Computer Engineering and Applications Vol. 9, No. 1, February 2020

# The Application of Push Button Switch as Inverse Kinematics Input on Adaptive Walking Method for Hexapod Robot

Pola Risma, Muhammad Bagaskara, Nyayu Latifah Husni, Masayu Anisah, Adella Rialita

*Department of Electrical Engineering, Mechatronics Study Program, State Polytechnic of Sriwijaya*
**Correspondence: nyayu_latifah@polsri.ac.id*

## ABSTRACT

In general, the occurrence of any kinds of natural disaster are undesirable. When they happen, there will be a lot of losses and damage. It is surely not an easy matter. It cannot be denied that property and people have to be relieved. In many cases, the misfortune usually was caused by buildings ruin. Some of the victims may still be alive and need helps as soon as possible, however, this is too risky for the rescue team since the location is still in dangerous level. Therefore, in this research, the hexapod robot detector was proposed. This robot has a function to replace the tasks of the rescue team in searching for the victims of the disasters in order to decrease the victims' number. The hexapod robot in this research is a six-legged robot shaping like a spider. This research focuses on the analysis of the push button switch as a robotic foot control input. This is because walking technique is an effective major factor in navigation of robots. A good method is required to maintain the height of the robot's foot while it is walking. Therefore, to overcome this problem, the push button switch application is used along with the inverse kinematics calculations on each routine program in order to adjust the position of the end effector on the floor surface. In shifting, the navigation runs well without any failure if the position of the foot does not touch the floor. The test is done in 2 steps, comparing the inverse kinematics calculations with x and y inputs which are applied to the robot program code then comparing the travel time condition by using push button switch and without push button switch. The result of robot in this study can be re-developed in the future, using servos with greater torque and better control input than push button switch.

**Keywords**: Hexapod Robot, Arduino Mega 2560, Inverse Kinematics.

## 1. INTRODUCTION

Indonesia is a natural disaster-prone country, regarding to pacific ring of fire. One of the natural disasters which is often happen is earthquake. Earthquakes are natural disasters that cannot be avoided and can happen anywhere. The bad effects caused by earthquakes include the collapse of buildings and loss of life and property. Among the deaths were caused by the ruin of the buildings. Some victim may still be alive and need helps as soon as possible, however this is too risky for the rescue team since the location is still in dangerous level. Therefore, the hexapod robot is proposed in this research in order to replace the tasks of the rescue teams in searching for the victims of the disasters, so that there will be no more victims left underground of the ruin.

Some of the studies that have discussed about the Hexapod Robot in detecting the victims of natural disasters are as follows: i) In paper by S. Rathnaprabha [1], it discussed the movement of hexapod robots that can move forward and backward. This robot cannot run on uneven roads, because of the lack of sensors used; ii) In paper by Marek Zak [2], it discusses the movement of hexapod robots which use a gait tripod so that the hexapod robots can move on all surfaces, both flat and uneven. This robot uses a force-sensitive sensor that is useful as a component to detect foot pressure input; iii) In paper by Tolga Karakurt [3], it discusses the control system of hexapod robots using remote controls and camera sensors that are useful to function as human detector.

Based on these studies, there are still many shortcomings. Robots can run on uneven surfaces, however it still uses rare components that cannot be obtained easily, such as force-resistive sensors. There are also some robots that have not applied the adaptive walking system. The Adaptive walking is an attempt by robots to improve their walking motion when they are walking on uneven surface conditions or when avoiding wall disturbances. In this research, the robot that is proposed to overcome this problem is a hexapod robot that detects victims of natural disasters using a switch button as an indicator of touching the foot. The use of ‘a switch button as an indicator of touching the foot’ in this research indicates that the robot can continue to run well by using components that are easy to find. The navigation system for hexapod robots to detect victims of natural disasters in this research utilizes the fuzzy logic. Each hexapod robot has 3 Degrees Of Freedom (DOF). By using fuzzy logic, robots can work automatically in correcting the wrong motion when run in uneven or many wall conditions. Servo used in hexapod robots that detect natural disaster victims, namely hitec 422 servo. The speeds are made varying depending on the terrain that will be faced. When it is applied on the flat surface, the hexapod robots will move quickly and when it is applied on uneven surfaces, the robot moves slowly. The push button switch application is used along with the inverse kinematics calculations on each routine program in adjusting the position of the end effector on the floor surface. The microcontroller which is used has functions as detector of the victims, is Arduino Mega2560. It is used due to many input and output ports are needed on this robot.

## 2. METHODOLOGY

### 2.1. HEXAPOD ROBOT

As the name implied, hexapod robot is a type of robot that moves using 6 (six) feet and has high flexibility. It can move freely since the body shape supports the ability to move. All system will be adjusted to the terrain that will be faced by the hexapod robot. It must be in accordance with the task to be carried out later. The robot has 6 feet where each leg has 3 Degrees of Freedom. Each robot's leg consists of 3 servo which allow movement on 3 axes, i.e. i) back and forth, ii) up and down and iii) left and right. The first servo which is attached to the body of the robot is used to make the end-effector move back and forth on the z axis. The second servo which is attached to the first servo is used to make the end-effector rise and fall, while iii) the third servo which is attached to the foot is used to make the end-effector get close or away from the robot body. The end-effector on the robot's feet is marked with black rubber and a push button switch located on the foot of the robot that is used to help the robot run so it doesn't slip on the slippery floor.

