LINE FOLLOWER ROBOT WITH PID ADAPTIVE CRUISE CONTROL

Elif Nur Baysar
Student, EEE
Abdullah Gul University
elifnur.baysar@agu.edu.tr

Teresa Sánchez Sarria
Student, IE
Abdullah Gul University
teresa.sarria@agu.edu.tr

Mehmet Fatih Göğüş
Student, EEE
Abdullah Gul University
mehmetfatih.gogus@agu.edu.tr

Oğuzhan Alasulu
Student, EEE
Abdullah Gul University
oguzhan.alasulu@agu.edu.tr

Abstract- This paper presents the design, implementation and performance analysis of a line following robot with PID adaptive cruise control using PID algorithms and record level programming with STM32F103RB microcontroller. The robot uses QTR-8RC sensor for line following and HC-SR04 ultrasonic sensor for distance control. A buzzer is used for real-time feedback. The system's real-time suitability and responsiveness have been successfully observed. PID controllers dynamically adjust motor speeds for accuracy and stability, enabling it to operate smoothly under changing conditions. The robot has been extensively tested, achieving reliable performance in line following and distance control. This work demonstrates the effectiveness of PID control in robotics and provides a foundation for future advances.

I) INTRODUCTION

This project is to design a Line Following Robot with Adaptive Cruise Control equipped with Proportional-Integral-Derivative (PID) control algorithm using register-based embedded system programming. Line tracking is a basic function that enables robots to navigate along predefined paths in controlled environments. This project stands out with its features such as adapting to dynamic conditions and working well enough in terms of precision using a PID controller that integrates the robot's path following performance through continuous real-time adjustments.

PID controller is a widely used closed-loop control system that calculates and minimizes errors based on feedback from the environment. Proportional control improves the system's response by addressing deviations in the system. Derivative control forecasts future mistakes to reduce overshoots and account for transient responses, whereas integral control removes steady-state errors by adding up previous departures. When these three elements are combined, even under difficult situations like tight bends or shifting environmental circumstances, the results are more accurate. In this project, the PID controller is implemented to dynamically adjust the robot's motor speeds and directions based on real-time data from a set of reflection sensors.

The robot is designed to maintain a distance of 60 cm from the moving board and is targeted to dynamically adapt to the board's speed changes. This allows the robot to dynamically adjust its speed according to the board's movement, acting as an adaptive cruise control. In addition, an audible alert has been added to the system to increase functionality by providing real-time feedback with the buzzer. The buzzer complements this function by providing an audible alert to indicate when the robot approaches a predefined distance threshold. If the robot comes closer than the specified distance or moves away from the target distance, the buzzer will be activated. This feature improves user interaction and system monitoring capabilities. The system is designed to accomplish two primary goals: (1) successfully follow the line and complete the path, and (2) maintain the desired distance from the board even when the board's speed changes. To achieve these goals, the robot includes basic hardware components such as an array of IR reflectance sensors for line detection and an ultrasonic sensor for distance measurement. The STM32F103 microcontroller is programmed at the record level to provide direct control over the system components, to provide efficient processing and precise operation. Thus, the robot's goals were completed as desired. Hardware interrupts to ensure the consistency of sensor sampling, noise filtering algorithms for data stability and iterative adjustments of PID parameters to achieve optimum performance were implemented within the scope of the project.

The system was completed with the steps of the robot's hardware design, development and testing of the PID controller. The system was tested extensively under different conditions on a predefined path and provided reliable and effective performance. The integration of the

PID control algorithm significantly increased the accuracy and stability of the robot in terms of reducing overshoot, preventing steady-state errors and minimizing system oscillations.

This project demonstrates the effectiveness of PID control in the designed robot by designing a system that integrates line tracking and adaptive cruise control with an audible alert for advanced functionality, providing a scalable, efficient solution. The integration of structured programming, efficient hardware use and comprehensive test processes ensured that the robot operated reliably and efficiently and formed an important basis for autonomous robotic applications.

