Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 12, December-2016

A Simple Line Tracing Mobile Robot

Myunggon Yoon
Department of Precision Mechanical Engineering
Gangneung-Wonju National University,
South Korea

Jung-Ho Moon
Department of Electronic Engineering
Gangneung-Wonju National University,
South Korea

Abstract—**In this paper we introduce a simple line tracing robot composed of minimal electronic components including a MCU (micro controller unit) and a half-H bridge. This robot has been used as a design project for an advanced mechatronics course offered for students in mechanical engineering. We have found that our simple robot can provide key ideas and fundamental concepts of general mechatronic system design.**

Keywords— Liner Tracer, Automatic Guidance Vehicle, Micro Control Unit

I. INTRODUCTION

Being a multidisciplinary subject, *mechatronics* is hard not only to learn but to teach. A main difficulty is that basically mechatronics covers too wide fields including mechanical and electrical engineering and computer science, to name a few.

As a result, from the viewpoint of a lecturer, it is a challenging task to design and prepare proper course topics and lab experiments for students majoring a particular field. It seems to be inevitable that some topics should be sacrificed for others. In fact, most students in electrical engineering find it hard to understand the concepts of torque, momentum and Newton’s law. Similarly, only a few students in mechanical engineering truly know the Ohms’ law even though they have already learnt it during early course works such as physics and elementary circuit theory.

In our mechanical department, we provide students two lectures on mechatronics; *Introduction to mechatronics* and *advanced mechatronics*. The first lecture *Introduction to mechatronics* deals mainly with elementary analog and basic digital circuit theory and the second *advanced mechatronics* course focuses on applications of micro controllers and related topics including analog and digital filtering. Most students in the second course have limited experiences on a computer programming and practical knowledge on digital electronics. Therefore they are not ready for designing and implementing a complex mechatronics system for themselves. Nevertheless mechatronics is a practical topic rather than a theoretical one and students should have real experiences during the course to enhance their practical knowledge.

In this situation, as an extension of our analog line tracer in [1], we designed a simple line tracing robot as a term project which most students can quickly make by themselves. For this, the line tracing robot should be minimal in the perspective of both hardware and software. In order to make the robot hardware as simple as possible, the controller board of the robot is only composed of one MCU and a half-H bridge motor driver, along with four optical sensors. For simplicity in software development, we used an Arduino© MCU and its IDE (integrated development environment).

In addition, we have provided students technical materials on the design steps and guides, which are summarized below.

II. DESIGN

A. Overview

Fig. 1 shows a photo of our line tracing robot. The sensor part includes four pairs of infrared LED and optical transistor. An Arduino Nano board was used as a main controller and a Half-H bridge driver L293D© for DC motors. Technical details of these two components can be found in [2,3].

Mechanical parts include two geared DC motors with plastic wheels, a 12mm metal ball caster under the body and a frame which is an aluminium plate with 1mm thickness. The overall size of the robot is about 13 cm x 12 cm x 6 cm, including a 9V battery.

Table 1 summarizes technical specifications of mechanical and electronic components of the robot

