Digital PID Controller

Problem definition:

Precise control of DC motor speed is crucial in various applications such as robotics, automation, and industrial systems. The main challenge is to maintain a constant speed despite external disturbances or variable loads. This project aims to implement a PID (Proportional-Integral-Derivative) controller on a microcontroller to adjust motor speed in real-time, ensuring it matches the desired value.

Requirements and Specifications:

Hardware Requirements:

- DC Motor with encoder
- Motor Driver (L298N)
- Microcontroller (e.g., ATmega32)
- Power Supply
- LCD Display (optional)
- Serial communication interface to PC

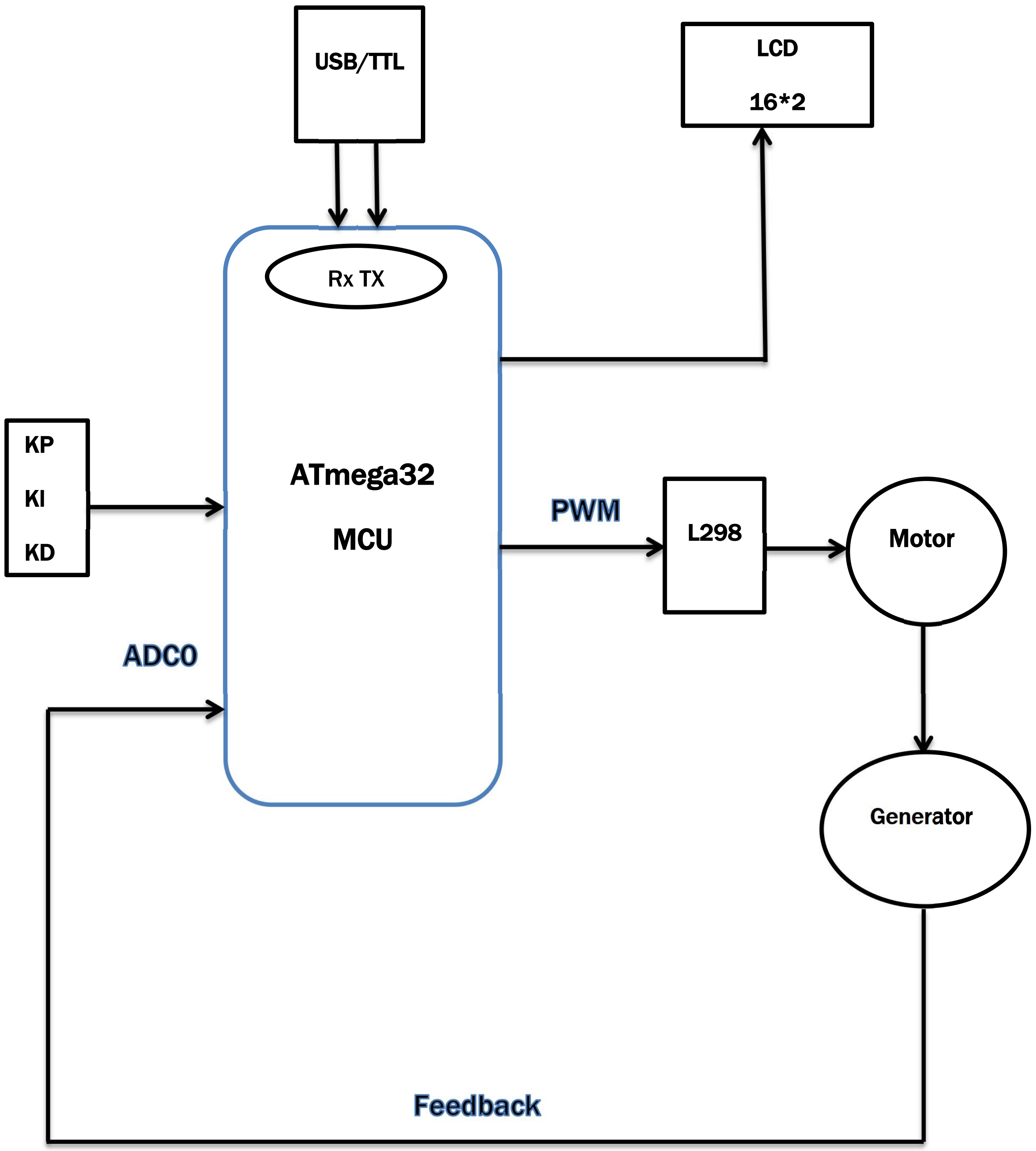
Software Requirements:

- Atmel Studio (Embedded C)
- Serial plotter or custom PC software to visualize data

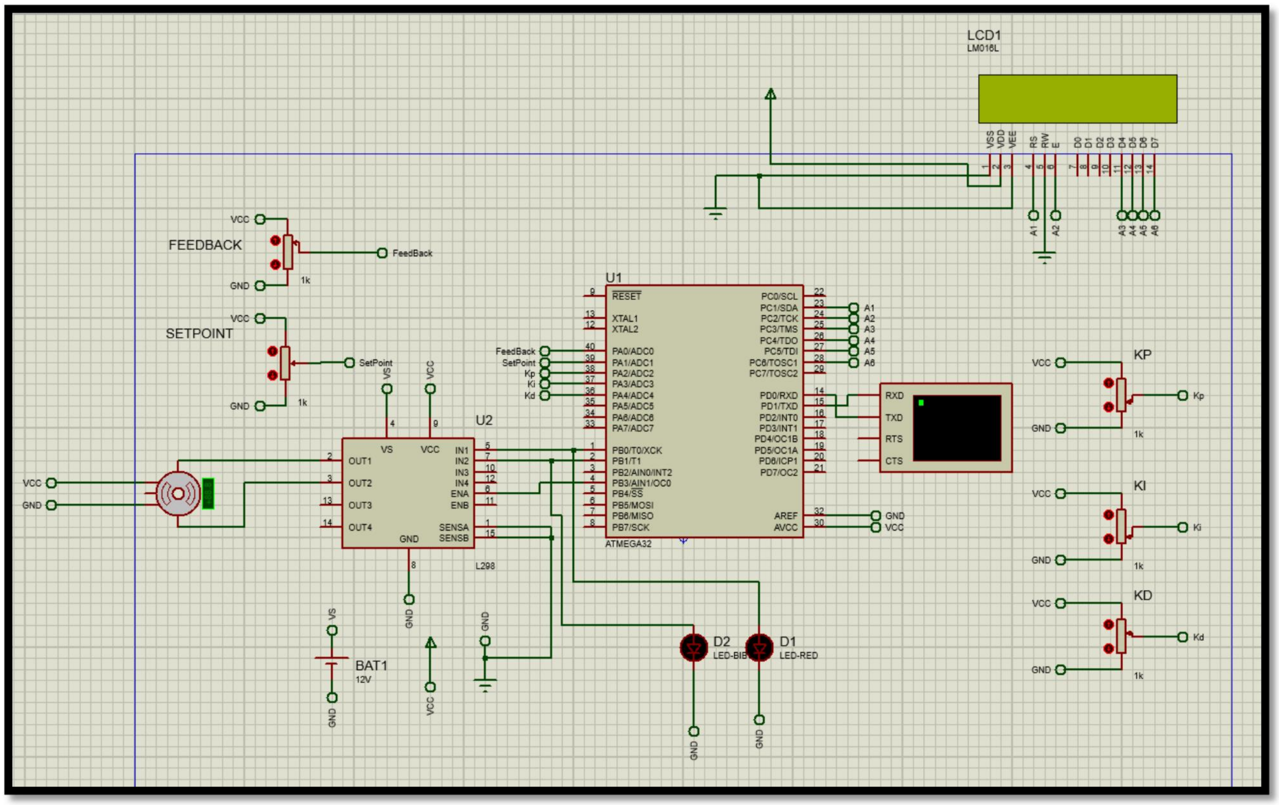
Specifications:

- Real-time speed control using PID algorithm
- Manual input for Kp, Ki, Kd values
- Display of actual vs desired speed in RPM

System Block Diagram:



Circuit Schematic and Description:



1. ATmega32 (Microcontroller):

This is the brain of the system. It receives the Set point value (desired motor speed) via the TTL interface and the Feedback value (actual speed) from the motor's sensor. It calculates the error between them and applies the PID algorithm using the tuned Kp, Ki, and Kd values (adjusted via potentiometers). The output is used to control the motor driver

2. L298 Motor Driver:

This is the driver circuit that powers and controls the direction and speed of the DC motor. It takes control signals from the ATmega32 and drives the motor accordingly. It allows bidirectional control and handles higher current that the microcontroller can't provide directly.

