THE DESIGN AND CONTROL OF A MOBILE VEHICLE WITH ARDUINO MICROCONTROLLER BASED ON ALGORITHM PID

Gîlcă Gheorghe, *"Constantin Brâncuși" University from Târgu-Jiu, ROMANIA*

ABSTRACT: *This paper presents the development of a line follower wheeled mobile robot. In this project, ATmega 328P which is based on the AVR enhanced RISC architecture is chosen as the main controller to react towards the data received from infra-red line sensors to give fast, smooth, accurate and safe movement in partially structured environment. A dynamic PID control algorithm has been proposed to improve the navigation reliability of the wheeled mobile robot which uses differential drive locomotion system. The experimental results show that the dynamic PID algorithm can be performed under the system real-time requirements.*

KEY WORDS: mobile robot, sensor, tracking control, line follower

1. INTRODUCTION

There are various types of controllers used in industry, laboratory, and routine applications. Some of the commonly used controllers are on/off, PID, fuzzy, and neural. The latter two are a bit complex, and use more sophisticated concepts like artificial intelligence. The controllers can also be differentiated as "feedforward" and "feedback" controllers. The feedforward controller works by giving a result based on the anticipation of the next step, while the feedback controller works by giving an observed result that changes the processing value of the later step. In this project, the entire focus is on the feedback control system that uses the PID algorithm.

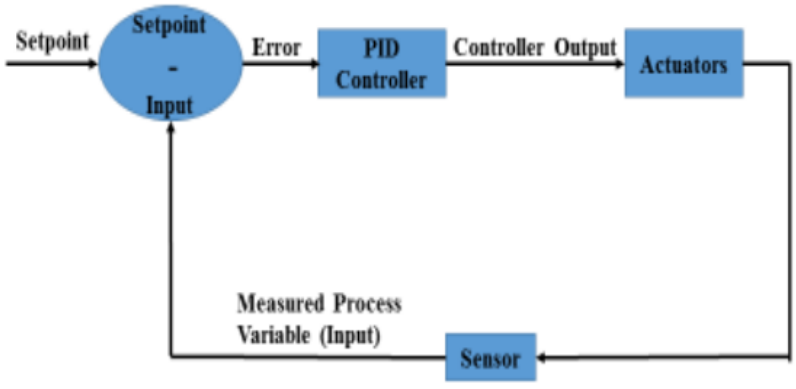

Figure 1. Block diagram with PID controller and negative feedback [1]

Simple line follower robot - It's a machine that follows a line either a black line on white surface or vice versa. High tech line follower robot - High tech line follower is also a line follower which follows either the black line on white surface or vice versa but in less time and less error by using feedback system. How high tech is different from simple? It takes less time to complete the path. It is

more reliable than simple line follower robot. The technology we used to make high tech line follower is PID [2]. A proportional integral-derivative controller (PID controller) is a control loop feedback mechanism. A PID controller calculates an "error" value as the difference between a measured process variable and a desired set point. The controller attempts to minimize the error by adjusting the process control outputs [3]. The PID controller algorithm involves three separate constant parameters. The proportional, the integral and derivative values, denoted P, I, and D. Simply put, these values can be interpreted in terms of time: P depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change.

2. THE ARCHITECTURE OF SYSTEM

For the system studied was considered the architecture of figure 2.