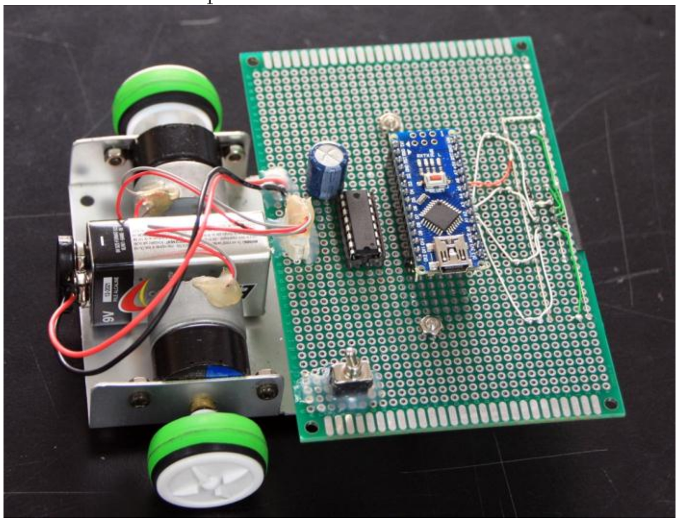
Figure 1 Photo of line tracer

TABLE I. COMPONENTS SPECIFICATIONS

Mechanical Parts	Body	Aluminum Plate 75 mm x 89 mm Thickness 1mm
	Ball Caster	12 mm metal ball outer diameter 13 mm ball diameter 11.86 mm
	Wheel	Plastic & Lubber Diameter 26mm Width 11 mm
	Geared Motor	nominal voltage 4.5 V no-load speed 580 rpm nominal torque 50 g cm mass 60 g
Electronic Parts	Infrared LED	SI5315 (x4)
	Photo Transistor	ST5811 (x4)
	Array Resistors	330(x1), 10 K(x1)
	Micro Controller Unit	Arduino Nano (x1)
	Half-H Driver	L293D (x1)
	Capacitor	330 μF 25V
	Power switch	Two ways, toggle
	PCB	Double sided 13mm x 12 mm
	Battery	9V alkaline
	PCB stand	Height 25 mm 3.0 mm female

B. Sensor Design

The optical sensor has four pairs of infrared LED (SI5315) and photo transistors (ST5811) from AUK Semiconductors© [4].

The electronic circuit of the sensor is given in Fig. 2. In order to save PCB space and to relieve soldering complexity, we have used two array resistors as shown in Fig. 3.

While changing the location of the robot as show in Fig. 4, four sensor outputs were read in the main CPU after 8 bits AD (analog-digital) conversion. The sensor data in Fig. 5 show that the characteristics of four sensors are rather different, depending on the LED/photo-TR alignment and electronic properties of properties.

