Design of hoeckens linkage based walking robot with MPU6050 IMU as navigation sensor

K R Ooi, M A A Rosli, A R A Latiff, W A F W Othman, S S N Alhady and A A A Wahab

School of Electrical & Electronic Engineering, Universiti Sains Malaysia, 14300 Nibong Tebal, Penang, Malaysia

wafw_othman@usm.my

Abstract. This article introduces a 4-legged walking robot using Hoeckens Linkage. In some applications, a legged robot is preferred if compared with a wheeled robot because it has a more considerable degree of freedom and can move on uneven surfaces. Hoeckens Linkage is one of the linkages that can convert rotational motion to approximate straight-line motion, which allows the robot to move faster than other linkages. This robot is designed using SolidWorks to simulate the Hoeckens linkage path, and its structure is built using 3D printed parts. Arduino Mega was used as a microcontroller to process the feedback signals so that the robot able to move in the desired pattern. The hall effect encoders were used so that DC motors always rotate with the desired RPM regardless of walking surfaces and power supplied to them. MPU 6250 gyro sensor also be used in this walking robot as a navigation sensor. Gyro sensor allows the robot to travel in a straight line or rotate a certain degree by computing yaw value from its raw data and DC motors' control speed using a P controller. The speed average of the fabricated robot is 3.00 m/min.

1. Introduction

In the 21st century, many robots with different functions were invented and used to replace human jobs. The manufacturing industry's human jobs are already replaced with robots because robots can work with higher efficiency and eliminate unnecessary errors. Besides, it could reduce manufacturing costs and also prevent the human from involving themselves in dangerous jobs. Most of the robot today travels on wheels because it is easier to design, fabricate and control. It also consumes less cost than the legged robot, which requires several parts with accurate dimensions to construct its joints. Robots that travel on the wheel can achieve higher travel speed than a legged robot, which needs to consider if they want to increase their travel speed.

However, the legged robot also has its advantages, such as a larger degree of freedom and the ability to move on uneven surfaces. For example, a legged robot is widely used in the mining industry, requiring robots to move within small spaces and rough surfaces. In this situation, the legged robot is preferred compared to the wheeled robot because it has a more considerable degree of freedom that allowed it to move in this condition. With legged robots, these dangerous jobs can be replaced by robots without involving humans. A legged robot is also preferred in few situations, such as building security patrol jobs, because the legged robot is more efficient in this particular application. For
example, if a multi-story building is going to have a security patrol, this job can be done using only one legged-robot because it can move from one floor to another floor, which cannot be achieved using the wheeled robot. In this scenario, the legged robot plays an important role.

There are many types of legged robots developed recently; examples include [1~15]. It can be categorized by the number of limbs that used by the robot. The legged robot with a larger number of legs can be more stable due to its big base and location center of gravity. For a legged robot with a lesser number of legs, stability is the primary consideration when traveling. This is because a legged robot with lesser legs has a smaller base area, so its center of gravity may vary enormously when the robot moves. The number of limbs of this legged robot is four, and stability is one of the design considerations. Leg mechanism is also the consideration of this legged robot when traveling. A few linkages, such as Jenson Linkage [1~6] and Klann Linkage [7~12], are widely used to create walking robots. These linkages convert the rotational motion of the actuator to the path of the linkage. The characteristic of linkage paths affected the travel speed and stability of the legged robot.

2. Methodology

2.1. Robot design

In this work, the 4-legged walking robot is constructed using Hoeckens linkage. This linkage is able to convert the rotational motion of the DC motor to approximately linear motion. It is able to maintain robot body height and increased the efficiency of the motors. Figure 1 below shows the Hoeckens linkage and its path.

![Hoeckens straight line linkage](image)

Figure 1. Hoeckens straight line linkage

The path of the Hoeckens linkage above also indicates the course of the legs of this walking robot. Every walking robot has similar characteristics in its linkage path, which are dragging part and lifting part. To make the robot walk while maintaining its stability, firstly, its legs must contact the ground and drag backward so that the robot body could move forward and follow by leg lifting. The walking robot's efficiency can be increased by reducing the legs' time lifting part or return time. Increasing its horizontal path length also helps to travel faster. The walking robot's stability can be maintained by increases the legs base area and hep of linear characteristics of the Hoeckens linkage path. The linear element of the linkage path could increase robot stability because the robot body would not move up and down while walking so that its center of gravity can be maintained at the same height.

