In the following, a translation of the report of my bachelor's thesis ("Final Project 1" in Electrical Engineering – Control) is provided. The original report was written in Persian and was successfully evaluated, resulting in a high grade of 19.5 out of 20 in 2019:

"Design and Construction of a Remote PID Control for a Distance-Sensitive Robot"

Abstract

The objective of this project is to develop a distance-sensitive robot capable of maintaining a safe distance from objects in front of it while responding to input commands for movement and positioning via a mobile device. The robot's control is achieved through a PID controller, which receives control variables and distance information from a smartphone application. Using an Arduino microcontroller, the robot is powered by a battery. An LCD display, with 16 characters, is integrated into the robot to provide visual feedback of the sensor output, completing the control loop.

This robot holds crucial applications in distance maintenance technology, such as maintaining safe distances between vehicles and even in autonomous and intelligent strollers to ensure a safe separation from the carrier.

1- Introduction

PID consists of three separate components: Proportional, Integral, and Derivative. Each component takes the error signal as input and performs its operation. Finally, their outputs are summed together. The output of this combination, which is the output of the PID controller, is sent to the system for error correction. The standard formula for PID is as follows:

$$\mathbf{Output(t)} = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau + T_d \frac{de}{dt} \right)$$

Therefore, the transfer function of a PID controller can be represented as follows:

$$G_c = K_p + \frac{K_i}{s} + K_d s$$

In many controllers, the derivative term is often omitted and the control is implemented in the form of PI due to its sensitivity to noise and implementation difficulties. The output of the PID controller, denoted as *U(t)*, is calculated based on a proportional contribution of the current system error (current performance), the cumulative sum of system errors (past behavior), and the derivative of the current error (linear estimation of future behavior). It is then applied to the system for error correction. The coefficients *K*, *Ti*, and *Td* can also be calculated optimally using established methods such as transfer functions, although in practical applications, they can be approximately determined through trial and error and by observing the system's behavior.

PID controllers are used to compensate for system output and steady-state errors. The proportional controller is employed to adjust the system output bounds, the integral controller is used to eliminate steady-state errors, and the derivative controller is utilized to improve transient response.

1-1- Objective and Performance

The objective of this project is to change the speed of a DC motor by varying the distance to the target object detected by an ultrasonic sensor. For this purpose, a PID control system is embedded within the microcontroller program, aiming to maintain a constant distance between the robot and the target object. Therefore, after passing a reference distance, the motor direction is changed, and the greater the distance (in both directions relative to the reference distance), the higher the motor speed will be.

Figure 1-1: Robot's Physical Appearance

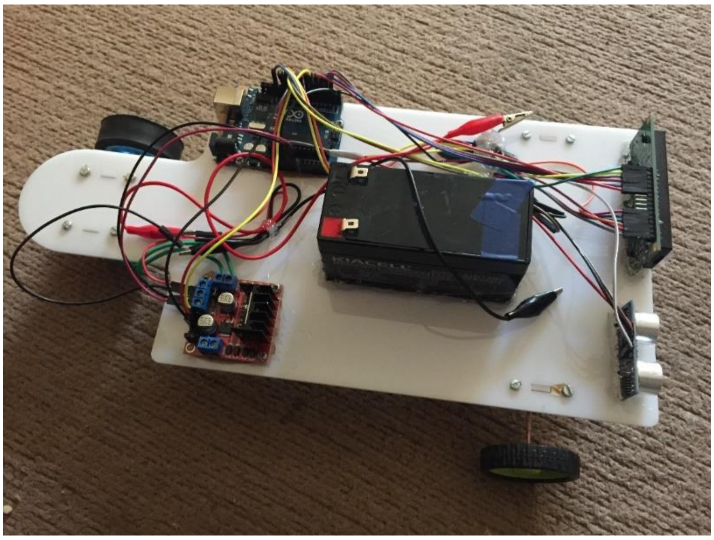


Table 1-1: Main Parts of Robot Construction

Device	Device Details	Device
Plexiglass	Simple	2
Ultrasonic Sensor	SRF05	1
DC Motor	12 V	1
LCD	32 Character	1
Microcontroller	AVR Atmega32+Arduino UNO	1
Driver	L298N Module	1