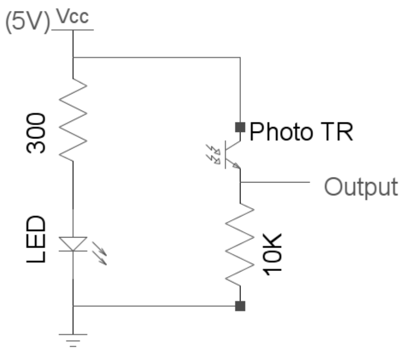

Figure 2 Sensor Circuit

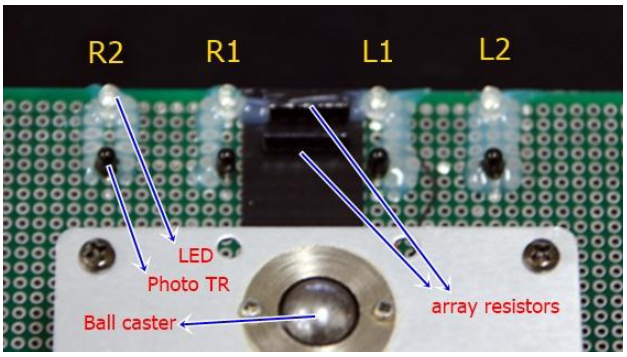

Figure 3 Sensor Configuration

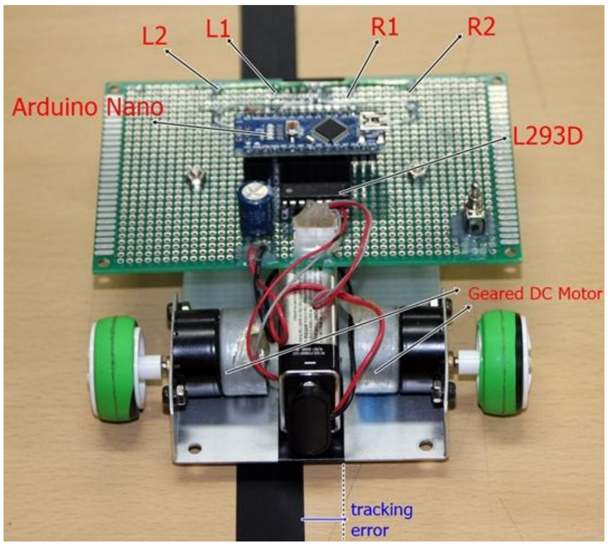

Figure 4 Sensor Experiment

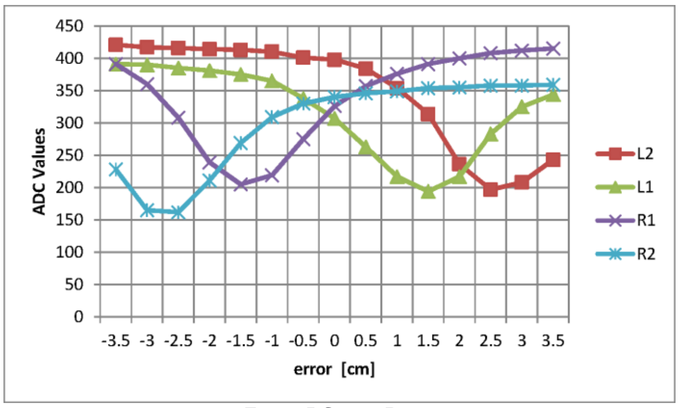

Figure 5 Sensor Data

For a compensation of the differences in the four sensors, the sensor data were normalized by the maximum value of each sensor. The normalized sensor outputs shown in Fig. 6 were used for an estimation of the tracking error of the robot (see Fig. 4 for the definition of tracking error).

For an estimation of tracking error, a linear combination of four sensor outputs L2, L1, R1, R2 given below

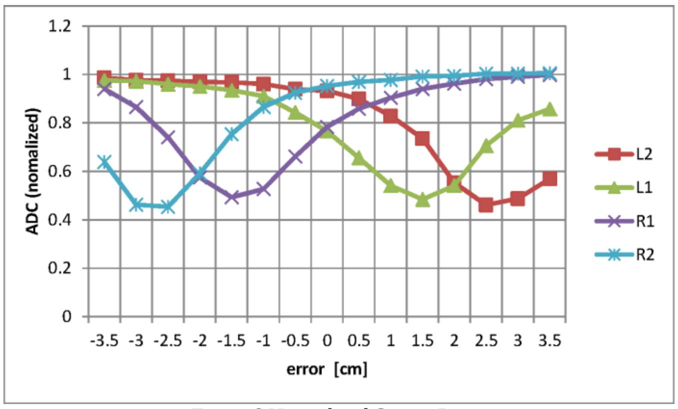

Figure 6 Normalized Sensor Data

$$x(e) = a_1 \frac{L_1}{L_1^{max}} + a_2 \frac{L_2}{L_2^{max}} + a_3 \frac{R_1}{R_1^{max}} + a_4 \frac{R_2}{R_2^{max}} \qquad (1)$$

where e denotes the error and the superscript "max" means the maximum sensor output.

As a next step, we chose the four parameters $\{a_1, a_2, a_3, a_4\}$ such a way that the function $x(e)$ is close to the error as much as possible for a wide range. Some trial and error gives the next parameters and the corresponding plot in Fig. 7.

$\frac{a_1}{L_1^{max}}$	$\frac{a_2}{L_2^{max}}$	$\frac{a_3}{R_1^{max}}$	$\frac{a_4}{R_2^{max}}$
-0.00808	-0.00374	0.00361	0.00924

The result in Fig. 7 shows that our sensors give reliable error estimation on the range $|e| \le 2$ (cm) approximately.

C. Motor Driver

Two motors of our line robot were derived by a half-H bridge driver L293D.

The connection diagram is given in Fig. 8. The logic power source (+5V) of L293D is supplied from a +5V regulator of the Arduino board and two enable pins are connected to the PWM (Pulse Width Modulation) digital port D5, D6 of the Arduino. The motor power source Vs is connected to the 9V battery.

