DEVELOPMENT OF AN OBSTACLE AVOIDING ROBOT

Mohammad Nasucha

Abstract— Development of an obstacle avoiding robot model is required as a fundamental step towards a bigger goal, for example development of an autonomous vehicle. An obstacle avoiding robot uses a proximity sensor module, besides other parts. In this case, this robot uses a proximity sensor developed by ourselves. The robot is controlled by a program that is embedded into a microcontroller. The logics produced by the microcontroller are further processed by an interface module, in this case, also developed by ourselves. The interface module translates microcontroller's logics into voltage and current that can practically drive the two motors. This article provides a report on the project activity, consisting of summary of the design, summary of the development process and report on the running test of the robot. Following the test and program fine-tuning, it has been proven that the robot model operated well just as programmed.

Index Terms— **obstacle avoiding, obstacle avoidance, proximity sensor, mobile robot, interface module for robot**

I. INTRODUCTION

Small robots have been used by hobbyists as toys, also by practitioners and scientists as proof-of-concept devices. For example, a small obstacle avoiding robot can be a model that proves the related obstacle avoiding procedure. While the robot itself can be a significant part of a bigger goal, for example development of an autonomous vehicle.

Many obstacle avoiding robots have been developed by different parties, one of which has been reported in [1] as a low cost obstacle avoidance robot.

As a step towards development of an autonomous vehicle, a small obstacle avoiding robot was required to be developed in our project. The whole activity of our project included design, development and running test of the robot. The design process consisted of designing overall robot construction, preparing the rotating parts, preparing microcontroller board, preparing the interface module, preparing the proximity sensor module, designing motion procedure, designing motor rotation procedure and designing the procedure of the whole robot operations. Chapter 2 reports this design activity in summary.

Chapter 3 reports the summary of the development process of the robot that comprises attaching the rotating parts, attaching the microcontroller board, attaching the interface module, attaching the proximity sensor module, writing codes for motion procedure, writing codes for motor rotation procedure, writing codes for the whole robot operation procedure.

Chapter 4 reports the result of the robot running test and the conclusion.

II. DESIGN

This chapter reports the design activity without addressing the details. The details of the interface module design and the proximity sensor module design are reported in different articles.

A. Designing Overall Robot Construction

The overall robot construction has been designed in such a way that the robot is small enough and light-weighted. A small robot, by physical advantage due to lower sensor position, is more responsive to small obstacles. It will be able to run on the floor of a room, approaching obstacles of big and small sizes before it decides to avoid them. While, the light weight gives an advantage of battery energy saving. This robot shape design is shown by Figure 1.

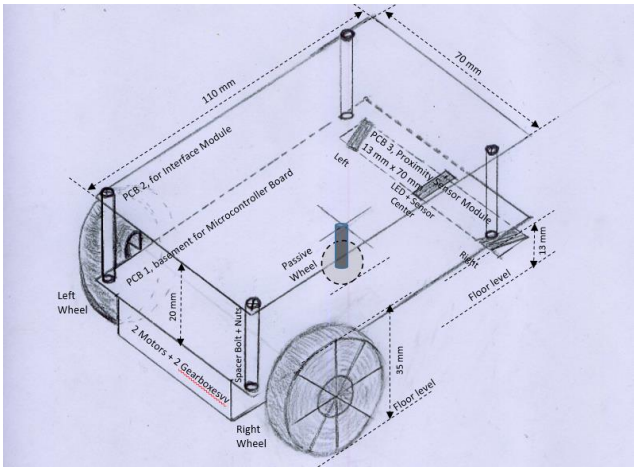


Fig. 1 Overall Robot Construction Design. The Printed Circuit Board (PCB) for the Proximity Sensor Module has size of 13mm x 70mm, is located underneath the PCB 1 so that the LED light beams are low enough and able to target many types of obstacle.

B. Preparing the Rotating Parts

The robot has been designed to be using 2 motors each with its gearbox, 2 wheels -one wheel for each motor gearbox- and 1 passive wheel. The two motorized wheel are placed in the rear while the passive wheel is placed in the front. The passive wheel is basically a ball-shaped bearing than can be directed to any direction by the force produced by the two motorized wheels. In this project the motors are twins, rated at 6vdc.