2- Introduction of Robot Parts

1-2 ATmega32A Microcontroller

The ATmega32a microcontroller is a powerful and versatile microcontroller from the Atmel AVR family. It is widely used in various embedded systems and electronic projects due to its advanced features and capabilities. The ATmega32a is based on the RISC architecture and operates at a clock frequency of up to 16 MHz. It offers a wide range of features, including:

- High-performance CPU: The ATmega32a is equipped with an 8-bit AVR CPU that provides fast and efficient processing of instructions.
- RAM and EEPROM: The microcontroller includes SRAM for temporary data storage and EEPROM for non-volatile data storage.
- Peripherals: It features a variety of built-in peripherals, such as timers/counters, UART, SPI, I2C, ADC, and PWM, enabling seamless integration with external devices and sensors.
- GPIO Pins: The ATmega32a offers a significant number of General Purposes Input/Output (GPIO) pins, allowing for extensive connectivity and control options.
- Low Power Consumption: It provides power-saving sleep modes and efficient power management features, making it suitable for battery-powered applications.
- Robust Communication: The microcontroller supports various communication protocols, including USART, SPI, and I2C, facilitating seamless communication with other devices and peripherals.
- Wide Operating Voltage Range: It can operate within a broad voltage range, typically from 2.7V to 5.5V, providing flexibility in different power supply configurations.

These features make the ATmega32a microcontroller a popular choice among electronics enthusiasts and professionals for a wide range of applications, including robotics, automation, IoT devices, and more.

Figure 1-2: Pinout ATmega32A

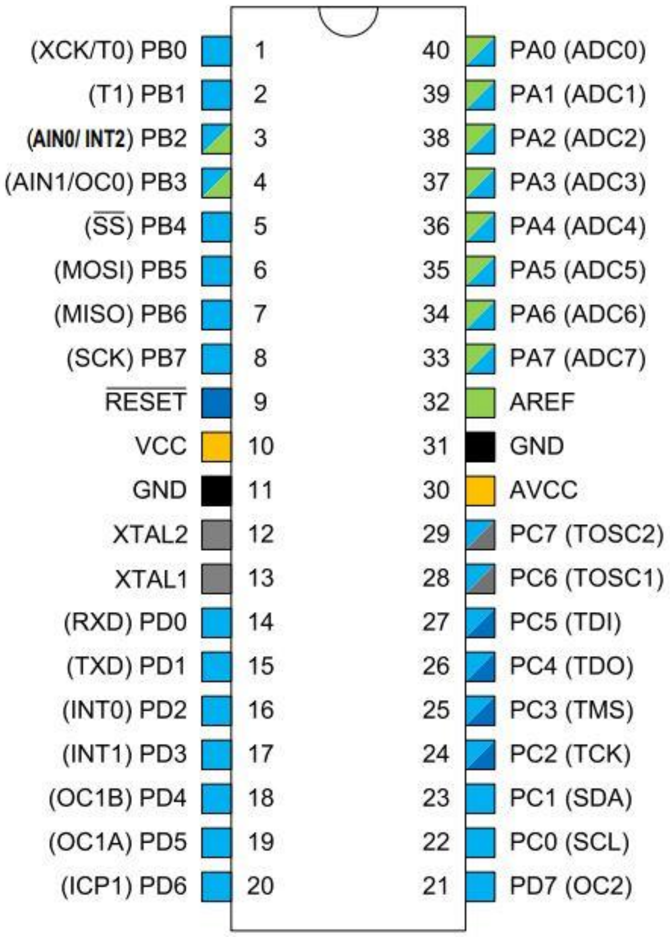
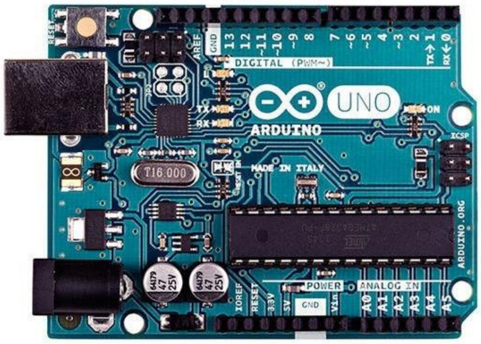


Figure 2-2: Arduino Physical Appearance



2-2 Ultrasonic SRF05

The SRF05 ultrasonic sensor is a widely used distance-measuring device that utilizes sound waves to determine the distance between the sensor and an object. It is a popular choice in robotics, automation, and various electronic projects due to its accuracy and reliability.

