects to designated locations. We provide a comprehensive analysis of the hardware and software components and present experimental results comparing the performance of different designs. These results highlight the optimal methods for achieving accurate navigation, precise object detection, and efficient pick-and-place operations. Our findings demonstrate the potential for a fully automated industrial robot to enhance material handling efficiency, thereby reducing the need for human intervention in industrial settings.

Keywords: Industrial Robots, Omnidirectional Navigation, Robots Vision

1 Introduction

Nowadays the interaction of humans in industry is being replaced by robots. Robots will do humans jobs faster and in a better quality. In general robots can be divided into two categories, resident robots and moving robots. Resident robots are robots that do their tasks in a single position and don't need to move themselves like robotic arms. Moving robots can move into other positions in order to do their tasks, such as storekeeper robots or human companion robots. The number of moving axis will affect their performance and efficiency in some cases. Omnidirectional moving systems will help these robots to move in any direction in a 2-dimensional space. The main idea in this article is to build an industrial robot that can navigate in a workspace and detect objects

using a vision system on a table and pick one of the objects according to the command that is given by a human or another software system. Then move toward another section inside the workspace and place that object there with specific position and angle. To build this robot we need a mechanical design for the moving system and the robotic arm. Moving system includes motors and special wheels that allow the robot to move in any direction without rotating itself. To meet this need, we can use omnidirectional wheels or "Mecanum" [4] wheels. These wheels have some small rings around it placed in 45 degrees. For the robotic arm, to access any position around itself, it needs at least six degrees of freedom (DOF). This is because a six-DOF arm can move in three-dimensional space and reach any position within its workspace. Each DOF provides a specific type of motion to the robotic arm. For example, a revolute joint provides rotational motion, while a prismatic joint provides linear motion. A six-DOF robotic arm typically has six joints, each providing a different type of motion. These joints are usually arranged in a serial chain, with the first joint located at the base of the arm and subsequent joints positioned along the arm's length. It's worth noting that some robotic arms may have more than six DOF, which can provide additional flexibility and precision. However, six DOF is the minimum required for a robotic arm to access any position around itself.

2 Robot Description

2.1 Mechanical Design

The design of the robot is very important because the more accurate a robot is, the more efficient our controller would be. The first step is to design a movement system which allows the robot to move in any direction without limitation. This kind of movement is possible using "Mecanum" wheels which have 45-degree rings around the main wheel which can rotate freely. After choosing the type of movement system the second part of designing the robot is simulating the mechanics in a 3D designing software. In this case SolidWorks [5] software is used to design the main structure of the robot. So, the main platform of the robot is a plate of aluminum with a thickness of 2 millimeters which all other parts will be placed on it using screws. In the next step, motors and wheels are added to this bent aluminum sheet. Two thin aluminum profiles are attached to the bottom plate to increase the strength of the main platform. The design of the robot's main platform is shown in Figure 1.