C. Preparing Microcontroller Board

In this activity the Atmel ATmega328P has been used as the microcontroller (uC). This is a 8-bit-microcontroller that has 32 kilo bytes flash memory and able to process

instructions at speed of 20 MIPS at the fastest. In particular for this occasion, a ready-to-use Arduino Uno R3 board has been used for simplicity and development time saving reasons. According to [3] pin layout of this microchip is shown by Figure 2.

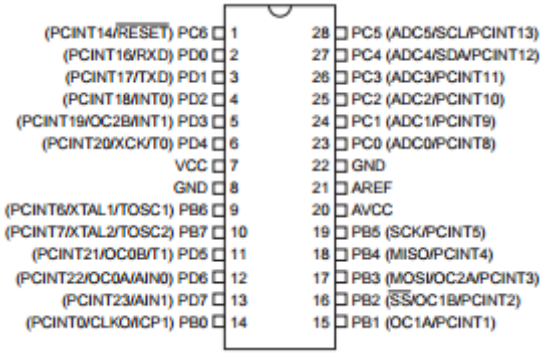


Fig. 2 ATmega328P Pin Layout according to [2]

According to [2] too, this microcontroller offers some features. In circuit development aspect, the chip provides simplicity as it requires very few external components. Fundamentally the chip needs only a single crystal for its oscillator. From operational point of view, the chip provides practicality as it can operate at a wide operating voltage range, that is, 1.8v – 5.5v. From energy saving point of view, this chip is an energy saver. It consumes 0.3 mA at active mode at the lowest, consumes 0.8 uA at power-safe-mode and 0.1 uA at power down mode. From the usage range view point, it offers high usefulness because it has various input and output pins. The chip has 14 pins that can be set as digital input or output (pin 4, 5, 6, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19), 5 of which can be set as analog (PWM) output too and has 6 default analog inputs (pin 23, 24, 25, 26, 27, 28). The analog input pins are able to measure analog voltage in 1024 gradations (ranging from 0 to 1023) while the PWM output pins are able to produces analog voltage in 255 gradations (ranging from 0 to 255).

D. Preparing the Interface Module

As mentioned in [3], the interface module in this context is a part of the robot system that converts logics 1 and 0 of the microcontroller's pins to applicable and suitable voltage and current to the motors. In this case, the two motors are identical, of 6vdc.

The system uses two lithium-ion batteries of the same voltage, that is, 4.2v. One battery is to supply the microcontroller and another one is to supply the interface module and the motors. As also explained in [4], that mobile robots –in some cases– run too fast and their batteries last fast, being the reason for the project to apply an alternative way of supplying the motors. According to a test on the motor, it runs fast even when it is supplied by a voltage of 4.2v (less than rated voltage). Furthermore, according to another test, the motors still run fast enough when they are supplied by a voltage of 2.1v but with much less current consumption. Based on these two tests and aiming a significant energy saving, the interface module has been designed to set the two motors in a serial connection. Assuming that the two motors are twins and having constant resistance R, setting the two in a serial connection results in a power saving factor of 4 compared to setting the two in a parallel connection.

Direction of each motor is determined by a relay's state, that's is controlled by the microcontroller. Thus, two relays are needed for the two motors. The logic state off the microcontroller flows to the relay's coil and the relay's magnetic switches determine the direction of battery's current flowing to the motor. This is the mechanism of directing the robot forward and backward.

The root mean square (rms) voltage itself –from the battery to supply the motor– is actually controlled by the microcontroller through a pulse width modulation (PWM) output that, in this case, is fed to the base of a power transistor. The transistor's collector pin then supplies the rms voltage to both motors (that are connected each other in a series). The value of the PWM itself is controlled by the program, which, responds to the value of a certain voltage that is produced by the middle pin of a trimmer potentiometer (trimpot).

A toggle switch is used on this interface board as a system power on/off switch.

E. Preparing the Proximity Sensor Module

As explained in [4] proximity sensing means detecting the presence or absence of an object. It is also mentioned that the measurement can be done by different type of sensors including hall-effect sensors, inductive sensors, ultrasonic sensors and contact-type sensors. In practice ultrasonic and infra-red sensors are often used for obstacle avoiding robots.