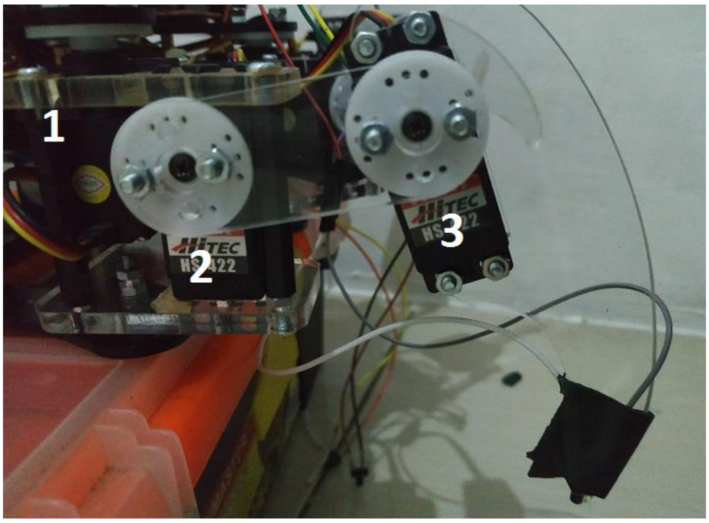


FIGURE 1. Robot’s feet structure

The first and second servo are installed coincidentally so that the middle axis distance between the servo is 2 cm. The second and third servo are associated with a link with a total length of 8.5 cm and the distance between the second and third servo centers is 4 cm. The third servo is directly related to the end-effector. The distance between the third servo center and the end-effector is 7cm (see Figure 1).

## 2.2. FLOWCHART

The flowchart of the robot proposed in this research can be seen in Figure 2. When the robot detected that there is obstacles, the robot will try to avoid it.  When it does not detect the obstacle, it will activate the TPA81, when it detects the object, it will move to the object.  When it does not detect the object, it will continue to activate the TPA81.  When it moves to the object and indicating the object fully, then it will stop moving and send the GPS coordinate to the server.  However, when it still cannot fully find the object it will still keep moving as can be seen in the flowchart presented in Figure 2.