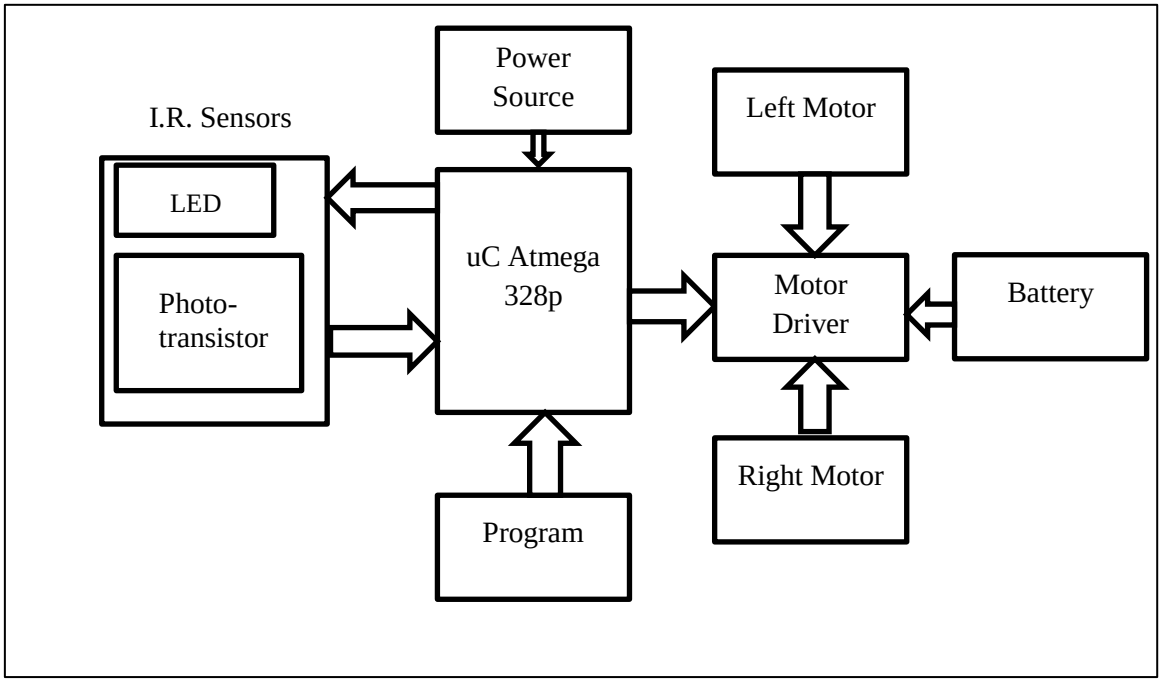

Figure 2. The block diagram of the mobile robot

Generally, the line follower robot is one of the self operating mobile machines that follow a line drawn on the floor. The path can be a visible black line on a white surface or vice versa. The basic operations of the line follower are as follows:

• Capturing the line position with optical sensors mounted at the front end of the robot. Most are using several numbers of photo-reflectors. Therefore, the line sensing process requires high resolution and high robustness.

• Steering the robot to track the line with any steering mechanism. This is just a servo operation; actually, any phase compensation will be required to stabilize tracking motion by applying digital PID filter or any other servo algorithm.

• Controlling the speed according to the lane condition. The speed is limited

during passing a curve due to the friction of the tire and the floor.
From physically building the robot platform, to setting up, programming, and hardware or software fine tuning, everything needs to be taken into account when building a differential wheeled mobile robot. A mobile robot can be regarded essentially as an ensemble of five main parts and subsystems.
• Chassis and body.
• Sensors and signal processing circuits.
• Microcontroller.
• Motor drivers
• Actuators (Motors and wheels)

2.1. The chassis and body

The chassis would be the main part of a robot's body. It is designed to carry all of the other components, transmission mechanisms, electronics and battery. It needs to be sufficiently large and provide adequate fixtures to accommodate all necessary parts, as well as sturdy enough to cope with the weight of the parts along with additional loads which can appear in dynamic conditions such as vibrations, shocks or chassis torsion and actuators torque. There are some good materials for designing robots such as plastic, aluminum and carbon-composites.

2.2. IR Sensors

It is also known as light sensor. It is used to detect black or white surfaces [4]. It consists of two main parts:
a) Led - It transmits Infra-Red light on the surface (transmitter).
b) Phototransistor - It detects the light (receiver).
When Led transmits light on the surface then, if surface is black the whole light will be absorbed by the surface and no light will get reflected back to the phototransistor and if the surface is white, light is reflected and it is detected by the phototransistor as shown in the fig. 3.

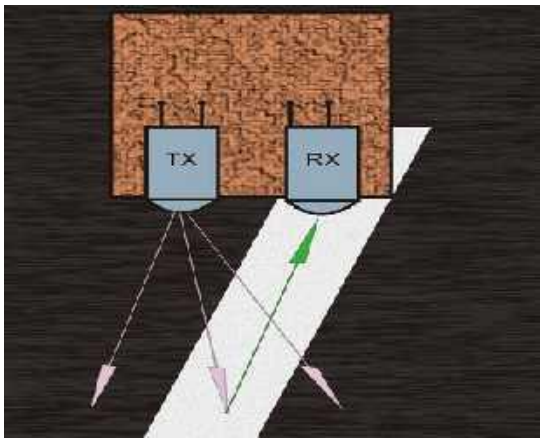

Figure 3. Working of IR Sensors

2.3. Microcontroller

The Atmel picoPower ATmega328/P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega328/P achieves throughputs close to 1MIPS per MHz. This empowers system designer to optimize the device for power consumption versus processing speed.