For the simplicity of a circuit, we did not install the so-called flywheel diodes in our circuit.

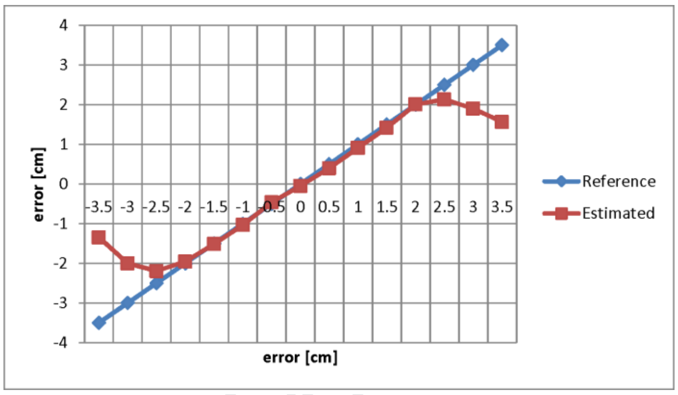

Figure 7 Error Estimation

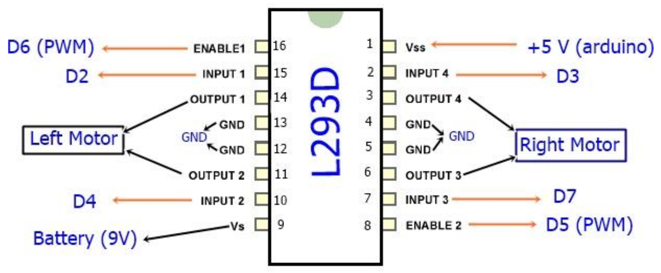

Figure 8 Motor Driver Circuit

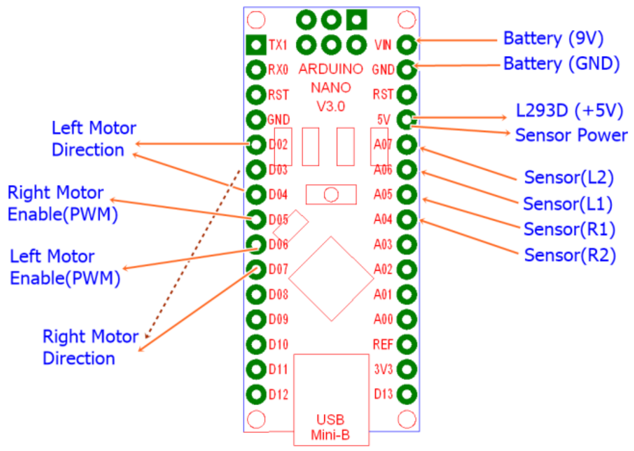

Figure 9 Controller Diagram

D. MCU

The connection diagram of the main controller, Arduino Nano, is given in Fig. 9. This board includes a +5V voltage regulator (AMS1117) and it supply 5V power to both motor driver (L293D) and the sensor part.

A PD (proportional and derivative) controller structure is chosen [5]. That is, the speed command of left/right motors is given as a sum of a nominal 8 bits PWM command plus/minus a PWM offset command;

$$\text{PWM command= Nominal PWM} \pm \text{control}$$

$$\text{control[n]} = k_p e[n] + k_D(e[n] - e[n-1]) \qquad (3)$$

where $e[n]$ and $e[n-1]$are current and previous errors in the unit of centimeter.

A complete Arduino source code for error calculation and motor control can be found in Appendix below.

E. Experiment

With the following controller parameters

Nominal PWM	k_p	k_d	Sampling Time
82	20	25	10 (ms)

a successful tracking of our robot could be observed. Snapshots of this tracking motion are given in Fig. 10.