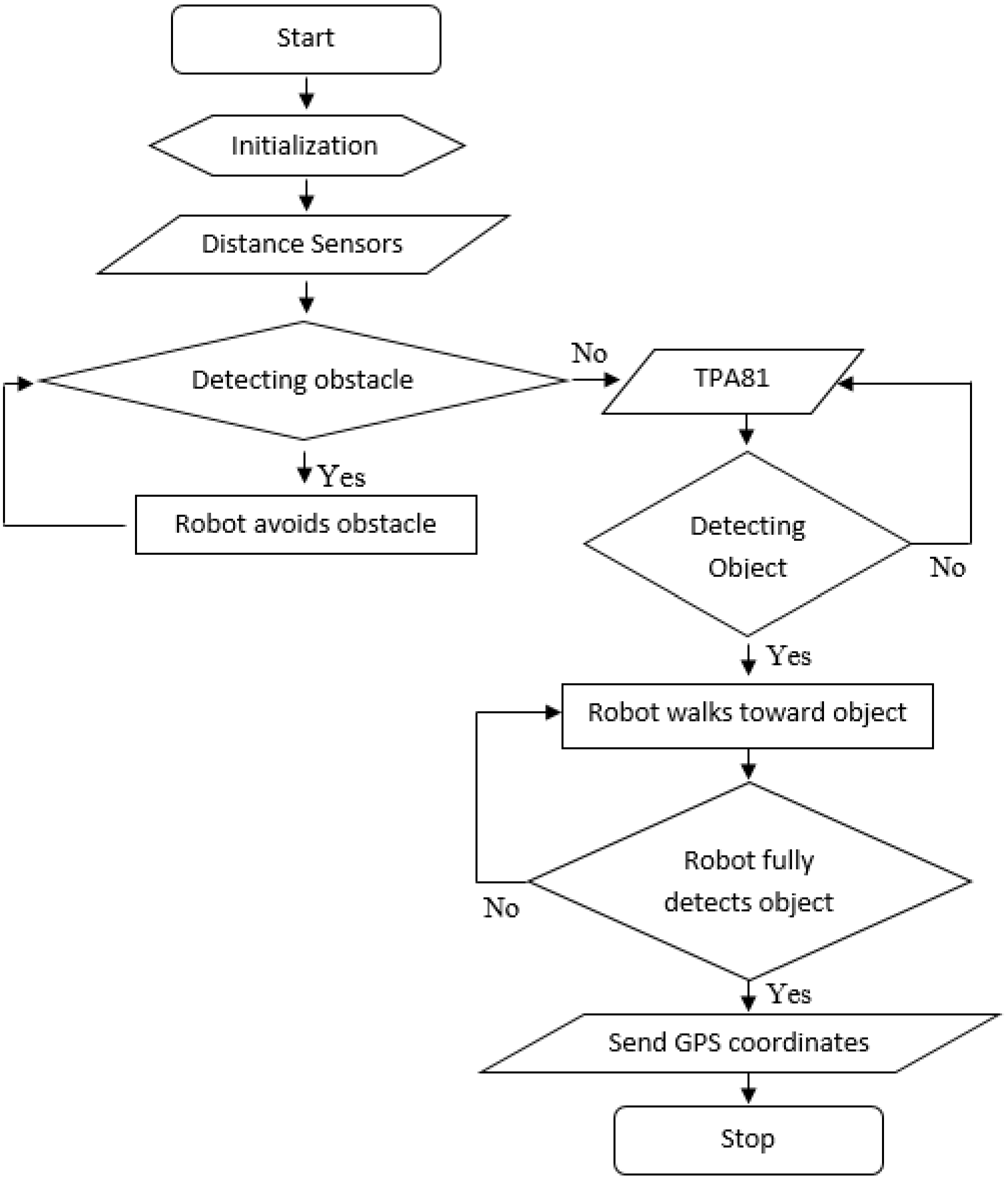


FIGURE 2. Flowchart

## 2.3. MECHANICAL DESIGN

The mechanical design of the proposed robot in this research involves some components, such as: i) 422 hitec servo motor, ii) pir sensor, iii) landfill sensor, iv) gps module, v) sim900A, vi) ultrasonic sensor and vii) arduino mega2560. The casing of the robots in this research is described in the next section.

### 2.3.1. HEXAPOD ROBOT BODY DESIGN

In this step, a hexapod's body design that has size 20 cm x 20 cm was made using Corel Draw X7. It uses acrylic with a thickness of 3 mm as its material. The body design of this hexapod robot can be seen in Figure 3.

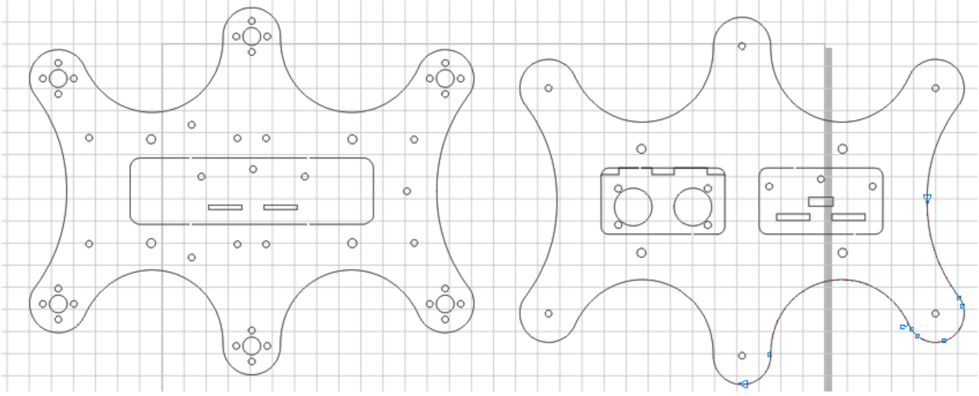
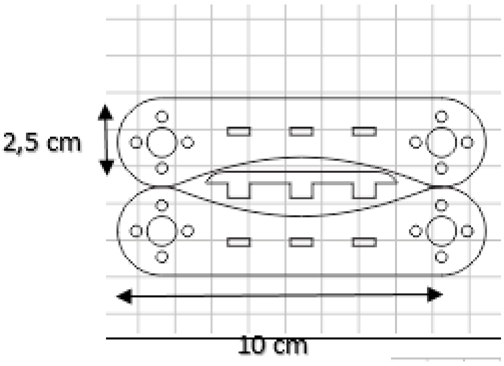


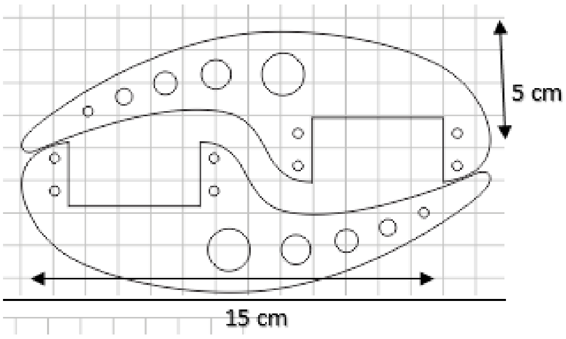
FIGURE 3. Hexapod body design

## 2.3.2. HEXAPOD ROBOT FOOT DESIGN

In this step, a leg design was made using Corel Draw X7. It uses 3 mm thick acrylic as its material. The leg design can be seen in Figures 4 (a) and (b).



(a)



(b)

FIGURE 4. Leg Design, (a) Femur design, (b) Tibia design

### 2.3.3. ENTIRE DESIGN

In this step, the entire mechanical hexapod robot is assembled based on the design that has been made with a predetermined size to fit the working criteria of the robot as presented in the section 2.3.1 and 2.3.2. Figure 5 shows the whole mechanical design of a hexapod robot in this research.