II) MECHANICAL COMPONENTS

A. Mechanical Components

In this section, the mechanical and electronic components used in the robot system will be explained. The robot's two-layer mechanical architecture organizes its parts in a systematic manner and the aesthetic appearance is beautiful and effective. Two DC geared motors with internal gears that provide sufficient torque for precise movement are housed inside the robot. These motors are securely fixed using screws and plastic brackets to ensure longevity and alignment. The first layer houses the power system, including the battery and motor driver. The battery, the heaviest component, is strategically placed in the chassis to align the center of mass with the robot's midpoint. It prevents problems such as slipping or excessive leaning during fast turns and regulates balance while moving. The robot's top layer is positioned to house the distance sensor and STM32 microcontroller that provide the adaptive cruise control functionality. The distance sensor is positioned in front of the top layer to accurately measure the distance to an object in front, allowing the robot to maintain a consistent 60 cm gap as required. Additionally, maintaining an optimum distance between the ground and the QTR-8 reflection sensor contributes to stability, especially during sharp turns. This precise positioning enhances the robot's ability to effectively detect and follow the line, even during dynamic movements. This double-layer structure not only provides an orderly layout for components, but also enables the robot to perform reliably and efficiently in both line-following and distance-keeping tasks. This design is ideal for achieving the project's goals by balancing stability and functionality.

B. Electronic Components

1-) Microcontroller (STM32F103RB NUCLEO BOARD):

The STM32F103RB Nucleo board is used to read sensor values and drive DC motors according to these data. This board hosts the ARM Cortex-M3 based STM32F103RB microcontroller. It is suitable for real-time control tasks by operating at a maximum clock speed of 72 MHz. The microcontroller is equipped with 128 KB Flash memory and 20 KB SRAM and includes various input/output peripherals such as 12-bit analog-to-digital converters (ADC), UARTs, timers and GPIO pins to interface with the reflectance sensor array and distance sensors. It provides easy programming and debugging with the integrated ST-LINK/V2-1 debugger and programmer.

In this project, the STM32F103RB Nucleo board was selected to perform line tracking and adaptive cruise control tasks. The microcontroller reads real-time data from sensors, applies the PID control algorithm and precisely adjusts the motor speeds to ensure the smooth operation of the robot. The GPIO pins are integrated with QTR-8 reflection sensors and an ultrasonic distance sensor to provide accurate line recognition and distance measurement. This microcontroller was essential for the dual functionality of the robot, making it an excellent choice for the project, allowing it to accurately follow the line and maintain the desired distance from a moving object.

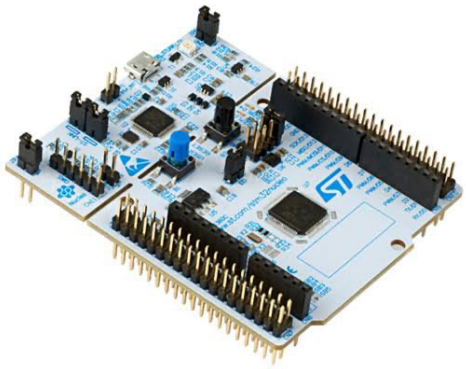

Figure 2.1 - Microcontroller (STM32F103RB)

2-) Motor Driver (L298N)

A motor driver module with L298N dual full bridge driver is used to control two DC motors. L298 motor driver is an integration widely used in many robotic applications and has a dual H-bridge configuration. This structure provides the possibility of controlling the speed and direction of each motor independently. The module works with a 12V DC voltage source and can provide up to 2 Amps continuous current and up to 3 Amps peak current. These features enable the control of larger and more powerful motors. These features enable the control of larger and more powerful motors. The motor driver has four input pins to which PWM signals are applied, and these signals provide speed and direction control of the motors. The output pins are designed to provide a maximum current of 2 A for each DC motor, and the motors can be easily connected via screw terminals. The wide input voltage range, high power capacity and reliable dual H-bridge configuration of L298N make this module a powerful and flexible motor driver integration,

providing an ideal solution for various robotic projects.