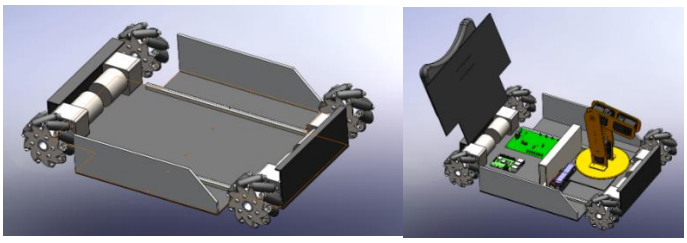

Fig. 1. Main *platform simulation*

2.2 Electrical Design

To control all the actuator motors and read sensors data it is needed to have an appropriate hardware on the robot. This hardware includes dc motor drivers, step motors drivers, feedback sensors, communication modules, voltage regulators. VNH5019 is a very common dc motor driver integrated circuit designed for driving high-current motors with precise control. It also incorporates several protection features such as over-temperature shutdown, under-voltage lockout, and over-current protection. Another common driver for step motors is TB6600 which is designed to control bipolar stepper motors. It provides a convenient and efficient solution for driving stepper motors in various applications such as CNC machines, 3D printers, and robotics. The TB6600 is often used as a drop-in replacement for older stepper motor drivers like the L298N or A4988. In order to make the robot movement and functionality more precise it is recommended to use controllers with closed loop feedback. two feedback sensors for movement system which are used in this robot is GY-25 [7] accelerometer [8] with Calman filter and medium resolution encoders for each DC motor. The GY-25 module, based on the MPU-6050 [9], extends its functionality by incorporating a digital compass (magnetometer) alongside the accelerometer and gyroscope. This combination of sensors enables the module to provide comprehensive motion sensing and orientation information. The module typically communicates with the microcontroller using I2C [10] (Inter-Integrated Circuit) protocol. The designed hardware will be connected to a computer through Wireless module which in this case ESP8266 [11] is used as Wi-Fi interface. The computer will do all the processing and calculation using Robot Operating System (ROS) [1] and outputs the velocity of each motor to the hardware system. The hardware controls all the elements to achieve the commands receiving from ROS. A camera is connected to the computer for vision-based object detection which is not discussed in this article. A 2-dimension laser scanner is also connected to the computer which help the robot to determine its position.

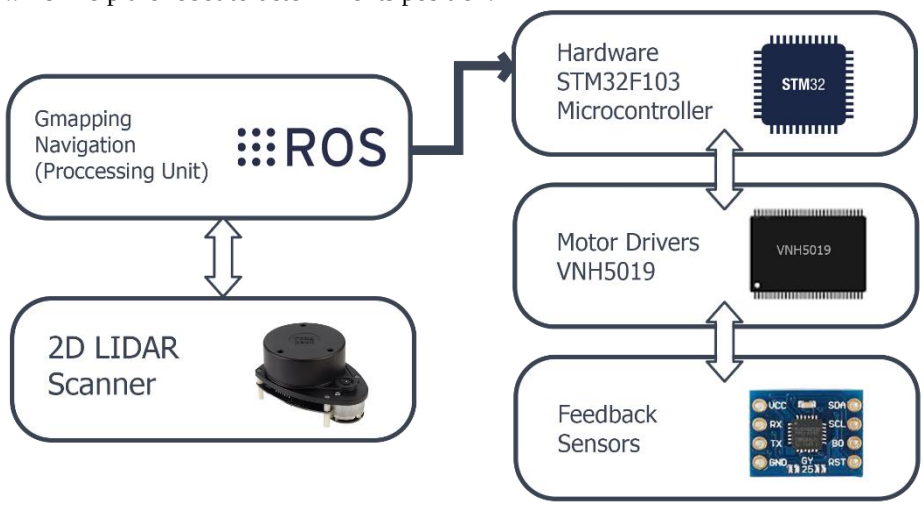

Fig. 2. Hardware Diagram

3 Software Design

3.1 Odometry

Odometry refers to the process of measuring the position, orientation, and velocity of a vehicle by analyzing its movement and the data collected from its sensors, such as wheel encoders or inertial measurement units (IMUs). By measuring the rotation of the wheels by rotary encoders or the accelerations and angular velocities sensed by the IMU, it is possible to estimate the displacement and angular changes over time. However, Factors like slippage, wheel skidding, sensor noise, and environmental conditions can affect the odometry estimates and lead to errors. To enhance the accuracy of odometry, combining this method with other techniques such as sensor fusion (combining data from multiple sensors), landmark navigation (using reference points or landmarks in the environment), or simultaneous localization and mapping (SLAM) algorithms can be employed. Functionalities …