The robot in this project used the proximity sensor module that has been developed earlier and reported in [5]. The module employs 3 sets of sensor, each consists of a blue LED and a visible light photo diode. Blue LEDs are chosen due to their highest energy efficiency compared to LEDs of other colors. It is also explained that each photodiode will receive blue light waves off its paired-LED that are reflected by an obstacle residing in front of it, within a certain range of distance. This proximity sensor module has ability to detect not only an obstacle residing in the front but also those residing at right front and left front sides. It has three sensor sets that are located and directed at three different situations. The first one is located at center and directed to the front. The second one is located at the right and directed to the right front. The third one is located at the left and directed to the left front. This arrangement of sensor set positions and their beam directions is reported in [5].

F. Designing Motion Procedure

The motion procedure of the robot has been designed as follows:

Stop:	Both motor do not rotate.
Move forward:	Both motor rotate forward.
Turn right:	The left motor rotates forward.
	The right motor rotates backward.
Turn left:	The right motor rotates forward.

The left motor rotate backward.

Move backward: Both motors rotate backward.

G. Designing Motor Rotation Procedure

In accordance with the MOTION PROCEDURE and the operation principles of the already designed INTERFACE MODULE, another procedure is required in order to correctly and practically drive the motors. This procedure has been designed as follows:

MR to rotate forward: CR to receive logic 0 (0v)
MR to rotate backward: CR to receive logic 1 (4.2v)
ML to rotate forward: CL to receive logic 0 (0v)
ML to rotate backward: CL to receive logic 1 (4.2v)

Where:
MR = Right Side Motor
ML = Left Side Motor
CR = Coil of Right Side Relay
CL = Coil of Left Side Relay

H. Designing the Robot Operation Procedure

The designed procedure of the whole robot operations is shown by Figure 3.