The SRF05 sensor operates on the principle of sonar, similar to how bats navigate in the dark. It emits ultrasonic waves, which are inaudible to human ears, and measures the time it takes for the waves to bounce back after hitting an object. This time measurement is then used to calculate the distance between the sensor and the object.

The working principle of the SRF05 sensor involves the following steps:

- Trigger Pulse: The microcontroller or host device sends a short high-level pulse to the sensor's trigger pin. This triggers the sensor to start emitting ultrasonic waves.
- Ultrasonic Wave Emission: Upon receiving the trigger pulse, the SRF05 sensor emits a burst of ultrasonic waves.
- Wave Propagation: The emitted ultrasonic waves propagate through the air and travel towards the object in front of the sensor.
- Wave Reflection: When the waves encounter an object, they bounce back or reflect off its surface.
- Echo Reception: The SRF05 sensor detects the reflected waves using its receiver element.
- Time Measurement: The sensor measures the time it takes for the emitted waves to travel to the object and back. It starts a timer upon sending the trigger pulse and stops it when the echo is received.
- Distance Calculation: By knowing the speed of sound in air and the time is taken for the round trip of the ultrasonic waves, the distance between the sensor and the object can be calculated using the formula: Distance = Speed × Time / 2.

The SRF05 ultrasonic sensor offers several advantages, including non-contact distance measurement, high accuracy, wide detection range, and simple interfacing with microcontrollers or other control systems. These features make it valuable in applications such as obstacle detection, distance sensing, robotics navigation, and many more.

Figure 2-2: SRF05 Timing Diagram, Mode 1

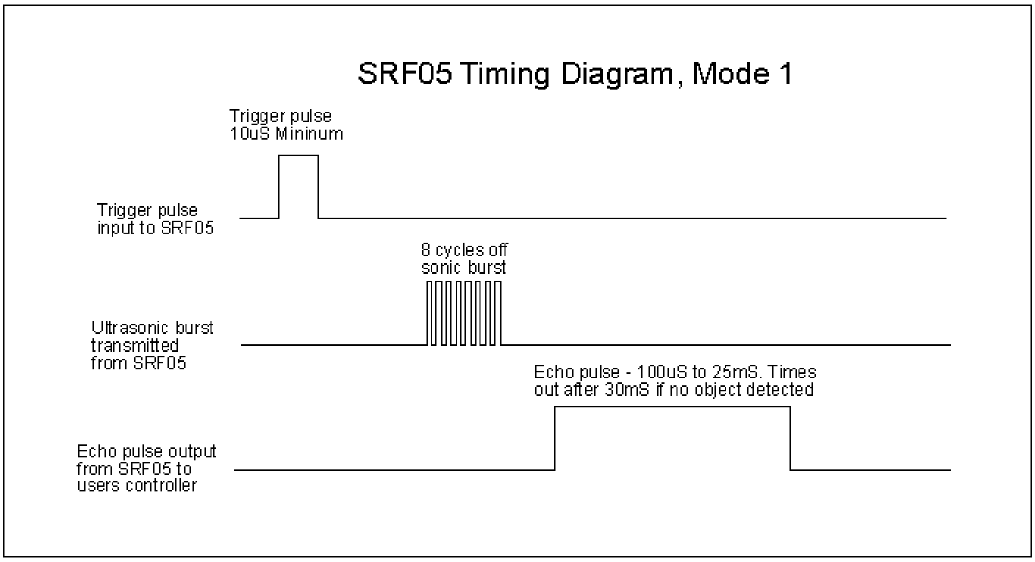
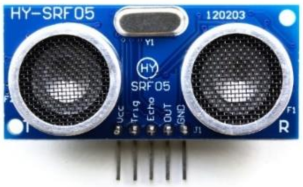