Feature

High Performance, Low Power Atmel AVR 8-Bit Microcontroller Family
• Advanced RISC Architecture
– 131 Powerful Instructions
– Most Single Clock Cycle Execution
– 32 x 8 General Purpose Working Registers
– Fully Static Operation
– Up to 20 MIPS Throughput at 20MHz
– On-chip 2-cycle Multiplier
• High Endurance Non-volatile Memory Segments
– 32KBytes of In-System Self-Programmable Flash program
Memory
– 1KBytes EEPROM
– 2KBytes Internal SRAM

– Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
– Data Retention: 20 years at 85°C/100 years at 25°C(1)
– Optional Boot Code Section with Independent Lock Bits
• In-System Programming by On-chip Boot Program
• True Read-While-Write Operation
– Programming Lock for Software Security
• Atmel QTouch Library Support
– Capacitive Touch Buttons, Sliders and Wheels
– QTouch and QMatrix Acquisition
– Up to 64 sense channels

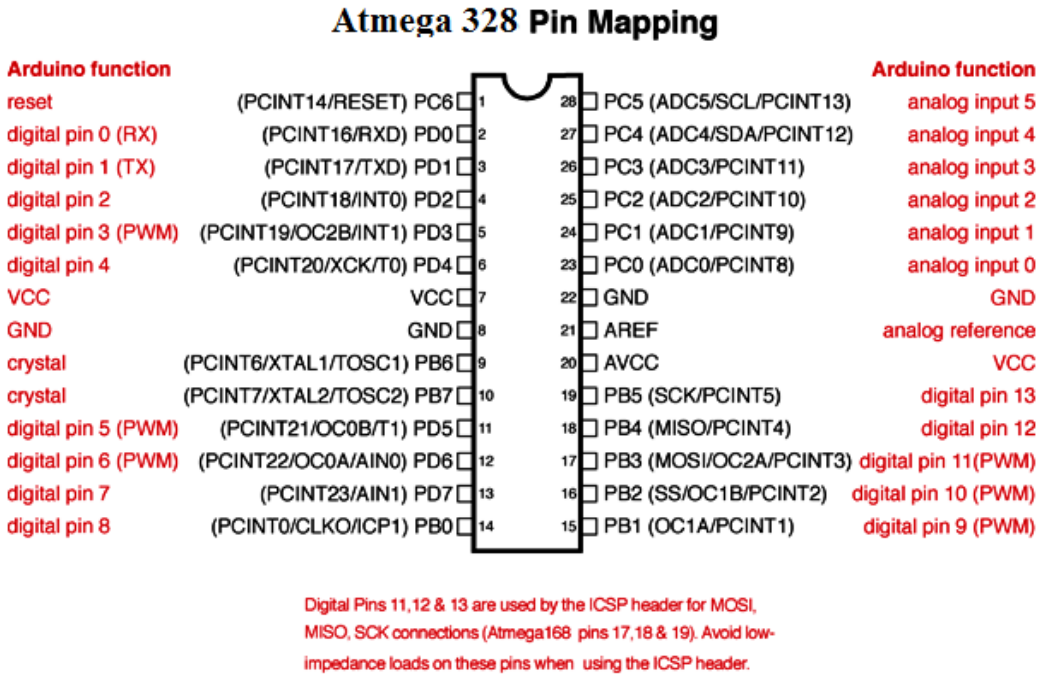

Figure 4. Pin Mapping of Atmega 328p microcontroller [5]

2.4. Motor drivers

This motor controller from Tronixlabs Australia is based on the L298N heavy-duty dual H-bridge controller, which can be used to drive two DC motors at up to 2A each, with a voltage between 5 and 35V DC - or one stepper motor with ease. The controller has fast short-circuit protection diodes, and a nice heatsink to keep the L298N happy.

1. DC motor 1 „+" or stepper motor A+
2. DC motor 1 „-„ or stepper motor A-
3. 12V jumper – remove this if using a supply voltage greater than 12V DC. This enables power to the onboard 5V regulator
4. Connect your motor supply voltage here, maximum of 35V DC. Remove 12V jumper if >12V DC
5. GND
6. 5V output if 12V jumper in place, ideal for powering your Arduino (etc)
7. DC motor 1 enable jumper. Leave this in place when using a stepper motor. Connect to PWM output for DC motor speed control.
8. IN1
9. IN2
10. IN3
11. IN4
12. DC motor 2 enable jumper. Leave this in place when using a stepper motor. Connect to PWM output for DC motor speed control.
13. DC motor 2 „+" or stepper motor B+
14. DC motor 2 „-„ or stepper motor B-