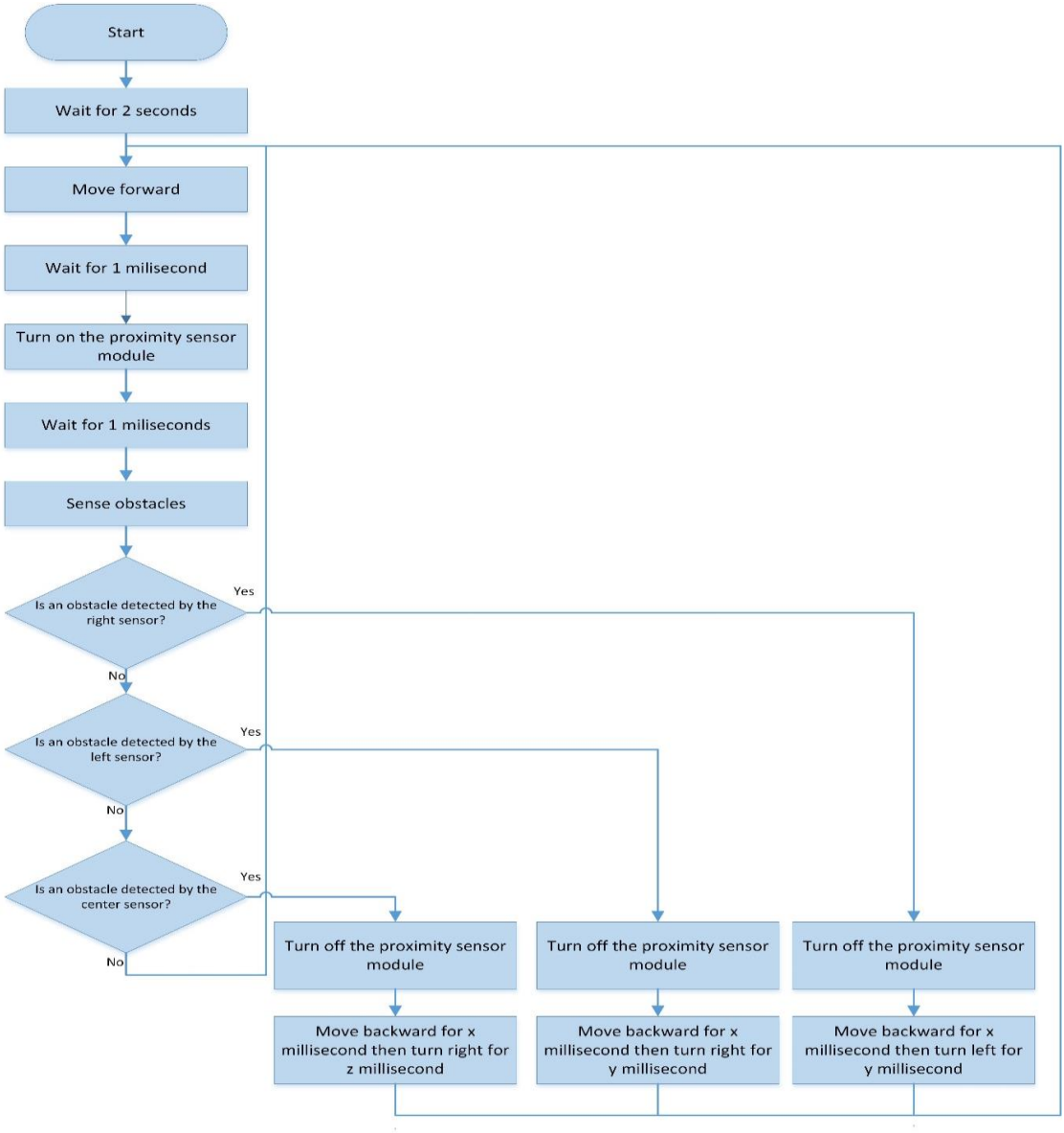


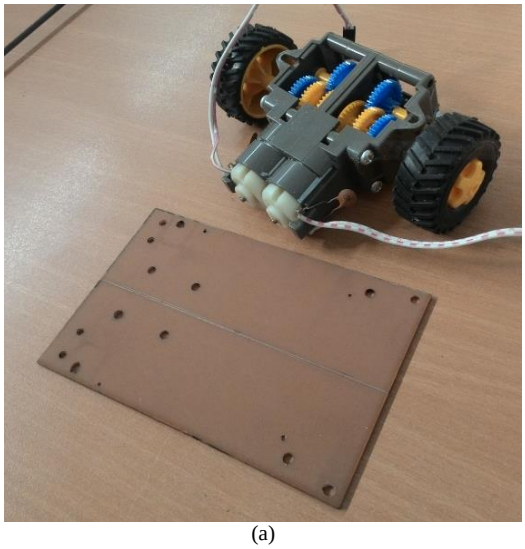
Fig.3 Procedure of robot operation shown in a flow chart. The operation starts when battery is on and on/off switch is pushed. The 2 second delay give a chance for operator to put the robot on the floor. The value of x, y and z are changeable and should be set in a way so that the robot turning motion is suitable for the room or path size. To be noticed that z > y. The obstacle avoiding pattern works as follows: (1) If no obstacle is detected, the robot will keep moving forward, (2) If an obstacle in right front is detected, the robot will move backward for a while then turn left for a while then move forward, (3) If an obstacle in left front is detected, the robot will move backward for a while then turn right for a while then move forward, (4) If an obstacle in the front is detected, the robot will move backward for a while then turn right for a bit longer then move forward.

III DEVELOPMENT

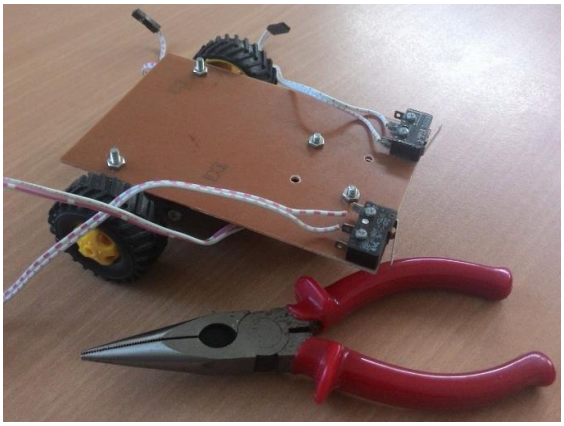
The whole development process consists of attaching the rotating parts, attaching the microcontroller board, attaching the interface module, attaching the proximity sensor module, writing codes for motion procedure and motor rotation procedure and at last writing codes for the whole robot operation procedure.

A. Attaching the Rotating Parts

The set of rotating parts consists of 2 motors, two gear boxes, two rear wheels and a passive wheel. The process of attaching the rotating parts is shown by Figure 4.



(a)



(b)

Fig. 4 Attaching the rotating parts: (a) Two motors, two gearboxes, two rear wheels and a Printed Circuit Board (PCB) of 110mm x 70mm have been prepared. (b) All rotating parts have been attached to the PCB, in this case underneath it. The passive wheel has been attached underneath the PCB too, located at front; it is not shown at this photograph. The two black cuboids are push-on switches that will be used as collision detectors as backup to the proximity sensors.