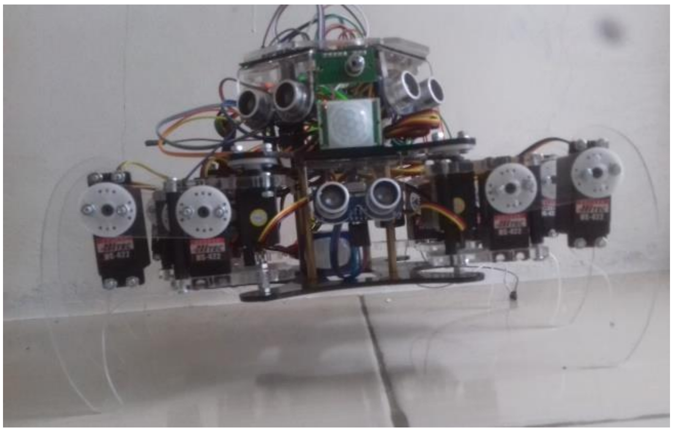


FIGURE 5. Hexapod robot

## 2.4. INVERSE KINEMATICS DESIGN

Inverse Kinematics is the use of a robot kinematics equation to determine the angles needed at each joint which give the desired position at the end effector position. A method is used to make a robot can adjust its path to the surface it is passing through. In general, this method uses the function of a pressure sensor. The pressure sensor can be replaced with several other types of sensors, such as push-button switches and limit switches. This method works by moving the robot's feet first, then detecting the touch on the robot's feet. If the robot's feet have not touched the surface, the robot will continue to lower its legs, however when the robot's feet have touched the surface, the robot will stop lowering the legs and adjusting the next step (see Figure 6).

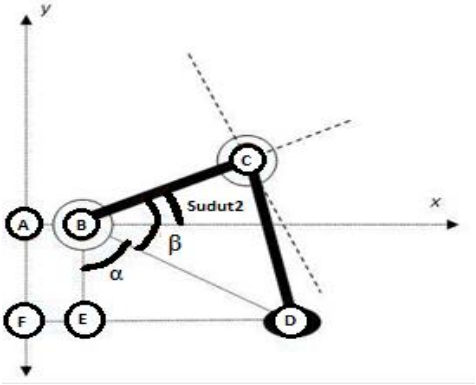


FIGURE 6. Feet model projected in X and Y axis

Inverse Kinematic applied to hexapod robots only includes the movement of the end-effector using X and Y coordinates. Point A is a servo 1, B is a servo 2, C is a servo 3, and D is an end-effector. The link between servo1 and servo2 is called coxa. The link between servo2 and servo3 is called the femur. The link between the servo3 and the end-effector is called the tibia.

To determine the X and Y coordinates, it can be obtained by analyzing Figure 2. Forward kinematics formula can follows the equation (1) and (2):

$$x = coxa + \ BD \sin \alpha \tag{1}$$

$$y = BD \cos \alpha \tag{2}$$

The angle 2 in FIGURE 2 of this research is the magnitude of the angle that will be applied to the servo2 (see equation (3)). Two triangles, namely the triangle BDE (presented in Figure 7) and BCD (presented in Figure 8) can help to solve the problem.

$$Angle2 = (\alpha + \beta) - 90° \tag{3}$$

To find the value $\alpha$ can be viewed from the triangle BDE.

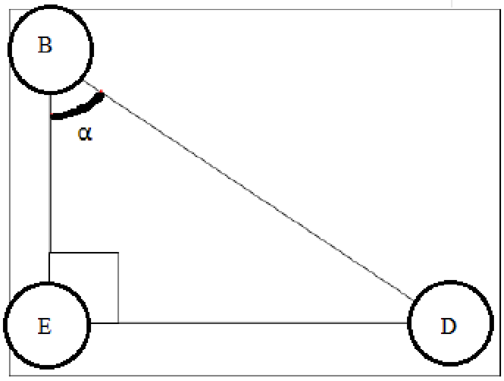


FIGURE 7. BDE triangle

BDE triangle is a right triangle, therefore to find the angle α, it can use the tangent formula from trigonometry as presented in equation (4) and equation (5):

$$\tan \alpha = \frac{DE}{BE} \tag{4}$$

$$\alpha \ = tan^{-1} \frac{DE}{BE} \tag{5}$$

Where DE can be determined by using equation (6):

$$DE = DF - EF \tag{6}$$

From the BDE triangle shown in Figure 7, it can also be determined by the distance between point B to point D using the phytagoras formula, where BD is the longest side or hypotenuse and DE and BE is the right side. Therefore, the equation will be presented as in equation (7)

$$BD = \sqrt[2]{DE^2 + BE^2} \tag{7}$$

Then, it can be seen that BCD triangle is an arbitrary triangle (see Figure 8).