Figure 3-2: Ultrasonic SRF05 Physical Appearance



3-2 L298 driver

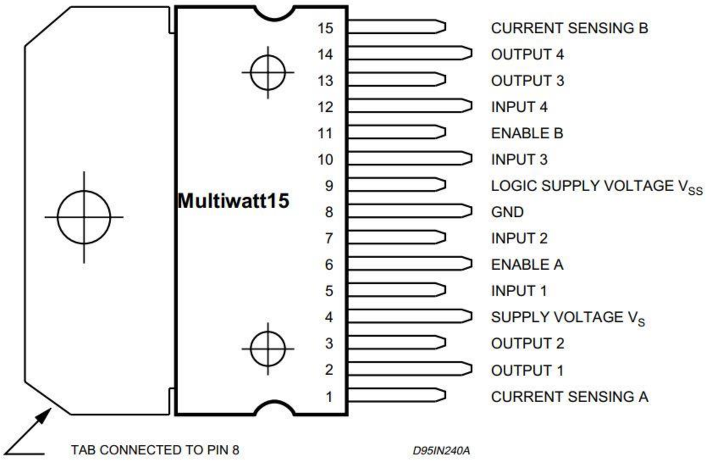
The L298 driver is a popular integrated circuit that controls and drives DC and stepper motors in various electronic applications. It provides a convenient and efficient solution for motor control, making it widely used in robotics, automation, and other projects that require precise motor movements.

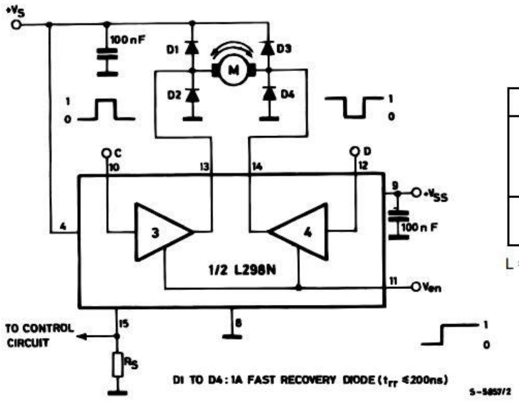
The L298 driver operates as a dual H-bridge, which means it can independently control the direction and speed of two motors. It consists of four high-power output transistors arranged in an H-bridge configuration, allowing bidirectional control of the motors. The H-bridge enables the motor to rotate forward or backward by changing the direction of the current flow.

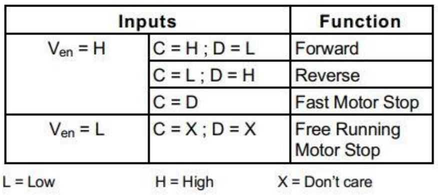
Here's how the L298 driver works:

- Motor Connections: The L298 driver is connected to the power supply and the motors. Each motor is connected to a pair of output terminals on the L298 driver.
- Control Inputs: The L298 driver has control inputs that determine the motor's direction and speed. These inputs include two enable pins (ENA and ENB) to activate the motor outputs, and four direction control pins (IN1, IN2, IN3, IN4) to set the motor rotation direction.
- Direction Control: By setting the appropriate logic levels on the IN1, IN2, IN3, and IN4 pins, the L298 driver controls the motor's direction. For example, to make the motor rotate forward, IN1 and IN4 should be set to logic high (or logic low, depending on the configuration) while IN2 and IN3 are set to logic low (or logic high).
- Speed Control: The speed of the motor is controlled using a technique called Pulse Width Modulation (PWM). By applying a PWM signal to the enable pins (ENA and ENB), the L298 driver adjusts the motor's average voltage, thereby controlling its speed. Higher duty cycles result in faster motor speeds, while lower duty cycles slow down the motor.
- Current Limiting: The L298 driver incorporates current sensing and limiting capabilities to protect the motor and the driver circuitry from excessive current. It monitors the motor's current and adjusts it to a safe level, preventing damage due to overcurrent conditions.