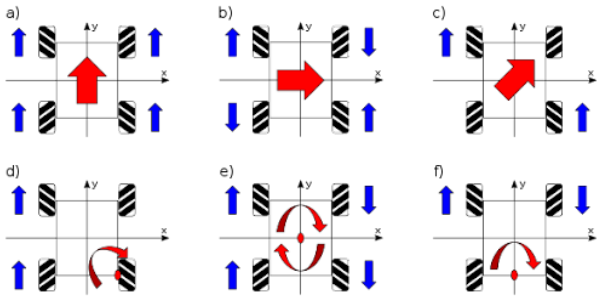

Fig. 3. Four Wheel omnidirectional Movement

$$\begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \\ v_4(t) \end{bmatrix} = \frac{1}{r}\begin{bmatrix} 1 & -1 & -(l_x + l_y) \\ 1 & 1 & (l_x + l_y) \\ 1 & 1 & -(l_x + l_y) \\ 1 & -1 & (l_x + l_y) \end{bmatrix}\begin{bmatrix} v_x(t) \\ v_y(t) \\ w_z(t) \end{bmatrix}$$

$$v_1(t) = v_x(t) - v_y(t) - (l_x + l_y).w_z(t)$$
$$v_2(t) = v_x(t) + v_y(t) + (l_x + l_y).w_z(t)$$
$$v_3(t) = v_x(t) + v_y(t) - (l_x + l_y).w_z(t)$$
$$v_4(t) = v_x(t) - v_y(t) + (l_x + l_y).w_z(t)$$

Where,

(x,y,θ)	= local position of the robot
$(v_1(t), v_2(t), v_3(t), v_4(t))$	= wheels velocities
$(v_x(t), v_y(t), w_z(t))$	= linier velocities on the static axis

3.2 Mapping System

In two-dimensional (2D) space, SLAM algorithms consist of Gmapping, HectorSLAM, Tiny SLAM, Karto, Google Cartographer [2], and others. Gmapping and Cartographer methods are widely used in the robotics community for mapping and localization.
Both Gmapping and Cartographer use sensor data such as lidar, odometry, and IMU, but Cartographer can also incorporate data from cameras and other sensors. Cartographer produces more accurate maps compared to Gmapping due to its advanced algorithm and ability to handle dynamic objects.
Cartographer method breaks the trajectory into pieces called "submaps". Each submap contains a local SLAM result, i.e., a partial map of the environment and the path that the robot took while this partial map was built. As the robot continues to explore its environment, it may revisit areas it has seen before. When Cartographer recognizes a place, it has been previously (known as loop closure detection), it can use this information to correct the robot's path and the map. This is the local optimization.
Once several submaps have been created and connected through loop closures, Cartographer performs a global optimization over the entire trajectory, known as the Pose Graph Optimization. This optimization tries to find a configuration of the robot's positions (poses) that is most consistent with the data and the constraints imposed by loop closures. This means adjusting the poses such that the map sections overlap correctly where they should (at the loop closures), with these adjustments propagated across the whole trajectory to minimize the total error.

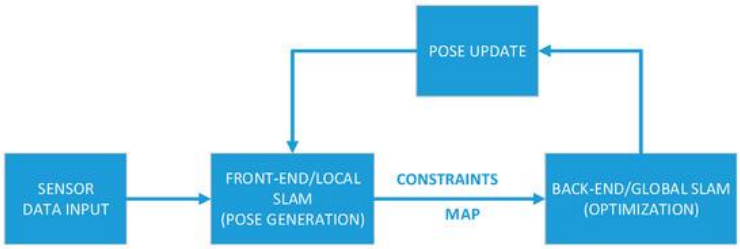

Fig. 4. Mapping Process Diagram

The global optimization process may repeat several times during the mapping process, continuously refining the map and the trajectory. The result is a globally consistent map and trajectory that best represents the robot's path and its environment.
The global optimization in Cartographer uses a technique called sparse pose adjustment (SPA), which is a form of graph-based SLAM. In this approach, the map is represented as a graph where nodes represent poses and edges represent constraints between poses. The goal of the optimization is to find a configuration of the nodes (poses) that minimizes the error imposed by the constraints (edges).