B. Attaching the Microcontroller Board

The microcontroller board has been attached to the first PCB and it is shown by Figure 5.

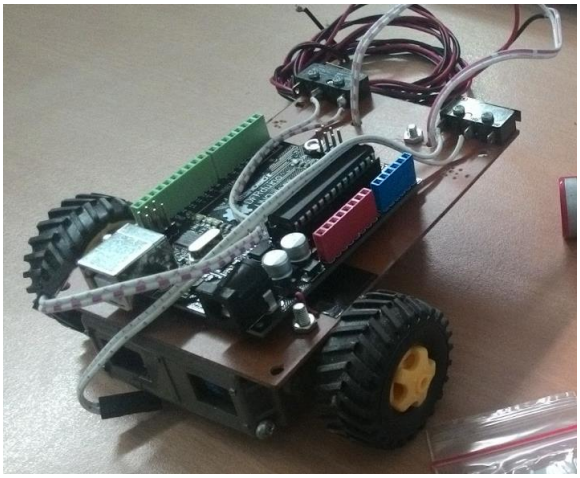


Fig. 5 Attaching the microcontroller board onto the first PCB. The PCB size has been designed to be suitable to the motors, gearboxes and the microcontroller board.

C. Attaching the Interface Module

The interface module has been attached to the robot structure as shown by Figure 6.

(a)

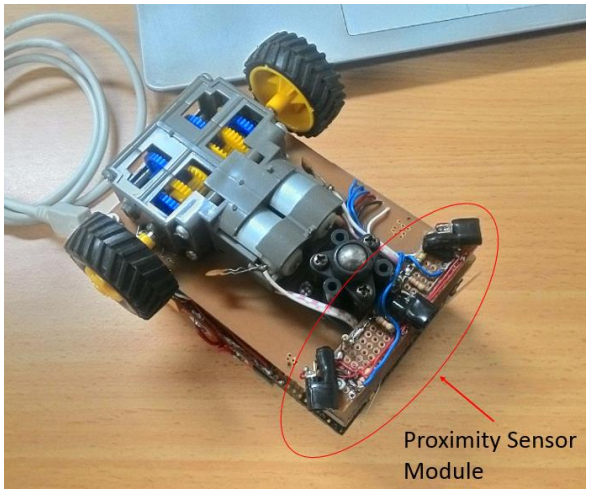
(b)

Fig. 6 (a) Preparing the interface module (b) The interface module has been attached to the robot.

D. Attaching the Proximity Sensor Module

The proximity sensor module has been attached to the PCB1 as shown Figure 7.

(a)



(b)

Fig. 7 (a) Preparing the proximity sensor module (b) The module has been attached to the PCB 1 as shown, where the robot was turned upside down.

E. Writing Codes for Motion Procedure and Motor Rotation Procedure

Motion procedure, as explained in section 2.5, together with motor rotation procedure, as explained in section 2.6 have been realized into program using C++ language with Arduino compiler. The codes are reported in [3]. As explained in [3] pin A0 of the microcontroller is used to read the analog voltage produced at the center pin of the "speed trimer potentiometer (trimpot)". This value then determines the output of pin 11, the pin that supplies analog voltage –in the form of PWM – to both motors, through a current amplifier of the interface module.

Pin 12 supplies HIGH or LOW voltage to the coil of a relay, that determines the rotation direction of the right motor (MR). Pin 11 supplies HIGH or LOW voltage to the coil of another relay, that determines the rotation direction of the left motor (ML).

F. Writing Codes for the Whole Robot Operation Procedure

The whole robot operation procedure, as explained in section 2 part H has been realized into program using C++ language with Arduino compiler. The codes are shown in Appendix-2.

Please notice that the robot has been equipped with additional sensors, that is, two push-on switches that are used as collision detectors as backup to the proximity sensors. The right switch is called SwR and the left switch is called SwL. It has been designed that the proximity sensing function and the collision detection function works together where the final result is obtained by applying OR logic. It means that an obstacle is considered existent if the proximity sensor detects it or the collision detector detects it.

G. Uploading the Program

The whole program has been uploaded from the computer to the microcontroller. A USB data cable has been required to deliver data as well as 5V supply from the computer to the microcontroller as depicted in Figure 8.