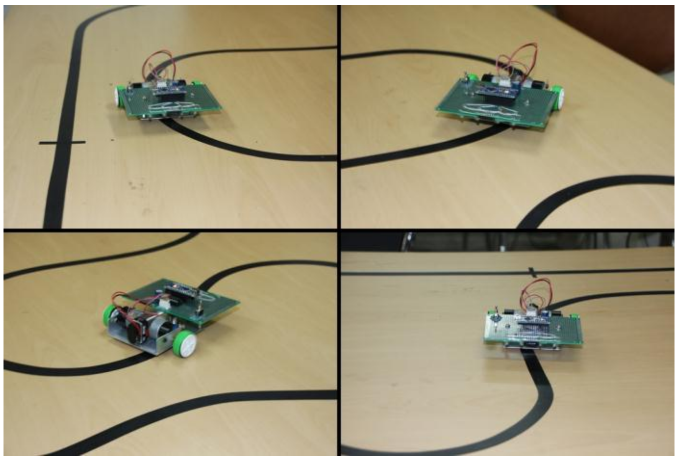

Figure 10 Tracking Experiment

III. CONCLUSION

We designed an Arduino-based simple line-tracing robot having minimal mechanical/electronic components. The simplicity of the robot allowed most students to make their own robots in a short time. After this design task, many students reported that they could understand the importance and subtlety of sensor part in a system design, which is a common fact in many mechatronic systems. In addition, thanks to the simplicity, students could save their time and focus more on developing their software skills. It is also remarkable that our simple line tracer can be used as a testbed for more advanced courses, such as the automatic control course.

REFERENCES

[1] Myunggon Yoon "A simple analog line tracer for mechatronics education", International Journal of Engineering Research & Technology, Vol 5, Issue 1, pp 205-207, January 2016
[2] "Arduino Nano" available at https://www.arduino.cc/en/Main/ArduinoBoardNano Accessed December 8, 2016.
[3] "L293D Datasheet" available http://www.ti.com/lit/ds/symlink/l293.pdf Accessed December 8, 2016
[4] "AUK semiconductors" available http://www.comsoc.com/semicon-auk.html Accessed December 8, 2016
[5] K. Ogata, Discrete-Time Control Systems. Prentice Hall, 1994.

APPENDIX

```
// sensor data, motor PWM
int L1, L2, R1, R2, Nominal_PWM=0, Lpwm, Rpwm;

// tracking error, control
float error=0, old_error=0, control=0;

// controller gains
float kp=20, kd=25;


void setup() {
    // Sensor Pins
    pinMode(A4,INPUT);pinMode(A5,INPUT);
    pinMode(A6,INPUT);pinMode(A7,INPUT);
    // Motor Pins (Left Direction)
    pinMode(2,OUTPUT);pinMode(4,OUTPUT);
    // Motor Pins (Left Direction)
    pinMode(3,OUTPUT);pinMode(7,OUTPUT);
  // Left Motor Enable (PWM)
    pinMode(6, OUTPUT);
   // Right Motor Enable (PWM)
    pinMode(5, OUTPUT);
    // Motor Initial Direction & Speed
    digitalWrite(2,LOW);digitalWrite(4,HIGH);
    digitalWrite(6,LOW);// Left motor stop
    digitalWrite(3,HIGH);digitalWrite(7,LOW);
    digitalWrite(5,LOW);// Right motor stop
    // Forward speed (PWM)
    Nominal_PWM =82; //
    Serial.begin(9600);// for error monitoring
}


void loop() {
    // error calculation
    L2= analogRead(A7);L1=analogRead(A6);
    R1= analogRead(A5);R2=analogRead(A4);
    error=-0.00808*L2-0.00374*L1
                  +0.00361*R1+0.00924*R2;
    Serial.println(error);// serial error display
    // PD controller
    contrrol=kp*error+kd*(error-olderror);
    olderror=error; // saving previous error
     // left speed control
    Lpwm= Nominal_PWM +control;
    analogWrite(5,Lpwm);
    // right speed control
    Rpwm= Nominal_PWM -control;
    analogWrite(6,Rpwm);
    // control loop delay
    delay(10);
 }
```