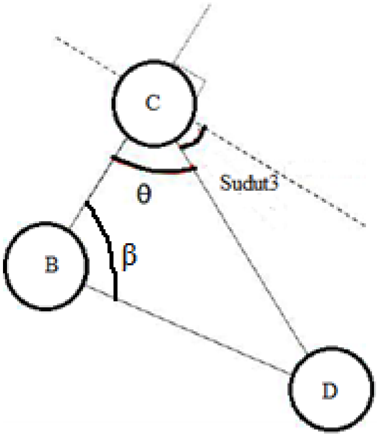


FIGURE 8. BCD triangle

From the triangle BCD, the three sides that are known, are BC, CD and BD. Thus, to find the angle β, it can use the cosine rule formula as in equation (8) and equation (9).

$$\cos\beta = \frac{BC^2+BD^2-CD^2}{2.BC.BD} \tag{8}$$

$$\beta = arc\cos\left(\frac{BC^2+BD^2-CD^2}{2.BC.BD}\right) \tag{9}$$

With X and Y axis analysis, values can also be determined from angle 3 with the help of BCD triangle. From Figure 8, it can be seen that the magnitude of the angle 3 and the angle θ form 90 ° angle, so that it can be written as shown in equation (10):

$$Angle3 = 90° - \theta \tag{10}$$

To find the value of θ, it can be seen from the BCD triangle, which is an arbitrary triangle that is known to all sides. The value of θ can be searched using the cosmic rule formula of trigonometry as presented in equation (11) and (12):

$$\cos\theta = (\frac{BC^2+CD^2-BD^2}{2.BC.CD}) \tag{11}$$

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

$$\theta = arc \cos \left(\frac{BC^2+CD^2-BD^2}{2.BC.CD}\right) \quad (12)$$

## 2.5. FUZZY LOGIC DESIGN

The fuzzy logic is used as the artificial intelligence in this hexapod robot. In general, the fuzzy logic can be define as a method that is used for transferring or imitating human knowledge into a machine. There are two types of fuzzy that are commonly used, namely mamdani and sugeno. Fuzzy logic method on hexapod robot that detects victims of natural disasters that will be created in the scilab 5.5.2 application is the mamdani type of fuzzy logic. The rule for the fuzzy logic of this research can be seen in Table 1.