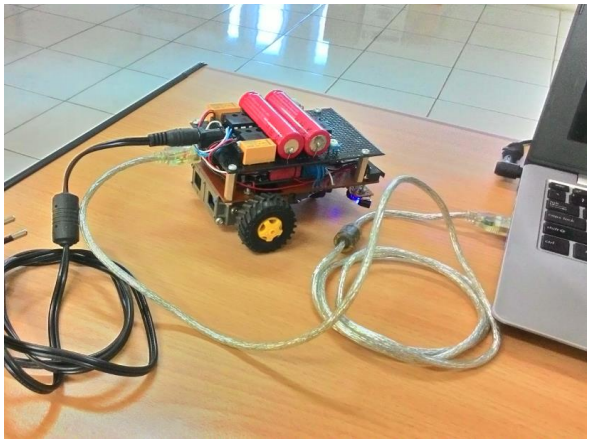


Fig. 8. Uploading the program to the microcontroller. During this process the interface module should be switched off.

III. Test Result

Robot operation test has been carried out in two ways: (i) pre-set test, where the robot was set in such away so that the robot motion direction at the start created certain angles in regards to the obstacle's surface, (ii) free style test, where the robot was released in a typical office room and ran freely.

A. Pre-Set Test

A pre-set test has been carried out where the robot, before the start, was placed in such position so that the coming angle (angle between robot motion direction and the obstacle surface) was 15°, 30°, 45°, 60°, 75°, 90°, 105°, 120°, 135°, 150° and 165°. Here in Figure 4.1-1 the setting for 45° and 90° are shown.

In the test it was found that for coming angle of 45°, 90° and 135° the robot responded to obstacle existence at its best. For those positions, the robot delivered responses at distance of 6.1cm, 6.2 cm and 6.0 cm, longer than distance of any other coming angles. The reason should be, that at coming angle of 45°, the left LED's light beam hit the wall surface perpendicularly so that the paired sensor received maximum reflected light. The same situation occurred for coming angles of 90° and 135° where the center and right LED's light beam, respectively, hit the wall surface perpendicularly and its paired sensor received maximum reflected light.

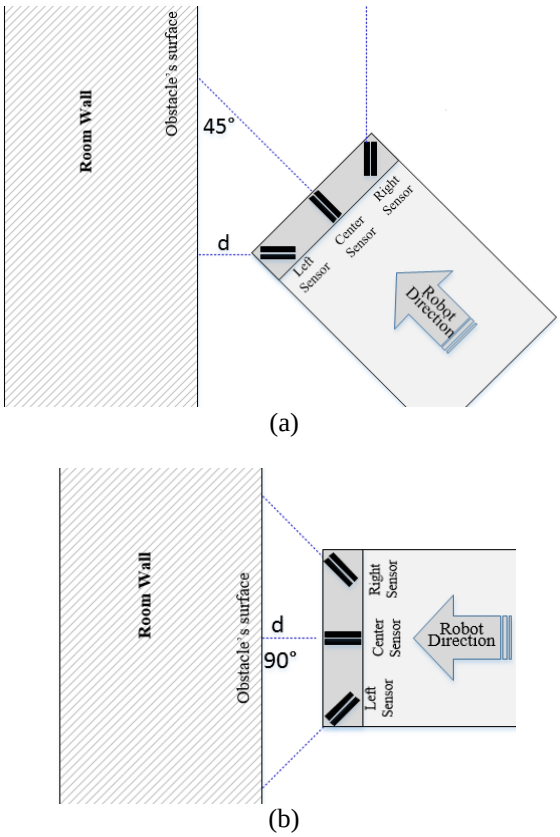


Fig. 9. Different coming angles at test were set, two of which, are here shown: (a) 45°, (b) 90°

B. Free Style Test

In this test the robot was released in a typical office room. The size of the room was 9m x 12m, it had 6 work desks, 12 chairs, two meeting cubicles with doors were let open. Other goods were also residing in the room such as file cabinets and some carton boxes. The test was carried out for 3 minutes at a daytime where some natural lighting from the sun came into the room combining with lights from room's fluorescent lamps. A photograph of the robot free style test is shown by Figure 10.

After the test and program fine-tuning, the result was in general, as expected where the robot was able to detect obstacles and avoid them in the way as programmed: when it faced no obstacle it ran straight forward, when it faced an obstacle in the right front it turned backward then turned left, when it faced an obstacle in the left front it turned backward then turned right.