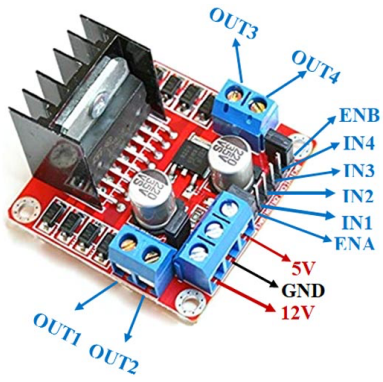

Figure 2.2 – Motor Driver (L298N)

3-) Reflectance Sensor (QTR-8RC):

QTR-8A is a type of sensor commonly used in line-following robots and has infrared contrast sensors placed at 1 cm intervals. The sensors consist of infrared light emitters, infrared light-sensitive MOSFETs and capacitors. The 8 pins corresponding to each sensor are connected to the GPIO pins of the STM32 for writing and reading operations. The MOSFETs regulate the grounding of the capacitors and are connected to ground and discharged when exposed to infrared light. The GPIO pins are briefly switched to input mode after a few milliseconds. If a particular sensor returns a logical 0, a reflective surface is present. Conversely, if the reading is a logical 1, it usually indicates a dark, non-reflective surface. Thanks to this mechanism, the microcontroller accurately determines the position of the line.

QTR-8A sensors can distinguish between black and white surfaces and can work stably and smoothly for a long time. However, it is important that the distance between the sensor and the ground does not exceed 6 mm; ideal detection distance is 3mm. Operating voltage ranges from 3.3V to 5V and draws a maximum current of 100mA. Each sensor provides an analog voltage output that varies depending on the distance to the ground or the reflectance of light. The greater the reflectance, the higher the output voltage. These features make the QTR-8A an effective and reliable sensor for line tracking applications.

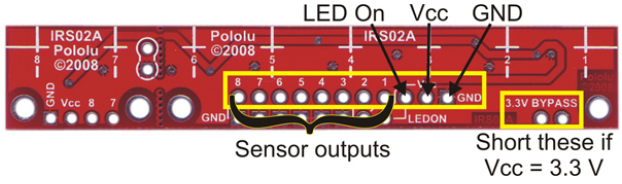

Figure 2.3 - Reflectance Sensor (QTR-8RC)

4-) Ultrasonic Distance Sensor (HC-SR04):

HC-SR04 is an ultrasonic distance sensor widely used in robotic projects to implement adaptive cruise control function. In this project, HC-SR04 ultrasonic distance sensor is used to implement adaptive cruise control function. This sensor was preferred because it can measure the distance to objects precisely and is cost-effective. HC-SR04 is an ultrasonic sensor that can measure distances between 2 cm and 400 cm and works with 3 mm sensitivity. The sensor works with 5V input voltage and uses the echo principle for its measurements. It easily communicates with the microcontroller thanks to the trig and echo pins on it. When a short pulse is sent via the trig pin, the sensor emits ultrasonic waves and the echo pin measures the time when these waves hit an object and return. This time allows the distance between the object and the sensor to be calculated. Thus, the robot will be stopped at the desired distance. These features increase the effectiveness of the sensor in the adaptive cruise control task and help the robot maintain the predetermined distance.

Figure 2.4 - Ultrasonic Distance Sensor (HC-SR04)

5-) Batteries:

Three rechargeable 3.7 V Li-ion batteries were used to power the robot system. These batteries collectively provide an input voltage of 11.7 V DC, which is connected to the 12 V input pin of the motor driver. It was determined that a minimum input voltage of 7 V was required to ensure that the motor driver's internal regulator could provide a stable 5 V output. This regulated 5 V output is critical for the efficient operation of the STM32 microcontroller and the QTR-8A reflectance sensors. The selected voltage level ensures reliable performance for both the control system and the sensors and ensures smooth operation of the robot.