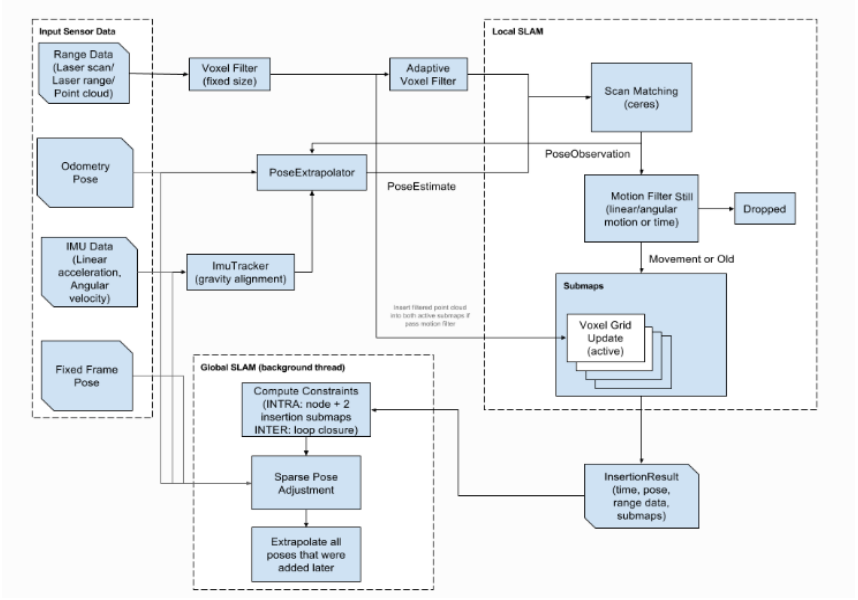

Fig. 5. Cartographer Flowchart

3.3 Positioning System

A positioning system in autonomous robots refers to the capability of determining the robot's position and orientation in its environment. It allows the robot to have situational awareness and make informed decisions about its movements and tasks. There are various techniques and algorithms used in positioning systems, and one commonly used approach in robotics is the Adaptive Monte Carlo Localization (AMCL) [3] algorithm. AMCL is a variant of MCL that adjusts the number of particles based on Kullback-Leiber Divergence Sampling (KLD-Sampling). The algorithm uses a particle filter to represent the probability distribution of the robot's position. The basic idea behind AMCL is to use a set of particles, each representing a possible location of the robot, to estimate the robot's position. The particles are randomly generated and assigned a weight based on how well they match the robot's sensor measurements, such as laser rangefinder or depth camera data. The particles with higher weights are more likely to represent the true position of the robot.

As the robot moves and takes new sensor readings, the particles are updated based on their likelihood of being in the correct position. The particles with lower weights are discarded and replaced with new particles generated from the high-weight particles. This process is repeated until the distribution of particles converges to the true position of the robot.

combining AMCL and odometry can improve the overall positioning accuracy in autonomous robots. Odometry provides estimates of the robot's motion based on wheel encoder readings, but it tends to accumulate errors over time due to factors like wheel

slippage or uneven terrain. On the other hand, AMCL leverages sensor measurements to correct these odometry errors and provide a more accurate pose estimation.
By fusing odometry and AMCL, the robot can benefit from the complementary strengths of both techniques. Odometry provides short-term, low-latency estimates of the robot's motion, while AMCL helps correct and refine the pose estimation over longer periods. This combination allows the robot to have better localization accuracy, especially in scenarios where odometry alone may lead to significant drift or uncertainty.

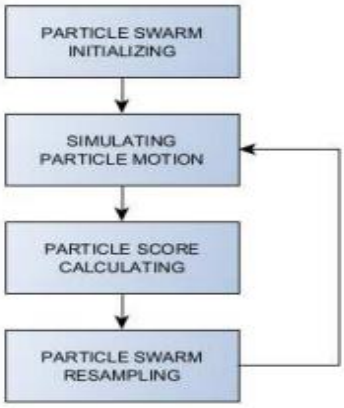

Fig. 6. AMCL Flowchart

3.4 Decision Making

The decision-making process for the robot's movement and object-handling operations relies on a combination of sensor inputs, vision-based algorithms, and pre-programmed strategies. The vision system, utilizing a camera array, captures images of the workspace and processes them through machine learning algorithms to identify objects and their precise positions. Based on this information, the robot's navigation system computes an optimal path to approach the object using the holonomic drive system. The decision to pick up an object is made based on task prioritization and command hierarchy, which may come from a human operator or an external control system.