Nevertheless there was a burden where for a few times the robot was late in detecting the obstacle so that it hit the obstacle first –due to momentum– before it moved backwards and turned right or left. This burden should be overcome in the next project. It is predicted that this burden can be solved by decreasing the robot speed and / or reworking on the proximity sensor module.

Fig. 10 Free style test in a typical office room. Three blue light beams of the sensor module are seen.

IV. Conclusion

The designed small avoiding obstacle robot has been developed successfully in this project and operated nearly as expected. For this robot design, the robot responded to an obstacle at its best at coming angle of 45°, 90° and 135°. The reason for this, is that at this coming angles one of the LED light beam hit the obstacle's surface perpendicularly. At other coming angles the robot responded to obstacle existence with less sensitivity.

Acknowledgements

I wish to thank Hendi Hermawan as the head of the department for equipping the department with a Robotics Lab and Prio Handoko for having managed the provision of robot components earlier, and to both for giving constructive feedbacks. I wish to thank Universitas Pembangunan Jaya for providing other supports to the research.

References

[1] V. Hanumante et al., "Low Cost Obstacle Avoidance Robot" in International Journal of Soft Computing and Engineering, Vol. 3, Issue-4, 2013

[2] ATMEGA328P Datasheet, ATMEL Corporation, San Jose, CA, 2008

[3] M. Nasucha, "Design and Development of an Interface Module for Obstacle Avoiding Robots", Microcontroller Lab, Univ. Pembangun Jaya, Tangerang Selatan, Indonesia, 2015

[4] M. Jouaneh, "Proximity Measurement" in Fundamentals of Mechatronics, SI Edition, United States: Nelson Engineering, ch.7, sec. 7.4, pp. 221-229, 2012

[5] M. Nasucha, "Design and Development of a Visible Light Sensor Module for Obstacle Avoiding Robots", Microcontroller Lab, Univ. Pembangunan Jaya, Tangerang Selatan, Indonesia, 2015

Mohammad Nasucha was born in Muntilan in 1971. This author finished his bachelor study (with honor) at the department of Electrical Engineering at Universitas Gadjah Mada in Yogyakarta majoring in Electronics and Telecommunications and accomplished his masters study at the department of Communications Technology of the Universitaet Kassel in Germany.

However, the author's great passion in analog and digital circuit design has been a drive to run a private lab by his own.

Having been working for years in his private lab, he produced several circuit designs, some of which, are ready to fabricate, including ones in embedded system area.

Mr. Nasucha has been, since 2011, a lecturer and researcher at Universitas Pembangunan Jaya in Tangerang Selatan, Indonesia. Funded by this institution, in 2012 Mr. Nasucha successfully led a team developing an electric car named Rinus-1. Mr. Nasucha is also the coordinator for the Robotics Lab at the institution