6-) Buzzer (HW -508):

Buzzer is a circuit element used to obtain sound, usually working with a voltage between 2 and 4 volts. It creates sound waves through vibration. There are two basic types: active and passive buzzer. Active buzzer works by applying only a DC voltage and automatically produces a sound at a fixed frequency thanks to its own oscillator. Passive buzzer, on the other hand, requires an external signal source and can therefore operate in a wider frequency range. This is a type of passive buzzer used in this project. It is designed to work integrated with the data received from the distance sensor and to report the distance detected by the robot audibly.

Figure 2.5 – Buzzer (HW-508)

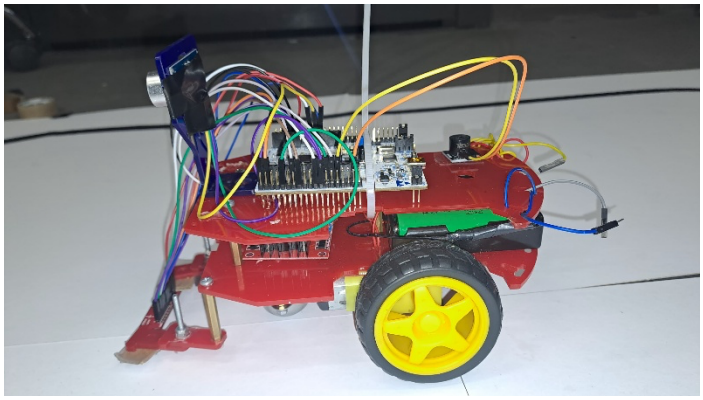

Figure 2.8 – Line Follower Robot Design

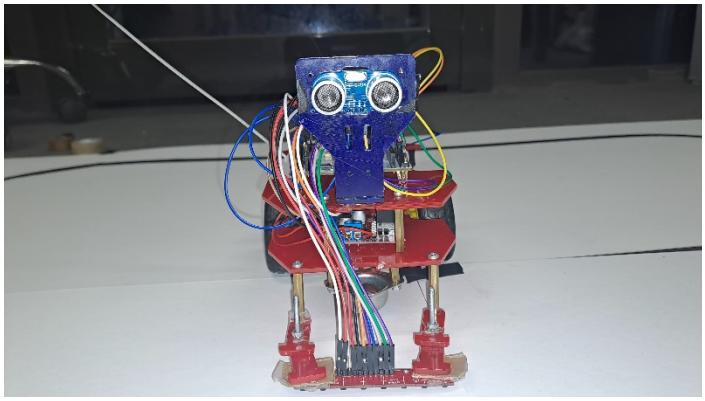

Figure 2.9 – Line Follower Robot Design

7-) Block, Circuit Diagram and Line Follower Design

The complete circuit diagram of the line follower robot is shown in the figure below. When the diagram is examined, the connections between the motor driver (L298N), microcontroller (STM32F103RB) and sensors (QTR-8A and HC-SR04) are clearly shown. The motor driver (L298N) is connected to the DC motors for driving and direction control, while the STM32F103RB microcontroller handles the sensor readings and motor control.