For material handling, the robot evaluates the object's characteristics such as size, shape, and location using the vision system. This data, along with the robot's internal kinematics, is used to determine the proper orientation and configuration of the robotic arm for grasping and handling. The decision-making framework integrates a feedback loop that continuously updates the robot's understanding of its surroundings, allowing it to adjust its movements and operations in real-time. The system also incorporates safety protocols to prevent collisions and errors during object handling.

4 Testing Workspace

The testing workspace was designed to simulate a typical industrial environment where the robot would operate. It includes several sections for object storage, retrieval, and placement, along with dynamic obstacles to evaluate the robot's navigation and obstacle-avoidance capabilities. Various industrial objects, such as containers, boxes, and irregularly shaped items, were placed in predefined locations to test the vision system's ability to detect and identify them under different lighting conditions.

The workspace also included marked pathways and areas with tight corners to assess the efficiency of the holonomic drive system. The robotic arm was tested by repeatedly performing pick-and-place operations with objects of different weights and sizes, ensuring precision in both grasping and releasing the items. Performance metrics such as task completion time, navigation accuracy, object handling precision, and system reliability were recorded and analyzed. These tests helped optimize the control algorithms and calibrate the hardware components to ensure that the robot could perform reliably in real-world industrial settings.

5 Results and Discussion

One of the most challenging problems in an autonomous robot navigation is the movement system. It is important that the robot movement be exactly as it is wanted in the trajectory that is given to the robot's hardware. In this case, a PID closed-loop controller is designed using the value received from GY-25 as the feedback sensor. In this controller we calculate three error value z1, z2 and z3 as the proportional, integral and differential value and add it with our input signal to motors which in case of residing and rotating ($v_x = 0$ and $v_y = 0$) there is only rotational velocity (w_z) needed. So must have:

$$z_1 = \theta - \theta_d \quad , \quad z_2 = \int_{t=i}^{t=i+dt} z1 \, . \, d\theta \quad , \quad z_3 = \frac{d}{dx} z1$$

In which z_1 is proportional error, z_2 is integral error, and z_3 is the differential error and we can calculate the delta of correction (C_1) value which is:

$$C_1 = K_p \times z_1 + K_i \times z_2 + K_d \times z_3$$

K_p, K_i and K_d are PID controller coefficients which in our case that $d_t = 1ms$ is the period, after experimenting different values for the PID coefficients and with help of Routh-Hurwitz criterion the result came to these numbers:

$$K_p = 1.05 \quad , \quad K_i = 98.5 \quad , \quad K_i = 0.97$$

As mentioned before the conversion method from linear and angular velocity to 4 motor velocity is possible using the function bellow and by placing $v_x = 0$ and $v_y = 0$ values the robot rotate in resident.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \frac{1}{r}\begin{bmatrix} 1 & -1 & -(l_x + l_y) \\ 1 & 1 & (l_x + l_y) \\ 1 & 1 & -(l_x + l_y) \\ 1 & -1 & (l_x + l_y) \end{bmatrix}\begin{bmatrix} v_x \\ v_y \\ w_z \end{bmatrix}$$

To experiment the PID controller we broadcasted the robots heading angle to the computer and recorded the value. Our experiment results for two different set of PID coefficients is shown in Fig. 3. The right graph is for 1000 milliseconds period for integral and differential which has an overshoot that will remain 6 seconds and the settling time will be so long. The left graph is the result for 100 milliseconds period which has the minimum settling time and can be a very nice achievement for this controller. The settling time in this case is less than 0.8 seconds for a big error like 50 degrees deviation.