3. DC Motor (with Feedback):

This is the actuator whose speed is being controlled. A feedback mechanism (like a speed sensor or encoder) measures the actual motor speed and sends it back to the microcontroller. This real-time feedback is essential for closed-loop PID control.

4. TTL to USB (Virtual Terminal):

This interface is used to send and receive data between the microcontroller and a computer. It used to input the set point and possibly to monitor values like the current speed, error, or PID output on a serial terminal.

5. Potentiometers (Kp, Ki, Kd):

These are variable resistors used to manually adjust the PID gains:

- Kp (Proportional Gain): Reacts to the current error.
- Ki (Integral Gain): Reacts to the accumulation of past errors.
- Kd (Derivative Gain): Predicts future error based on the rate of change.
  Each potentiometer is connected to an analog input of the ATmega32, allowing real-time tuning of the PID parameters.

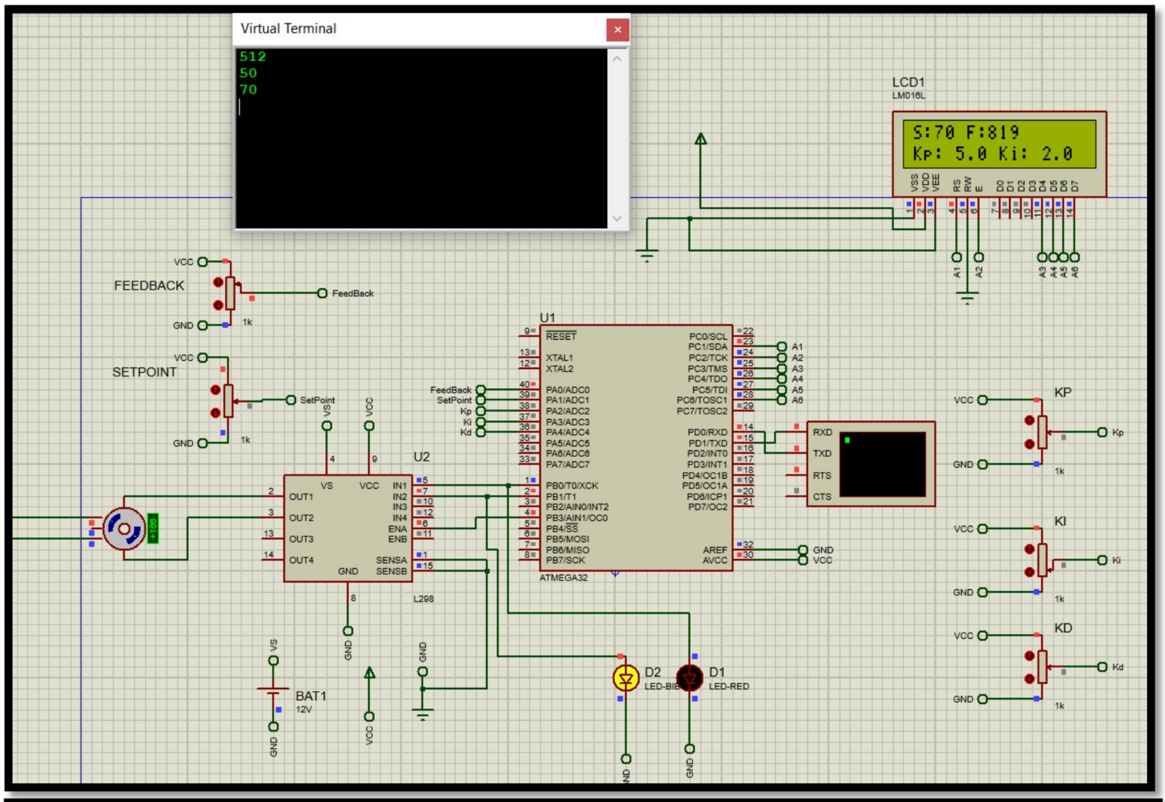
6. LCD Display (e.g., 16x2 Character LCD):

The LCD is used to display real-time system information. It is connected to the ATmega32 microcontroller and is updated continuously during operation. Typical information shown on the LCD includes:

- Current motor speed (feedback)
- Set point
- Error value
- Current values of Kp, Ki, and Kd