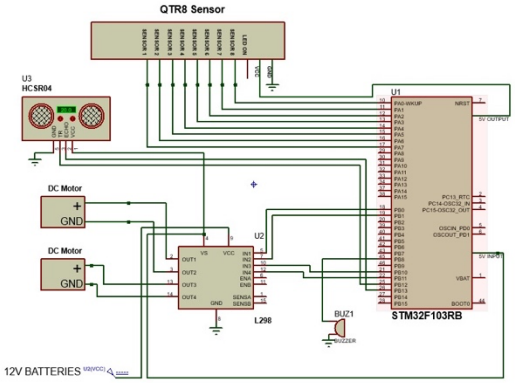

Figure 2.7 – Closed-Loop Block Diagram of the System

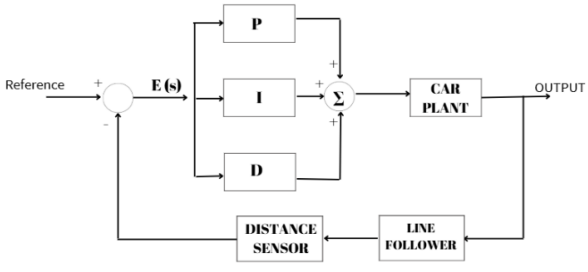

Figure 2.6 (Circuit Diagram on Proteus)

III) PID CONTROLLER DESIGN

A PID controller (Proportional-Integral-Derivative Controller) is a feedback-based control loop mechanism commonly used to manage machines and processes that require continuous control and automatic tuning. It continuously compares the system output to a reference value and calculates the difference, which is used to determine the corrective input required to bring the system closer to the desired target state. The basic components of a PID controller consist of three separate control terms: Proportional (P), Integral (I), and Derivative (D). The proportional term responds to the current error value by producing an output directly proportional to the magnitude of the error. It controls the sensitivity of the system and provides instantaneous corrections. The integral term eliminates persistent inconsistencies by accumulating past errors to eliminate persistent steady-state errors that persist over time. The derivative term predicts future errors based on the rate of change, reducing overshoot and improving system stability. The combined effect of these three terms determines the output of the PID controller. Proper adjustment of the PID constants (Kp, Ki, and Kd) is essential to achieve the desired system performance. These constants are usually set according to the specific

characteristics of the system and the required performance criteria. The tuning process usually involves

a trial-and-error approach, starting with small values and gradually adjusting until optimum control is achieved.

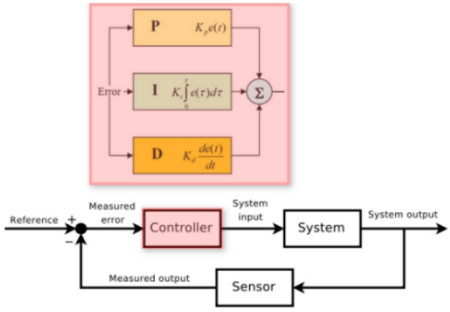

Figure 3.1– General Structure of PID Controller

The basic structure of PID control is shown above. The control action is produced by the parallel operation of proportional, differential and integrator on the error signal. The three responses are combined to obtain the controller output. The controller output is used as an input to the system you want to control and changes some aspect of the system. The controller tries to minimize the error over time by adjusting the control variable.

The mathematical form of the PID control action is given as:

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt}\, e(t)$$

K_p= Proportional gain

K_i = Integral gain

K_p = Derivative gain

e: Error=Set point –Process Value

A. Calculation of the Positional Error

The positional error is determined by analysing the data obtained from the sensors. The centre of the sensor is set to a value of 4500, as it corresponds to the desired tracking of two centrally located sensors, determining the reference position value

The registration of the sensors provides information calculated using the mathematical expression 1, and the expression 2 is used to determine the error based on the values provided by the sensors.

$$x_{position} = \frac{\sum_{n=1}^{9} 1000 * S_n}{\sum_{n=1}^{9} S_n} \quad (1)$$

$$e = 4500 - \frac{\sum_{n=1}^{9} 1000 * S_n}{\sum_{n=1}^{9} S_n} \quad (2)$$

The position is calculated by multiplying the digital outputs of the sensors by 100, obtaining that the average of sensors 4 and 5 is 4500.