Figure 2 shows the design and overall dimensions of the walking robot using SolidWorks. Its dimension is 300 mm × 302 mm × 140 mm. The length of a is 20 mm (Fig. 1), and after the walking robot's simulated walking mechanism, its horizontal path length of dragging part is 91 mm, and the height of the lifting element is 18mm.

Figure 3 depicts Hoeckens linkage used in the proposed walking robot. As mentioned earlier, the length of link a is set to 20 mm, and all other linkages are calculated based on the ratio shown in Figure 1. The link with 95 mm is called the follower link, transmitting motion from the front linkage that DC motors actuated to the rear linkage. The rear linkage is called coupler linkage, which would follow the front linkage path with the follower's power. It is also used to decrease rotational mobility.
of the legs by add another joint on the legs. The interested reader can refer to Appendix A for the complete sequence of the robot motion.

Figure 2. 4-legged walking robot using Hoeckens Linkage with complete dimension

Figure 3. 4-legged walking robot using Hoeckens Linkage with complete dimension

Figure 4 shows the one the static legs used in this walking robot. This leg is attached to the robot body, and it will move according to the robot body's movement. The static leg is designed to have a larger base area such that the center of gravity of the robot always remains inside the base of support while walking. If the robot's center of gravity cannot stay within the supporting base's area, a moment will be produced and might cause the walking robot to flip over on the ground.

Figure 4. Static leg of the proposed walking robot
2.2. Robot behavior

Figure 5 shows the flowchart of the main function of the proposed walking robot. It consists of five user-defined functions which are \texttt{rpm\_calc()}, \texttt{getyaw()}, \texttt{forward()}, \texttt{stop()} and \texttt{pid\_motor()}. The \texttt{rpm\_calc()} is a timer interrupt function, and it is excluded in the main function's flowchart. The source code started with variable declaration and setup. Its setup consisted of a \textit{P} controller setup, encoders setup, gyro sensor setup, timer interrupt setup, and Arduino \texttt{pinMode} setup. The \textit{P} controller setup is used to set constants for \(K_p\). The encoders setup is to define the correct total pulse per revolution of DC motors' external shaft such that the exact RPM can be determined accurately. Offset values in MPU6050 IMU were defined so that the yaw angle can be calculated correctly. The timer interrupt also needs to be initialized before starting the main program.

![Flowchart of the robot behavior](image)

\textbf{Figure 5: Complete flowchart of the robot behavior}

After the program declaration and setup part, the program will enter a false value because the stable value is initialized as false during the variable declaration phase. After that, the IR sensor reading will
be taken and used to make the following statement decision. If the IR reading was less than 100, the IR sensor is triggered, and the threshold value is increased by 1. This process will continue until the threshold value equals 7, then the robot will start to move.

The IR sensor is used to trigger the robot to launch. This noise filtering technique can eliminate unwanted signals that might disturb the launch of the robot. The IR sensor has to be activated for at least 70 ms continuously to launch the robot. The threshold value will be reset, and the stable variable will become true. It will also enter getyaw() user-defined function to get the current yaw angle and set it as the walking robot's setpoint angle.

Next, the program will execute a true statement because the stable variable turned true during initialization. Motor forward-moving direction setup will be performed in forward() function, and P controller will start to compute the PWM value for enabling pins of L298N motor driver. The robot will move in a straight line according to the yaw angle that is saved during the robot's initialization.

The timer interrupt function is executed at every 5 ms to calculate the current RPM of the DC motors and used by the P controller to compute the new PWM values. The P controller's constant values have to be well-calibrated to achieve a faster transient response and a minimum of overshoot, which would produce vibration due to sudden change in DC motors' speed.

The IR sensor is used as a switch to turn off the robot. The same technique is implemented to reduce unwanted noise. IR sensor has to be triggered for at least 70 ms continuously. After the robot has stopped, the robot is reset to its initial condition, waiting for the IR sensor's trigger signal to launch it. Before starting the robot, the robot heading can be adjusted manually. The robot will move in that predefined direction using the gyro sensor and the P controller.