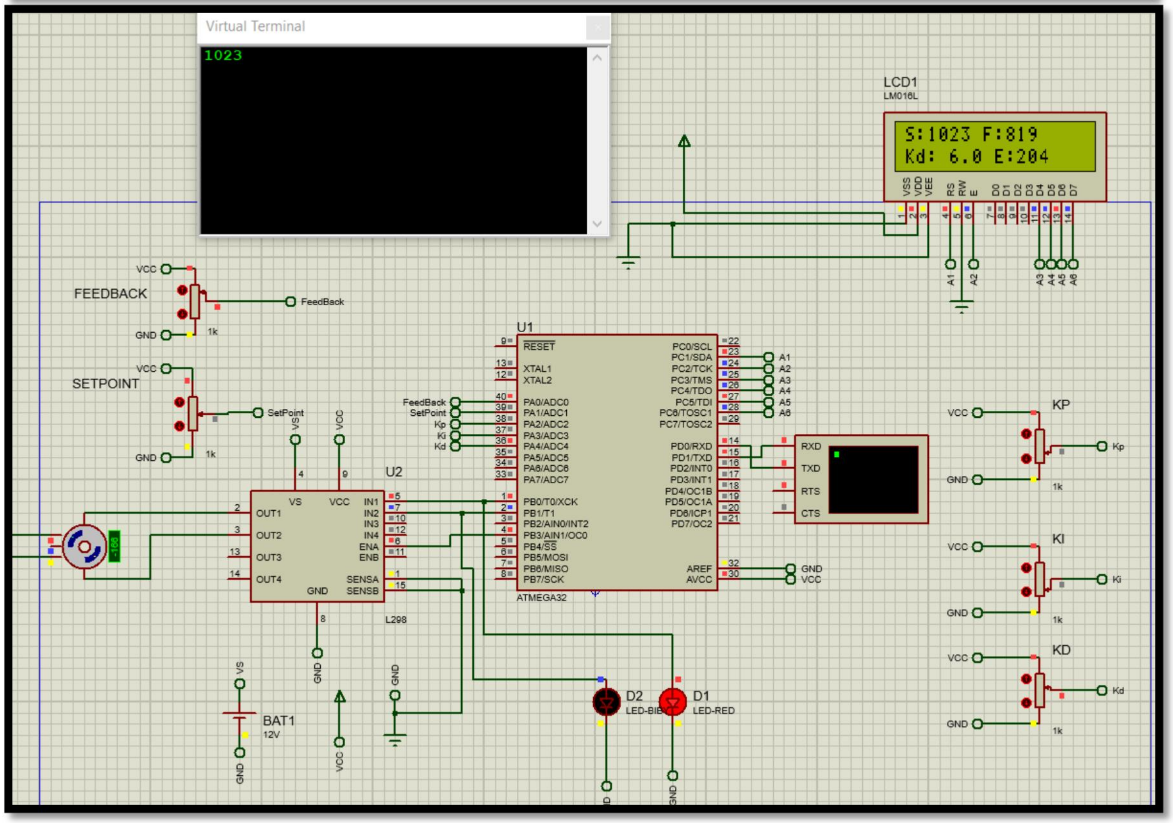
This makes the LCD a useful interface for monitoring and debugging, especially when tuning PID parameters or observing system performance during testing.