APPENDIX

Codes for the Whole Robot Operations

```
//Hardware Setup, is executed once only.
void setup()
{
  pinMode(2, INPUT);   // for right switch SwR
  pinMode(3, INPUT);   // for left swicth SwL
  //A0, A1, A2, A3, A4 and A5 are analog inputs by  default.
  //A0 is to measure the voltage of speed trimpot, matched with pin 1.
  //A1, A2, A3 is to measure the voltage output off the sensor SR, SC and SL.
  pinMode(5, OUTPUT);  // is to supply voltage to the three blue LEDs for those sensors.
  pinMode(11, OUTPUT); // produces PWM as supply voltage to two serial motors . It runs the motors and determine their speed.
  pinMode(12, OUTPUT); // to reverse the rotation direction the right motor MR (move backward).
  pinMode(13, OUTPUT); // to reverse the rotation direction the left motor ML (move backward).
  analogWrite(11, 0);
  delay(2000);            // waiting for 2 seconds before running the next instruction.
}
  //Declare global variables and constants, to be executed once only.
  int VRvoltage;
  float compensation  =  1.1;
  // As the battery voltage < 5V, the voltage reading value will not return 1023, so a compensation is needed to put it back to 1023, or more.
  int speed;
  int SwR, SwL;
  int SR1, SR2, SR, SC1, SC2, SC, SL1, SL2, SL;
  int obstacle_at_right;
  int obstacle_at_front;
  int obstacle_at_left;
  int delta        =  3 ;
 //the difference between the sensor value after and before turning on the blue LEDs; it relates to the distance between sensor and the obstacle.
  int get_ready    =  1000;  //Duration of a delay (preparation time) before the robot departs
  int pause            =  1;       //Duration of a pause before turning motor direction to eliminate momentum
  int backward   =  600; //Duration of backward motion
  int right_more =  600; //Duration of turning right longer
  int right_sligh =  300;  //Duration of turning right slightly
  int left_more   =  600; //Duration of turning left longer
  int left_slght    =  300; //Duration of turning left slightly

void loop()
{
 VRvoltage = analogRead(A0);
 speed = (VRvoltage*compensation)/4;
 if (speed > 255){speed = 255;}
 if (speed == 255 || speed < 255) {speed = speed;}

SR1  =  analogRead(A1);  SC1  =  analogRead(A2);  SL1  =  analogRead(A3);
 delay(1);
 digitalWrite(5, HIGH);  // Turning on the3 blue LEDs
 delay(2);
 SR2  =  analogRead(A1);  SC2  =  analogRead(A2);  SL2  =  analogRead(A3);
 delay(1);
 digitalWrite(5, LOW);   // Turning off the LEDs

 if (SR2 - SR1 < delta) {SR = LOW;}
 if (SR2 - SR1 == delta || SR2 - SR1 > delta) {SR = HIGH;}
 if (SC2 - SC1 < delta) {SC = LOW;}
 if (SC2 - SC1 == delta || SC2 - SC1 > delta) {SC = HIGH;}
 if (SL2 - SL1 < delta) {SL = LOW;}
 if (SL2 - SL1 == delta || SL2 - SL1 > delta) {SL = HIGH;}

 SwR = digitalRead(2);
 SwL = digitalRead(3);

obstacle_at_right  = SR || SwR;
obstacle_at_front = SC;
obstacle_at_left   = SL || SwL;

 //CLEAR (NO OBSTACLE)
 if (obstacle_at_right == LOW && obstacle_at_front == LOW && obstacle_at_left == LOW)
{
  //MOVE FORWARD
  digitalWrite(12, LOW);
  digitalWrite(13, LOW);
  analogWrite(11, speed);
}

//OBSTACLE IN FRONT
if (obstacle_at_front == HIGH)
{
   //PAUSE
  analogWrite(11, 0);
  delay(pause);

  //MOVE BACKWARD
  digitalWrite(12, HIGH);
  digitalWrite(13, HIGH);
  analogWrite(11, speed);
  delay(backward);

  //PAUSE
  analogWrite(11, 0);
  delay(pause);

  //TURNING RIGHT LONGER
  digitalWrite(12, HIGH);
  digitalWrite(13, LOW);
  analogWrite(11, speed);
  delay(right_more);

  //MOVE BACKWARD
  digitalWrite(12, LOW);
  digitalWrite(13, LOW);
  analogWrite(11, speed);
}

 //OBSTACLE AT RIGHT FRONT
 if (obstacle_at_right == HIGH)
 {
   //PAUSE
  analogWrite(11, 0);
  delay(pause);

  //MOVE BACKWARD
  digitalWrite(12, HIGH);
  digitalWrite(13, HIGH);
  analogWrite(11, speed);
  delay(backward);

  //PAUSE
  analogWrite(11, 0);
  delay(pause);

  //TURN LEFT SLIGHTLY
  digitalWrite(12, LOW);
  digitalWrite(13, HIGH);
  analogWrite(11, speed);
  delay(left_slght);

  //MOVE FORWARD
  digitalWrite(12, LOW);
  digitalWrite(13, LOW);
  analogWrite(11, speed);
 }

 //OBSTACLE AT LEFT FRONT
 if (obstacle_at_left == HIGH)
 {
```

```
 //PAUSE
analogWrite(11, 0);
delay(pause);

//MOVE BACKWARD
digitalWrite(12, HIGH);
digitalWrite(13, HIGH);
analogWrite(11, speed);
delay(backward);

//PAUSE
analogWrite(11, 0);
delay(pause);

//TURN RIGHT SLIGHTLY
digitalWrite(12, HIGH);
digitalWrite(13, LOW);
analogWrite(11, speed);
delay(right_slight);

//MOVE FORWARD
digitalWrite(12, LOW);
digitalWrite(13, LOW);
analogWrite(11, speed);
 }
 delay(100);
}
```