This rpm_calc() function is a function that is triggered by the timer interrupt (Figure 6). The timer is set to 5 ms, which means this rpm_calc() function will execute at every 5 ms. This function is to calculate the current RPM of both DC motors. And the calculated rpm will be used in pid_motor() function to compute the PWM values that should send to enable pins of the motor driver. The formula for motor RPM calculation is \{difference of pulses × 60\} / \{total pulse per complete revolution of output shaft × timer interval\}. The total pulse per complete revolution of output is equal to 1440 pulses, and the timer interval is 5 ms. 60 is included in the calculation to convert revolution per second to revolution per minute (RPM). Before the calculation part started, the program will update the current pulse of both motors. After calculating new RPM values, the current pulse values will overwrite into previouspulse1 and previouspulse2 variables used in the next cycle of calculation.

Figure 7 shows the flowchart of getyaw() function. This function is used to compute the yaw value of the robot heading from the MPU6050 gyro sensor's raw data. The calculated yaw will be set as a setpoint in the P controller so that the robot heading is always directed to its initial direction. There are 6 data get from MPU6050 in the initial part of this function, and these data are saved in variables of ax, ay, az, gx, gy and gz. However, the computation of the yaw of the robot only involves gz variable. Yaw value is equal to the summation of the previous yaw value and \( a_y a w \) where \( a_y a w = (g_z × 0.045) / 131 \). A constant of 131 is divided in the calculation to convert raw data of gz to a degree per second since 131 counts for each degree per second at the most sensitive setting of the MPU6050 gyro sensor. A constant of 0.045 is used to minimize the effect of the drift situation in the yaw value.

The forward() function is used to send a forward movement signal to the motor driver's direction pins. This function is executed before the robot started moving forward. For the robot to move in the forward direction, the left motor should rotate in a clockwise direction. In contrast, the right motor rotates in a counter-clockwise direction because they are symmetrical to each other.

The stop() function is executed when an IR sensor triggers the stop signal. During the stop() function, both DC motors will stop from rotating by sending a PWM value of 0 to enable the motor driver's pins. And small delay also will be executed after this to eliminate unwanted noise from the IR sensor.

Figure 8 is flowchart of pid_motor() function. This function is executed during the moving of the proposed robot. It will call getyaw() function to compute for the error of the robot heading. The P controller for left and right motors will then be computed to adjust the robot heading to its initial
direction for the robot to move in a straight line. After that, the $P$ controller for motor speed control will compute PWM values for both motors to achieve the speed required to adjust the robot heading to its initial heading. Then, the PWM value for both motors will be sent to enable pins of the motor driver. $P$ controllers of RPM required and motor PWM values will adjust the robot to its initial heading with the fastest transient response and minimum of overshoot situation.

2.3. Circuit diagram

Figure 9 depicts a complete circuit diagram of the proposed walking robot. It consists of six components: Arduino Mega 2560, MPU6050 inertial measurement unit, infrared sensor, 18650 rechargeable batteries, geared DC motor with hall effect encoders L298N motor driver. Arduino Mega was used as a robot microcontroller to process the signal from the input sensor and feedback to actuators such as DC motors. It is also used for the $P$ controller to compute error and input back to the actuator.

MPU6050 is used as a navigation sensor for this robot to walk in a straight line regardless of the environment's disturbance and power. FC Inter-Integrated Circuit, FC protocol was used to interface MPU6050 IMU with Arduino Mega. The raw data from MPU6050 is computed to get the robot's yaw value before moving and traveling. The advantage of FC over other protocols is that it only required
two wires to interface several types of sensors with the microcontroller. Other devices will have their address frame, and they can be accessed separately by just comparing their address. AD0 pin of MPU6050 is connected to 0 V, and its address is set as 0x68.

![Figure 9. Complete schematic diagram of the proposed walking robot](image)

The infrared sensor signal is connected to an analog pin even though the IR sensor is a digital sensor. IR sensor will send a signal of 0 V when triggered by any object and send a signal of 5 V when nothing is approaching. The analog input is used with an IR sensor to filter out noise signals from the environment. There are three pieces of 18650 LiPo batteries connected in series to provide a nominal voltage of 11.1V to the robot. When the batteries are fully charged, their combined output voltage reaches 12.6 V. This output voltage was connected to the Vin pin of Arduino Mega and L298N to power up them. Arduino Mega on-board linear voltage regulator AMS1117 5.0 will step down 12.6V of input voltage to 5V to power up the microcontroller.