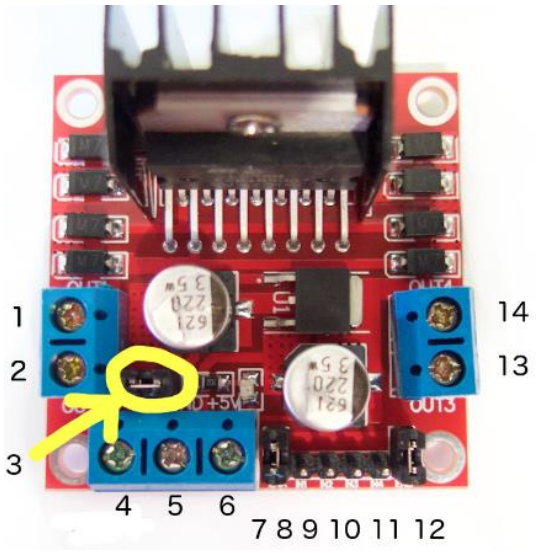

Figure 5. The circuit of motor drivers [6]

3. REAL TIME TASK SCHEDULING

We built a light-weighted and high-speed robot because points are awarded based upon the distance covered and the speed of the overall robot. Therefore, we used two high speed motors and a highly sensitive signal conditioning circuit. The body weight and wheels' radius have effects on the speed, too. The microcontroller sends instructions to the driver after processing the data received from sensors. The driver powers the motors according to the inputs. Actually the driver supplies positive voltage to one of the motor pins and negative voltage to the other. There are five states of movement:

• To move forward; both of the motors are turned on and rotate forward simultaneously.

• To move left; the right motor is turned on and the left motor is turned off.

• To move right; the left motor is turned on and the right motor is turned off.

• To move left fast; the right motor rotates forward and the left motor rotates backward.

• To move right fast; the left motor rotates forward and the right motor rotates backward.

Most embedded system applications need to react to the inputs or environment changes in real time, which means that the accuracy of computations is as important as their timelines. Furthermore, digital control algorithms need a fixed sampling time interval for measuring inputs and delivering output commands. Therefore, the idea of applying interrupts for task scheduling is introduced in this work.

3.1. The Quadratic Line-Detection Algorithm

A better way of detecting the line position, compared to the other simple line-following robots, by using a quadratic interpolation technique is introduced. Eight reflective optical sensors were used, and the coordinate of the leftmost sensor was 0. To find out the correct position of the black line, we had to locate three consecutive sensors with higher output readings than the other five sensors as shown in fig. 6.

Assume that the coordinates of these 3 sensors are x1, x1+1, and x1+2, and the true shape of the sensor output values are in the range of [x1, x1+2] which can be approximated by a quadratic curve. One can then find the following relationships between the coordinates of the sensors and the output values:

$$y_1 = ax_1^2 + bx_1 + c \quad (1)$$

$$y_2 = a(x_1 + 1)^2 + b(x_1 + 1) + c \quad (2)$$

$$y_3 = a(x_1 + 2)^2 + b(x_1 + 2) + c \quad (3)$$

The coordinate value, at which the output value of the quadratic curve is the aximum, is considered as the true position of the line. By using the basic calculus, one would know that the coordinate value is:

$$x = \frac{-b}{2a} \quad (4)$$

$$a = \frac{y_1 + y_2 - 2y_3}{2} \quad (5)$$

$$x = y_2 - y_1 - 2ax_1 - a \quad (6)$$

It is assumed that the coordinate for the center position of the line-following robot is 0. Therefore, the error *e* between the line position and the center position of the robot is:

$$e = 0 - x = -x \quad (7)$$

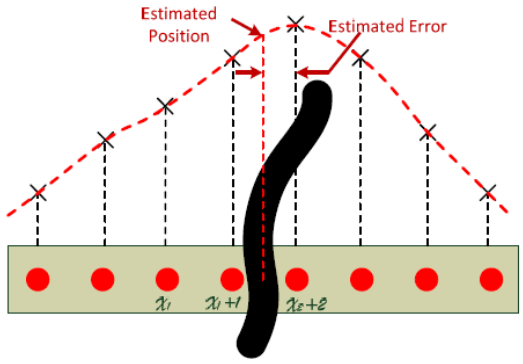

Figure 6. The line detection algorithm via quadratic interpolation.

3.2. PID tracking control algorithm

The error between the center of the sensors and the track to be followed was then processed by the PID controller to generate velocity commands for the right and left wheels. The PID controller
The PID controller works as follows: first calculate the current position and then calculate the error based on it. The speed of the motors will be given by the magnitude of the error using the proportional element. If the error does not drop to zero, the integrator enters the action until the robot is on the line. In order to reduce the effect of the oscillation around the line and to improve the response to disturbances, the derivative element of the controller enters into action.