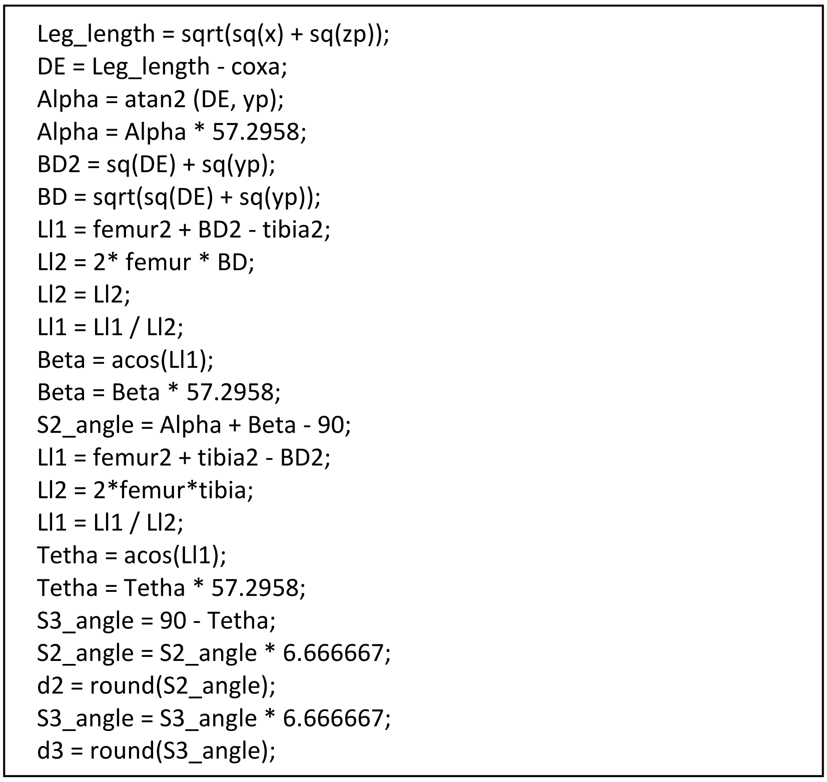
TABLE 1.
Fuzzy Rule

| Input | | | | Output | |
|---|---|---|---|---|---|
| Left Ultrasonic | Front Ultrasonic | Right Ultrasonic | TPA81 | Navigation | GSM |
| Range>15cm | Range>15cm | Range>15cm | Detect | Forward | Off |
| Range>15cm | Range>15cm | Range>15cm | Not detecting | Forward | Off |
| Range>15cm | Range>15cm | Range≤15cm | Detect | Turn left | Off |
| Range>15cm | Range>15cm | Range≤15cm | Not detecting | Turn left | Off |
| Range>15cm | Range≤15cm | Range>15cm | Detect | Forward | Off |
| Range>15cm | Range≤15cm | Range>15cm | Not detecting | Stop | Send |
| Range>15cm | Range≤15cm | Range≤15cm | Detect | Turn left | Off |
| Range>15cm | Range≤15cm | Range≤15cm | Not detecting | Stop | Send |
| Range≤15cm | Range>15cm | Range>15cm | Detect | Turn right | Off |
| Range≤15cm | Range>15cm | Range>15cm | Not detecting | Turn right | Off |
| Range≤15cm | Range>15cm | Range≤15cm | Detect | Forward | Off |
| Range≤15cm | Range>15cm | Range≤15cm | Not detecting | Forward | Off |
| Range≤15cm | Range≤15cm | Range>15cm | Detect | Turn right | Off |
| Range≤15cm | Range≤15cm | Range>15cm | Not detecting | Stop | Send |
| Range≤15cm | Range≤15cm | Range≤15cm | Detect | Backward | Off |
| Range≤15cm | Range≤15cm | Range≤15cm | Not detecting | Stop | Send |

## 2.6. PROGRAM CODE

### 2.6.1. INVERSE KINEMATICS

In its calculations, kinematic inverses require inputs of 3 (three) variables, including the position of the feet on the x, y and z axes. The declaration of kinematic inverse code as a sub-program should be designed immediately, including a declaration for these 3 (three) variables. In reaching a point, good precision is needed. Therefore, input can be a decimal number, so that the declarations of variables x, y and z are single data type. The calculation of kinematic inverse formulas must have the right rules because the program will execute the program line from the top. In this formula there are variables that are used in parallel. Therefore, if the position of the calculation sequence is incorrect, it will cause chaos on the results of the calculation. See Figure 9 for knowing the code used in this research.



```
Leg_length = sqrt(sq(x) + sq(zp));
DE = Leg_length - coxa;
Alpha = atan2 (DE, yp);
Alpha = Alpha * 57.2958;
BD2 = sq(DE) + sq(yp);
BD = sqrt(sq(DE) + sq(yp));
Ll1 = femur2 + BD2 - tibia2;
Ll2 = 2* femur * BD;
Ll2 = Ll2;
Ll1 = Ll1 / Ll2;
Beta = acos(Ll1);
Beta = Beta * 57.2958;
S2_angle = Alpha + Beta - 90;
Ll1 = femur2 + tibia2 - BD2;
Ll2 = 2*femur*tibia;
Ll1 = Ll1 / Ll2;
Tetha = acos(Ll1);
Tetha = Tetha * 57.2958;
S3_angle = 90 - Tetha;
S2_angle = S2_angle * 6.666667;
d2 = round(S2_angle);
S3_angle = S3_angle * 6.666667;
d3 = round(S3_angle);
```

FIGURE 9. Inverse Kinematics code

Information:

| | |
|---|---|
| Leg_length | = Leg length from X and Z axis projections. |
| Zp / Z | = Z coordinates. |
| Yp / Y | = Y coordinates. |
| Xp / Y | = Coordinate X. |
| DE | = DE side length. |
| BD | = BD side length. |
| BD2 | = BD side length (squared). |
| Coxa | = Coxa length. |
| Femur | = Length of the femur. |
| Femur2 | = Length of the femur (squared). |
| Tibia | = Length of tibia. |
| Tibia2 | = Length of tibia (squared). |
| S1_angle | = Servo angle 1. |
| S2_angle | = Servo angle 2. |
| S3_angle | = Servo angle 3. |
| Ll1, Ll2 | = Formula assembly variable. |

d1, d2, d3 = Servo angles 1,2 and 3 which have been used as pulses.

In making the inverse kinematics program, several mathematical functions are used in Arduino, such as: i) Atan2 is a function of inverse tangent, ii) Acos is a function of inverse cosine. The value of 57.2958 is a multiplication constant to find the angle of radians. The use of Ll1 and Ll2 is intended to make it easier to understand the process of calculating inverse kinematics. D1, d2 and d3 use a round function to round up the value obtained, because the servo pulses cannot have decimal fractions. The value of 6.6667 is the value per degree on the Hitec HS-422 servo, where the range of the servo pulses in the range 0 ° - 180 ° is 900 to 2100.

### 2.6.2. PUSH BUTTON SWITCH LOOPING

To use the push button switch as a feedback input, several lines of code are used (See Figure 10).

```
void cek_switch1(int s, int serf, int sert){
  ix = 7.0;
  iy = -10.3;
  switch1 = digitalRead(s);
  while(switch1 == 1){
    cekgps();
    invers(ix, iy, 0);
    ix = ix - 0.1;
    iy = iy - 0.2;
    pik2 = 1500 - d2;
    pik3 = 1500 - d3;
    if(pik2 <= 900){
      pik2 = 900;
    }
    if(pik2 >= 900){
      pik2 = 900;
    }
    if(pik3 >= 2100){
      pik3 = 2100;
    }
    if(pik3 <= 900){
      pik3 = 900;
    }
    serv(serf, pik2);
    serv(sert, pik3);
    switch1 = digitalRead(s);
    spd(10);
  }
}
```

FIGURE 10. Push button switch code

Information:
S = The push button switch used.
Serf = The femoral servo number used.
Sert = The tibia servo number used.