Simulation:



```
512
50
70
```

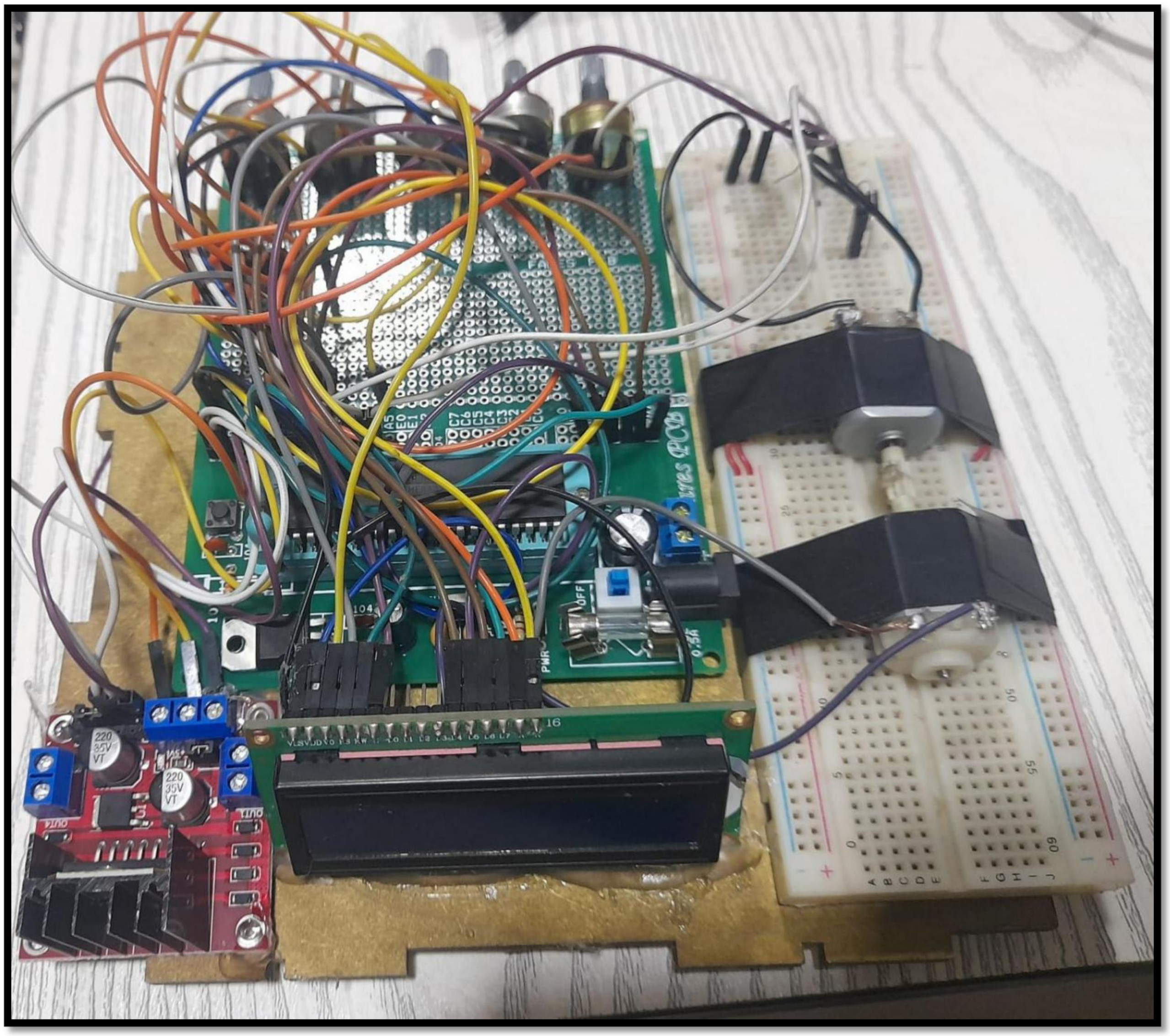
```
S:70 F:819
Kp: 5.0 Ki: 2.0
```

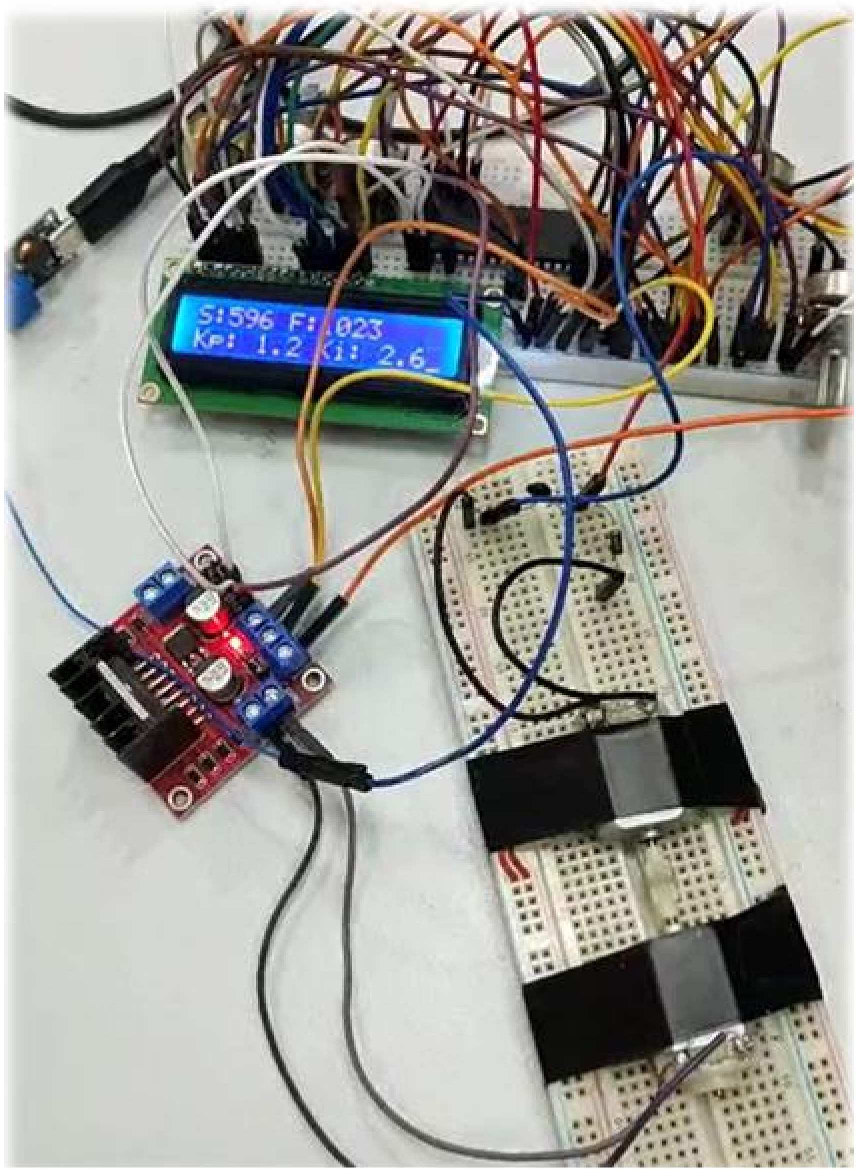


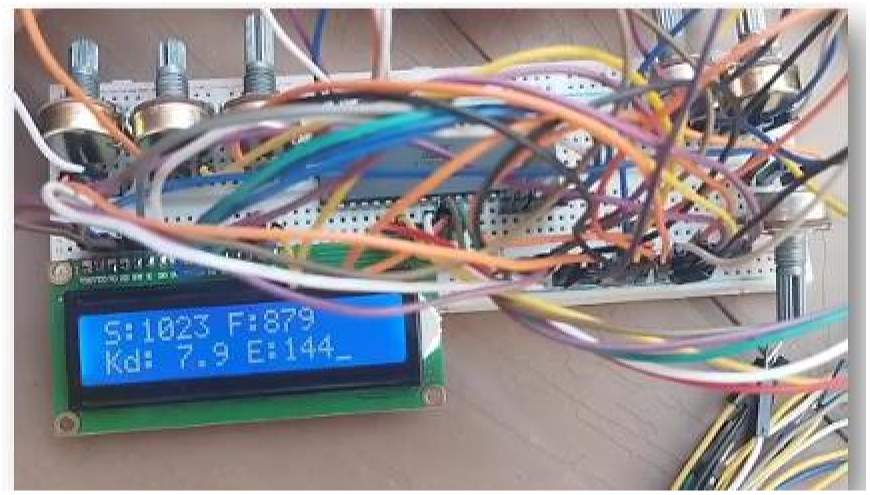
```
1023
```

```
S:1023 F:819
Kd: 6.0 E:204
```

Real Time:




S:596 F:1023
Kp: 1.2 Ki: 2.6_



Conclusion:

In this project, a digital PID controller was successfully implemented using the ATmega32 microcontroller to control the speed of a DC motor. The system uses real-time feedback from the motor to calculate error and dynamically adjust its output using the proportional (Kp), integral (Ki), and derivative (Kd) gains. These gains were manually tuned using potentiometers, providing a flexible and hands-on method for optimizing control performance.

The L298 motor driver effectively interfaced between the microcontroller and the DC motor, allowing for smooth and bidirectional motor control. Set point values were received via a TTL to USB interface, enabling easy communication with a PC or serial terminal. Additionally, an LCD display provided real-time monitoring of system parameters, including set point, feedback speed, PID output, and gain values.

Overall, this project demonstrates a practical application of embedded systems and control theory. It not only deepens understanding of PID control algorithms but also enhances skills in microcontroller programming, sensor integration, and real-time system design.