Figure 4-2: the L298 Driver Pinout



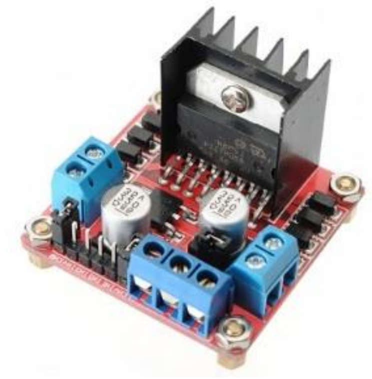




Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Figure 4-2: the L298 Driver's Physical Appearance

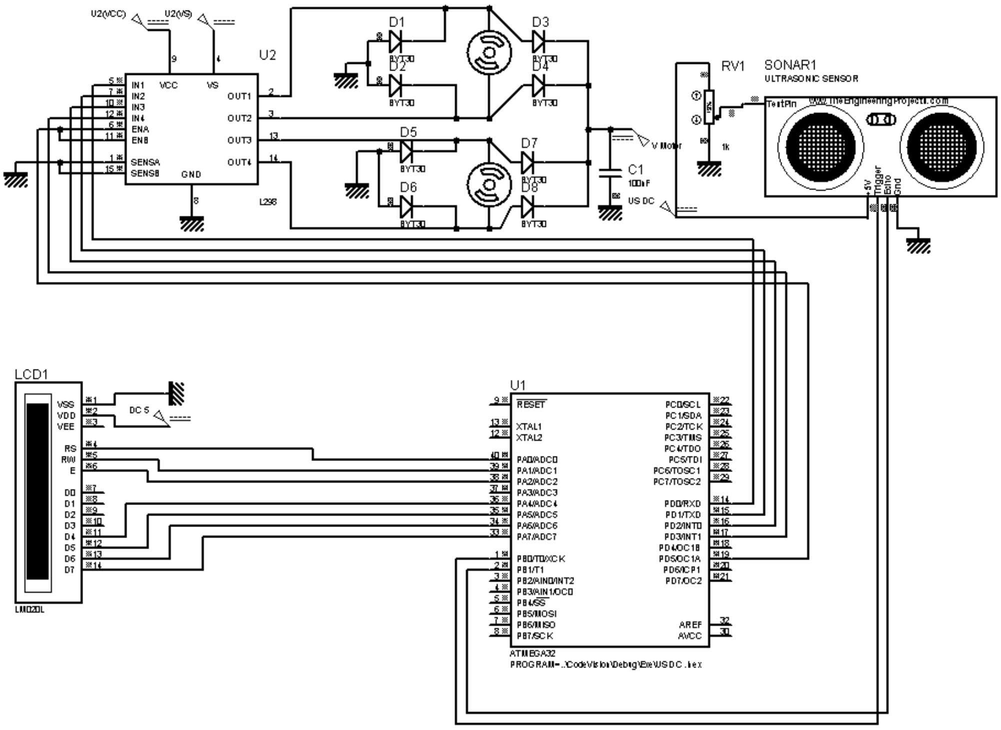


3- Design

1-2 System Design in Proteus 8

At the beginning of the work, to simulate the system before implementation on the microcontroller, Proteus software was used. An overall view of the system was drawn in this software, and the result was observed:

Figure 1-3: Overall View of the System



1-2 Coding with CodeWizard

The primary software for managing the Robot is coded in the C programming language. The code is presented below, and each segment is explained in the accompanying comments.

```
include <LiquidCrystal.h>#

const unsigned int TRIG_PIN=9;

const unsigned int ECHO_PIN=10;

int EN = 6;

int IN1 = 7;

int IN2 = 8;

float U , U1;

float kp1 = 1.5 , kd1 = 0.0065 , ki1 = 0.055;

float kp , kd , ki;
```

```
int fixdistance = 30.0;
int error = 0 ;
float errorold = 0;
unsigned long start1 , finished , finished1 , elapsed;
float derror , ierror;
float dt =0.15;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {

lcd.begin(16, 2);
Serial.begin(74880);
pinMode(EN, OUTPUT);
pinMode(IN1, OUTPUT);
pinMode(IN2, OUTPUT);
pinMode(TRIG_PIN, OUTPUT);
pinMode(ECHO_PIN, INPUT);
lcd.clear();
lcd.print("   Instrumentation          Project");
delay(2000);
lcd.clear();
{

void loop() {

start1= micros();
lcd.clear();
digitalWrite(TRIG_PIN, LOW);
delayMicroseconds(2);
digitalWrite(TRIG_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);
const unsigned long duration= pulseIn(ECHO_PIN, HIGH);
int distance= duration/29/2;                         //Sound speed is about 29 cemtimeters per microsecend
if (distance < 200 & distance > 4)
}
error = fixdistance - distance;
derror = (error - errorold)/dt;
ierror = ierror + (error + errorold)*dt/2;
if (error < 2 && error >= -2)                    //These parameters are for Fuzzy Controller //
}
kp = 0.3 , kd = 0.0025 , ki = 0.03;
{
if (error < 10 && error >= 2)
}
kp = 1.2 , kd = 0.003 , ki = 0.05;
{
if (error < 20 && error >= 10)
}
kp = 1.7 , kd = 0.004 , ki = 0.06;
{
if (error >=20)/
}
kp = 2 , kd = 0.0055 , ki = 0.07;
{
if (error < -2  && error >= -10)
}
kp = 0.3 , kd = 0.003 , ki = 0.05;
{//
if (error < -10  && error >= -20)
}
kp = 1.2 , kd = 0.004 , ki = 0.06;
{
```

```
if (error <= -20)//

}

kp = 2 , kd = 0.0055 , ki = 0.07;

{


U1 = kp1*error + kd1*derror + ki1*ierror;

U = kp*error + kd*derror + ki*ierror;          // This controller will work for U between -
100 to 50

errorold = error;

if (U1 > 0.5)

}

digitalWrite(IN1 , HIGH);

digitalWrite(IN2 , LOW);

if (abs(U1)+40 < 254)

}

analogWrite(EN , gerd(100 + abs(U1)));

lcd.clear();

lcd.print(distance);

delay(5);

{

{

if (U1 < -0.5)

}

digitalWrite(IN2 , HIGH);

digitalWrite(IN1 , LOW);

if (abs(U1)+40 < 254)

}

analogWrite(EN , gerd(100 + abs(U1)));

lcd.clear();

lcd.print(distance);

delay(5);

{

{
```

```
if (U1 > -0.25 && U1 < 0.25)
}
digitalWrite(IN1 , LOW);
digitalWrite(IN2 , LOW);
{
Serial.println(distance);
{
else
}
if (distance <= 4)
}
lcd.clear();
lcd.print("Too close!");
delay(5);
{
if (distance >= 200)
}
lcd.clear();
lcd.print("Too far!");
delay(5);
{
{
delay(5);
finished=micros();
elapsed=finished-start1;
dt=elapsed/1000000.0;
start1 = elapsed = 0;
{
```

4- Conclusion

In conclusion, the report highlights the successful design and construction of a remote PID control system for a distance-sensitive robot. The utilization of Arduino, ultrasonic sensor, motor driver, LCD, and electrical motor, along with the aid of Proteus software for system design and simulation, has yielded positive results.

The remote PID control system allows the robot to maintain a constant distance from a target object by adjusting the speed of the electric motor based on the readings from the ultrasonic sensor. This functionality was achieved through the integration of the Arduino microcontroller, which processed the sensor data and implemented the PID control algorithm. The use of Proteus software played a vital role in the development process by allowing for the simulation of the system before physical implementation. This enabled the identification of potential issues and the fine-tuning of the system design, resulting in improved performance and functionality.

Overall, the successful implementation of the remote PID control system showcases the effectiveness of the Arduino platform, the ultrasonic sensor, the motor driver, and the LCD, and the integration of these components into a cohesive robotic system. The report serves as a valuable reference for future projects involving similar designs and provides insights into the application of PID control in robotics.