Pseudo Code for the PID Controller:
Kp = 10
Ki = 1
Kd = 100
offset = 45 ! Initialize the variables Tp = 50
integral = 0 ! the place where integral value will be stored
lastError = 0 ! place where last error value will be stored
derivative = 0 ! place where derivative value will be stored
Loop forever
LightValue = read sensors ! read sensors.
error = -x ! calculate the error using equation (7).
integral = integral + error ! calculate the integral
derivative = error - lastError ! calculate the derivative
*Turn = Kp*error + Ki*integral + Kd*derivative*
powerA = Tp + Turn ! power level for motor A
powerB = Tp - Turn ! power level for motor B
MOTOR A direction=forward
power=PowerA
MOTOR B direction=forward
power=PowerB
lastError = error ! save the current error
end loop forever ! do it again.

PID controller requires the *Kp*, *Ki* and *Kd* factors to be set to match wheeled line follower robot's characteristics and these values depends on robot structures, actuators, sensors and other electronic circuits. There is no easy way to calculate *Kp*, *Ki* and *Kd* factors. It requires manual trial and error method until you get the desired behavior. We defined these factors following these guidelines;
• Start with low speed and setting values of *Kp*, *Ki* and *Kd* to 0.
• Then, try setting *Kp* to a value of 1 and observe the robot. The goal is to get the robot to follow the line even if it is very wobbly. If the robot overshoots and loses the line, reduce the value of *Kp*. If the robot cannot navigate a turn or seems sluggish, increase the *Kp* value.

• Once the robot is able to follow the line, set *Kd* value to 1 and then try increasing this value until you see less wobble.
• Once the robot is fairly stable at following the line, assign a value of .5 to 1.0 to *Ki*. If the *Ki* value is too high, the robot will jerk left and right quickly.
If it is too low, you won't see any perceivable difference. Since integral is cumulative, the *Ki* value has a significant impact. You may end up adjusting it by .01 increments.
• Once the robot is following the line with good accuracy, you can increase the speed and see if it is still able to follow the line. Speed affects the PID controller and will require retuning as the speed changes.

4. RESULTS AND DISCUSION

A line following robot is programed with simple (on/off) control as a comparison purpose in evaluating the performance of the dynamic algorithm controlled robot.
The results of the experiment are summarized in Table 2. From the data in the table, it can be observed that dynamic PID algorithm controlled robot has better performance in every criteria listed in the table compared to simple (on/off) control robot. The dynamic algorithm controlled robot has higher velocity, consumes less time to complete one whole circuit, tracks the line smoother and has lower tendency to astray from line compared to uncontrolled robot. Therefore this system can be used in training undergraduate students on dynamic PID algorithm control system, its application and implementation in the real world and the advantages that it offers.

Table 2. Result for Line Following Robot

Criteria	Dynamic PID algorithm	Simple (on/off)
Time to complete one whole circuit	47.6s	71.4s
Line tracking	Smooth	No so Smooth
Velocity	0.2m/s	0.14m/s
Tendency to astray from line	low	high

5. CONCLUSIONS

The designed wheeled line follower mobile robot has eight infrared sensors on the bottom for detecting the line. The controller board includes ATmega 328P microcontroller and the motor driver L298 which were used to control the direction and the speed of motors. The proposed dynamic PID algorithm derives the line follower locomotion by adequately combining the information from sensor module.
The proposed algorithm can successfully achieve target following in various scenarios, including straight line and circular motion, sharp-turn motion and S-shape line tracking.

REFERENCES

[1] Peacock, F. (2008). How the PID algorithm works and why it works, Retrieved from www.PIDTuning.com
[2] http://www.ijecse.org/wp-content/uploads/2012/06/Volume-3Number-1PP-51-56x.pdf
[3]https://play.google.com/store/apps/dev e loper?id=Hobbyprojects.com&hl=en
[4]http://ieeexplore.ieee.org/xpl/login.jsp ?tp=&arnumber=6566505&url=http%3A %2F%2Fieeexplore.ieee.org%2Fiel7%2 F6559347%2F6566328%2F06566505.p df%3Farnumber%3D6566505
[5]http://ww1.microchip.com/downloads/ en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf
[6]https://tronixlabs.com.au/robotics/mot or-controllers/l298n-dual-motor-controller-module-2a-australia/