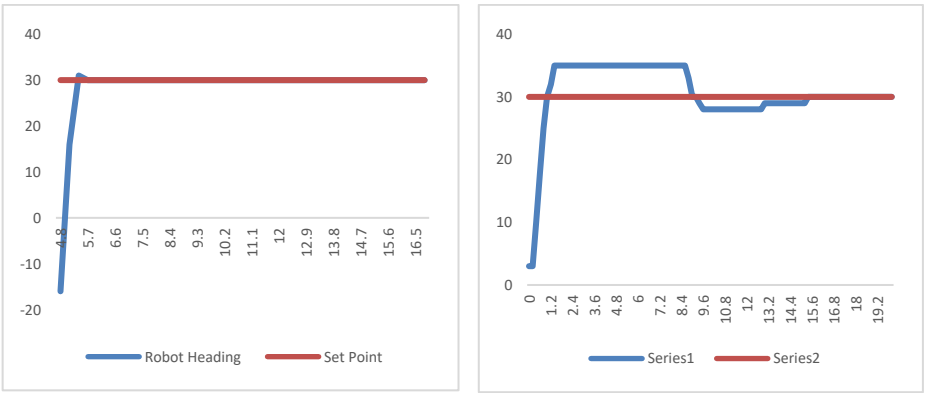

Fig. 7. PID Experiment Recording Results

6 Conclusion

This paper has presented the construction of the four wheeled Omnidirectional mobile robot. It shows the good efficient of SLAM using Cartographer package for building the 2D map. This article also performs successfully the path planning for the Omnidirectional robot under the effect of both the static obstacle and the unknown dynamic obstacle due to the update of the submap generated by local optimization in the simulation and realistic environment. The mapping system uses the google cartographer algorithm combined with eulerdometry, and the positioning system uses the Adaptive Monte Carlo Localization (AMCL) algorithm combined with eulerdometry in order to achieve less errors and high accuracy. At the end we implemented the outputs of the mentioned methods to a PID system for achieving the best performance on the robot. The robot hardware is also built to facilitate the integration of peripherals based on ROS.

References

[1] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng. ROS: An open-source Robot Operating System, in IEEE International Conference on Robotics and Automation Workshop on Open-Source Software (IEEE, 2009).

[2] Dwijotomo, A., Abdul Rahman, M.A., Mohammed Ariff, M.H., Zamzuri, H. and Wan Azree, W.M.H., 2020. Cartographer slam method for optimization with an adaptive multi-distance scan scheduler. Applied Sciences, 10(1), p.347.

[3] Zhang, L., Zapata, R. and Lépinay, P., "Self-adaptive Monte Carlo Localization for Mobile Robots Using Range Sensors," Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009), St. Louis, MO (2009) pp. 1541–1546.Google Scholar

[4] C. He, D. Wu, K. Chen, F. Liu, N. Fan, "Analysis of the mecanum wheel arrangement of an omnidirectional vehicle," Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, vol. 233 no. 15, pp. 5329-5340, DOI: 10.1177/0954406219843568, 2019.

[5] Ji F L, Wang Y J, Li Y, et al. 2009 Static and fatigue analysis of high-pressure vessels based on solidworks simulation Mechanical Engineer (5).

[6] Yin, L., Wang, F., Han, S., Li, Y., Sun, H., Lu, Q., Yang, C. and Wang, Q., 2016, October. Application of drive circuit based on L298N in direct current motor speed control system. In Advanced laser manufacturing technology (Vol. 10153, pp. 163-169). SPIE.

[7] Albaghdadi, A. and Ali, A., 2019. An optimized complementary filter for an inertial measurement unit contain MPU6050 sensor. Iraqi Journal for Electrical and Electronic Engineering, 15(2), pp.71-77.

[8] Alfian, R.I., Ma'arif, A. and Sunardi, S., 2021. Noise reduction in the accelerometer and gyroscope sensor with the Kalman filter algorithm. Journal of Robotics and Control (JRC), 2(3), pp.180-189.

[9] Fedorov, D.S., Ivoilov, A.Y., Zhmud, V.A. and Trubin, V.G., 2015. Using of measuring system MPU6050 for the determination of the angular velocities and linear accelerations. Automatics & Software Enginery, 11(1), pp.75-80.

[10] Mankar, J., Darode, C., Trivedi, K., Kanoje, M. and Shahare, P., 2014. Review of I2C protocol. International Journal of Research in Advent Technology, 2(1).

[11] Mehta, M., 2015. ESP8266: A Breakthrough in wireless sensor networks and internet of things. International Journal of Electronics and Communication Engineering & Technology, 6(8), pp.7-11.