L298N is a dual channel motor driver with a peak current of 2 A. It is powered by 12.6 V of DC voltage, and it would supply this voltage to DC motors according to signals received. Enable pin controls duty cycle of the duty ratio of output voltage, and direction pin controls the direction of current flow in DC motors. It also has an on-board linear voltage regulator LM7805, which can be used to power up the microcontroller that required 5 V to operate. The DC motor model is SPG30E-150K with a rated speed of 26 rpm and torque of 588 mNm. It comes with a hall effect encoder in its rear part that is used to compute motor rotation speed. This encoder has a resolution of 3 pulses per rear shaft revolution, and it has two digital outputs. This motor's gear ratio is 1:120, so its maximum encoder output resolution is 1440 pulses per main shaft revolution.

3. Results & Discussion

Figure 10 shows the fabricated prototype of the proposed 4-legged walking robot. The desired speed for DC motors of this walking robot is 33 revolutions per minute, and in one complete revolution of the motors, the robot can travel up to 91 mm. In perfect condition, this walking robot would travel at a speed of 3.00 m/min. An experiment was carried out to calculate the actual travel speed of this robot. Table 1 shows the result of this experiment.

After the experiment, the average walking speed of this robot was deemed at 2.43 m/min. The average speed is less than theoretical speed due to uneven walking surface, time-taken in transient response, deviation of robot heading and time-taken for adjustment, the slipping situation between the foot and the ground, and some other environmental factors. The fabricated walking robot's travel speed...
can be improved by increasing the DC motor's speed so that a higher number of Hoeckens Linkage cycles can be completed within one minute.

![Fabricated Hoeckens linkage based walking robot complete with MPU6050 IMU](image)

**Figure 10.** Fabricated Hoeckens linkage based walking robot complete with MPU6050 IMU

| Attempt | Displacement (m) |
|---------|-----------------|
| 1       | 2.44            |
| 2       | 2.49            |
| 3       | 2.36            |
| 4       | 2.40            |
| 5       | 2.47            |
| 6       | 2.49            |
| 7       | 2.36            |
| 8       | 2.40            |
| 9       | 2.47            |
| 10      | 2.44            |

The power consumption of this walking robot is calculated by measuring its input current. In normal condition, the current measured is 0.46 A with fully charged batteries. But in the condition that the motor needs to produce larger torque, its input current will increase to 1.3 A. So, the estimated power consumption can be calculated from the multiplication of input voltage and input current, 12.60 V and 0.46 A, respectively. The estimated power consumption of this walking robot was 5.80 W in regular operation. The robot’s weight included batteries, is 1.4 kg.

**4. Conclusion**
The designed robot was able to walk in a straight line. Using of Hoeckens Linkage increases walking efficiency due to the linear characteristic of its linkage path. The forward time of this linkage is also larger than the return time of this linkage. Forward time over the return time ratio of this linkage is
2:1, which means 66.7% of the motor power is used in dragging motion and shortens the time taken in a lifting motion.

IR sensor also functioning well for triggering starting of the robot. There is no noise at all for the robot triggering part. Using MPU6050 helped the robot navigate itself in the correct direction regardless of environmental factors and its power. P controller shortened the transient response time and increased the overall efficiency of the robot. Overshoot situation also minimized with the proper P controller constant tuning. Other than that, the robot could also travel in a different direction by changing its yaw setpoint. Hall effect encoders also help the motor rotate at the desired speed, making both motors move at the same speed and achieve perfect straight-line motion of this walking robot.

Furthermore, the robot can also be improved by increasing its travel speed. It can be done by replacing the motors with a high-power motor that can rotate at a higher rate and produce larger torque. Travel speed also can be increased by solving the constraint problem in the mechanism part. This problem can be solved by adding bearing in each of the joints and using shafts with high accuracy in their diameter instead of using screws to construct the linkage's joint. This would eliminate tolerance in the dimension of the screw. With these modifications, the robot would be able to travel at a higher speed and obtain higher efficiency since it is moving all the time.