Ix, iy = The ideal end-effector coordinate.

When stepping on foot, the robot will wait for a response from the switch push button input whose initial input is 1 (high) and when it is pressed, it will give input 0 (low). This is done by repeating the push button reading and changing the servo position until the push button is pressed. The while command will repeat the reading when the push button condition is 1 (high). The position of the end-effector coordinate in the calculation will change by -0.1 every while in the while program.

## 2.7. DATA AND CALCULATIONS

### 2.7.1. CALCULATION RESULTS OF INVERSE KINEMATICS

Before applying the full program algorithm to the robot, testing was done on the calculation of inverse kinematics. Calculation of inverse kinematics on robot feet has a reference to the midpoint value for each coordinate. The X value is 7 and Y is -10.3. Table 2 is the calculation data for servo angle 2 and servo 3 that used X and Y coordinates.

TABLE 2.
Angles calculation of servo 2 and 3 using X and Y coordinates.

| Trial | X input | Y input | Servo2 | Servo3 |
|---|---|---|---|---|
| 1 | 7.5 | -9.8 | 41.38 | 21.85 |
| 2 | 7.4 | -9.9 | 33.28 | 17.81 |
| 3 | 7.3 | -10 | 25.09 | 13.60 |
| 4 | 7.2 | -10.1 | 16.81 | 9.23 |
| 5 | 7.1 | -10.2 | 8.45 | 4.70 |
| 6 | 7 | -10.3 | 0 | 0 |
| 7 | 6.9 | -10.4 | -8.53 | -4.86 |
| 8 | 6.8 | -10.5 | -17.14 | -9.89 |
| 9 | 6.7 | -10.6 | -25.83 | -15.09 |
| 10 | 6.6 | -10.7 | -34.60 | -20.45 |

In the first experimental data, the values of inverse kinematics were calculated for the input of the specified X and Y coordinates. This value became the main reference value of the movement of the servo degree to point 0 or standing position of the robot.

### 2.7.2. DATA RESULTS OF INVERSE KINEMATICS

After the robot downloads the inverse kinematics program, the data can be applied to inverse kinematics in several conditions. The next step of this research is the implementation data of inverse kinematics.

TABLE 3.
Inverse kinematics data when the robot is stationary.

| Trial | X input | Y input | X output | Y output |
|---|---|---|---|---|
| 1 | 7,5 | -9,8 | 7,3 | -9,5 |
| 2 | 7,4 | -9,9 | 7,3 | -9,6 |
| 3 | 7,3 | -10 | 7,2 | -9,8 |
| 4 | 7,2 | -10,1 | 7,2 | -10 |
| 5 | 7,1 | -10,2 | 7,1 | -10 |
| 6 | 7 | -10,3 | 7 | -10 |
| 7 | 6,9 | -10,4 | 7 | -10,1 |
| 8 | 6,8 | -10,5 | 6,9 | -10,2 |
| 9 | 6,7 | -10,6 | 6,8 | -10,5 |
| 10 | 6,6 | -10,7 | 6,7 | -10,5 |

When the robot was in stable condition (stationary), there was difference between the X and Y input coordinates with X and Y output. This occured because of the load of the robot and the lack of torque on the servo.

TABLE 4.
Inverse kinematics data when the robot is walking.

| Trial | X input | Y input | X output | Y output |
|---|---|---|---|---|
| 1 | 7 | -10.3 | 6.8 | -9.7 |
| 2 | 6.9 | -10.4 | 6.9 | -9.9 |
| 3 | 6.8 | -10.5 | 7 | -10.1 |
| 4 | 6.7 | -10.6 | 7 | -10.2 |
| 5 | 6.6 | -10.7 | 7 | -10.5 |

When the robot was running. there was a greater difference between the X and Y input with X and Y output. This happened because when the robot ran. the robot used the gait tripod logic of the 3 feet with position greater than or smaller than the other. With a gait tripod. robotic servo torque would be more charged since each step used only 3 feet.

### 2.7.3. TRAVEL TIME DATA

Robots runs in a scalable arena. Arena was made of rice cardboard paper (Figures 11 and 12). The experiment was carried out 10 times to compare the travel duration of a robot when used inverse kinematics and without inverse kinematics. Table 5 below shows the robot travel time data.

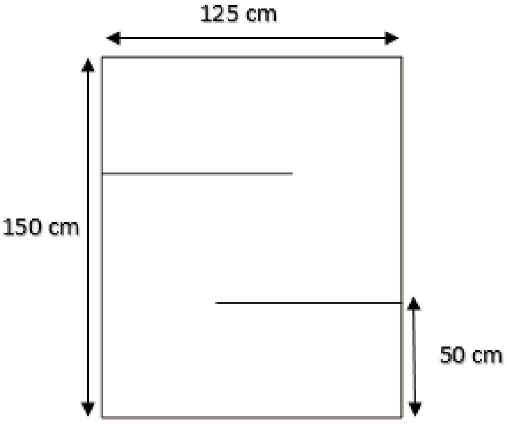


FIGURE 11. Test arena scheme

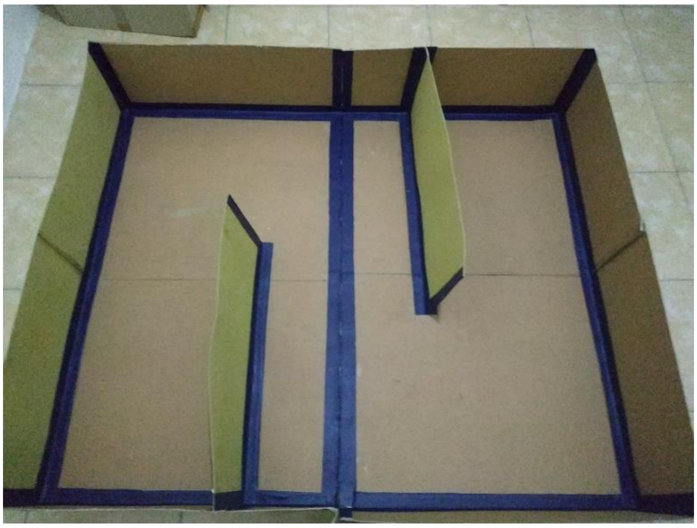


FIGURE 12. Testing Arena

TABLE 5.
Robot's travel time data.

| Using Inverse Kinematics | | Without Inverse Kinematics | |
|---|---|---|---|
| Trial | Travel time | Time | Travel time |
| 1 | 1 minute 32 seconds | 1 | 1 minute 29 seconds |
| 2 | 1 minute 47 seconds | 2 | 2 minutes 44 seconds |
| 3 | 1 minute 15 seconds | 3 | 2 minutes 6 seconds |
| 4 | 2 minutes 20 seconds | 4 | 1 minute 54 seconds |
| 5 | 1 minute 30 seconds | 5 | 2 minutes 11 seconds |

On the record of travel time. it was found that travel time using inverse kinematic was superior in several experiments. This is because. when using inverse kinematic. robots tend to run straight without any errors or noise in setting foot on land.

## 3. ANALYSIS

After conducting an experiment on a robot that detects victims of natural disasters using an ultrasonic. TPA81 proximity sensor and a push button switch as its control input. it can be concluded that this robot can function corretly to its purpose. With the help of push button switches. robots can run more precisely and neatly. Using inverse kinematics is a good choice. it is because of it will adjust the end-effector position of the robot to the floor surface. However. the calculation of inverse kinematics with the original application of the robot will look different. When the robot is stable. there is a difference between the X and Y coordinates input with X and Y output. This occurs because of the load of the robot and the lack of torque on the servo. Whereas when the robot is running. there is a greater difference between the X and Y input with the X and Y output. This happened because when the robot ran. the robot used the gait tripod logic of the 3 feet with position greater than or smaller than the other. With a gait tripod. robotic servo torque is more charged because each step uses only 3 feet.

When reading the input switch. a looping program will be run to continuously lower the position of the foot to touch the floor. Travel time with the floor depends on the program code written. This is listed in the line "spd (10)". where it is stated that the customer will do the sub program by filling in the value 10. The sub program spd is a command to run the servo with the travel time value that we input. The smaller the value on the sub program. the faster the robot's feet will touch the floor. When using a push button switch on slippery surfaces such as ceramics. robots tend to face errors. If there is a lack of friction between the switch and the floor. then the legs will look like they are trying to touch the switch and the floor. Therefore. rice paperboard is used as an arena because it has a non-slippery surface.

With this method. the robot runs more effectively. This is caused by a method of motion that will only continue if each sequence of step patterns has been fulfilled. The robot will not force its movement if a foot has not touched the floor. This is what underlies the neat navigation movement on the robot.

## 4. CONCLUSION

After conducting the experiments. some conclusion that can be adopted are listed as follows:

1. To move a robot's feet on an uneven surface. a foot touch sensor is needed against the floor so that the robot can find out the condition of the feet touching the floor or not.
2. The use of a push button switch on a hexapod robot as a foot touch sensor can improve the running of a robot which is not straight or neat during the navigation process.

3. There are differences in input and output when calculating and applying inverse kinematics. this is caused of a lack of servo torque.
4. Surface conditions are very influential on the push button switch response. because with a lack of friction force. the push button switch will have difficulty in activating the contact.

## REFERENCES

[1] S. Rathnaprabha. S. Nivetha. S. Pradeepa. and S. Sindhu. "Designing of hexapod robot." *International Journal of Advanced Research in Electrical. Electronics and Instrumentation Engineering*. vol. 5. (2) pp. 702-706. 2016.

[2] M. Zak. J. Rozman. "Design construction and control of hexapod walking robot". *2015 IEEE 13th International Scientific Conference on Informatics*. pp. 302-307. Nov 2015.

[3] T. Karakurt. A. Durdu. and N. Yilmaz. " Design of six legged spider robot and evolving walking algorithms." *International Journal of Machine Learning and Computing*. vol. 5. (2) pp. 96-100. April 2015.