Acknowledgements
This work was supported by Universiti Sains Malaysia - RU Grant Scheme (Grant number: USM/PELECT/8014113).

Appendix A
Sequence of stages in the robot motion.
References

[1] Singh R and Bera T K 2020 Walking model of Jansen Mechanism-based quadruped robot and application to obstacle avoidance Arabian Journal for Science and Engineering 45(2) pp 653-664

[2] Vujošević V, Munović M, Tomović A and Tomović R 2018 Robot based on walking Jansen mechanism In IOP Conference Series: Materials Science and Engineering 393(1) p 012109

[3] Bhavsar K, Darji P, Gohel D, Modi J and Parmar U 2021 Comparison of Kinematic Analysis of Robot Made of Conventional Theo Jansen Mechanism, Modified Theo Jansen Mechanism of PLA and Modified Theo Jansen Mechanism of Mild Steel In Recent Trends in Mechanical Engineering (Singapore: Springer) pp 545-559

[4] Roslee H H, Tew J C, Ismail M A U, Ahmad-Adli A A, Othman W A F W, Alhady, S S N and Wahab A A A 2021 Development of Theo Jansen Inspired All-Terrain Quadruped Mini Mobile Robot In Proceedings of International Conference on Intelligent Robotics, Mechatronics and Automation Systems

[5] Bhavsar K, Gohel D, Darji P, Modi J and Parmar U 2020 Kinematic Analysis of Theo Jansen Mechanism-Based Eight-Leg Robot In Advances in Fluid Mechanics and Solid Mechanics (Singapore: Springer) pp 75-82

[6] Pop F, Lovasz E C, Pop C, Dolga V and Grigorescu S M 2017 Trajectory Analysis for Modified Jeng Leg Mechanism Configuration In New Advances in Mechanisms, Mechanical Transmissions and Robotics (Cham.: Springer) pp 409-416

[7] MohdIsharudden F, Mohamed H, Rafaai Z M, Ho T Y W and Kamarudin M S 2020 Design and Prototyping of a Motorized Legged Robot with Klann Linkage Mechanism International Journal 8(5)

[8] Gutarra A, Palomino S and Alegria E J 2021 Hexapod Walking Mechanism Based on the Klann Linkage for a 2DoF Amphibious Robot In International Symposium on Multibody Systems and Mechatronics (Cham.: Springer) pp 302-310

[9] Mahendran K, Zhi H L, Yazid M R, Othman W A F W, Wahab A A A and Alhady S S N 2021 Development of Four-Legged Klann Linkage Walking Robot In 11th International Conference on Robotics, Vision, Signal Processing and Power (Singapore: Springer)

[10] Xu W Q, Yan H and Zhang W H 2018 Simulation-based optimization on multi-legged walking robot based on Klann mechanism Journal of Machine Design 05

[11] Sheba J K, Elara M R, Martinez-Garcia E and Tan-Phuc L 2017 Synthesizing reconfigurable foot traces using a Klann mechanism Robotica 35(1) 189

[12] Martinez-Garcia E A, Dominguez J and Lavrenov R 2019 Dynamic modelling and control of an underactuated Klann-based hexapod In 12th International Conference on Developments in eSystems Engineering (IEEE) pp 64-69

[13] Tee H S, Mohd-Idros M A, Gunasegaran K, Othman W A F W, Alhady S S N and Wahab A A A 2021 Obstacle Avoiding 4-Legged Mobile Robot using 4-Bar Mechanism In Proceedings of SympoSIMM 2020 (Singapore: Springer)

[14] Wong L H, Sivanesan S, Ahmad Faisal M F, Othman W A F W, Wahab A A A and Alhady S S N 2021 Development of Quadruped Walking Robot with Passive Compliance Legs using
XL4005 Buck Converter In Proceedings of International Conference on Intelligent Robotics, Mechatronics and Automation Systems

[15] Lee C Y, Abdull-Sharif M H, Vinayagan S, Othman W A F W, Alhady S S N and Wahab A A A 2021 Design and Development of a Quadruped Shuffling Mobile Robot In Proceedings of International Conference on Intelligent Robotics, Mechatronics and Automation Systems