A negative error indicates that the robot is deviating to the left, while a positive error indicates a deviation to the right. The trajectory is corrected by the PID controller by adjusting the speeds of the motors, modifying the duty cycle of the PWM signal sent to the motors: if the error is negative, it reduces the speed of the left motor and increases the speed of the right motor; if the error is positive, the opposite happens.

$$PID = K_p\, e(n) + K_i \sum_{n=1}^{9} e(n) + K_d[e(n) - e(n-1)$$

(The modified trajectory calculated by the PID controller)

The possible sensor conditions and the corresponding actions of the robot are shown below. When the centre sensors are enabled (A), the robot moves forward in a straight line. If the right side sensors are activated (F,G,H,I), the robot corrects by turning to the left, adjusting the motor speeds. Similarly, if the left lateral sensors (B,C,D,E) are activated, the robot corrects by turning to the right.

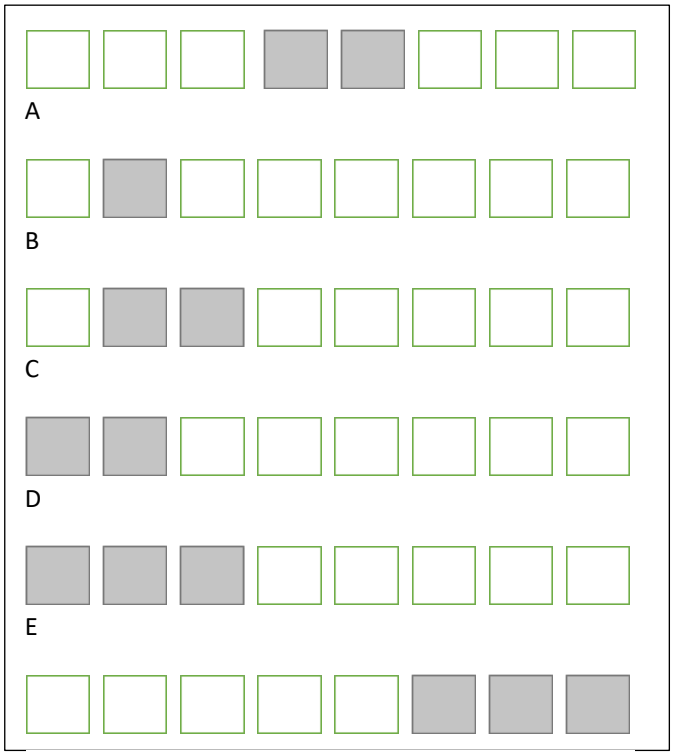

Figure 3.2 - (Sensors State Diagram)

B. Proportional Term (K_p)

The proportional term, called P, is one of the 3 components of the PID control system. Its main function is to reduce the error towards zero. The calculation of this term is done by multiplying the error by a proportional gain (Kp), the decisive setting for the response of the controller.

A high proportional gain produces instability in the system while also generating a higher controller output for a given error. Conversely, a low gain minimises oscillations, resulting in a slower error response. Therefore, selecting an appropriate value of Kp as a function of the system is crucial.

The proportional term is responsible for correcting the system output when the error is significant. However, its influence decreases as the error approaches zero.

The mathematical formulation of proportional control is shown in equation 4.

$$u(t) = K_p \qquad (4)$$

C. Derivative Term (K_d)

The derivative term is intended to provide greater stability and to anticipate possible future errors. For this purpose, the rate of variation of the error is multiplied by a constant called the derivative gain (Kd).

The derivative gain is an adjustable parameter that, as it increases, generates a pronounced control response to changes in the error rate, producing a faster response to changes. However, very high values could lead to instability and overshoot, which requires careful selection, as the derivative term is sensitive to noise, which could lead to erroneous controller outputs.

Conceptually, the derivative term is calculated as a function of the rate of change of the error, improving system stability and anticipating future errors by allowing the controller to predict and deal with errors before their occurrence.

Equation 5 illustrates the mathematical expression for derivative control.

$$u(t) = K_d \frac{de(t)}{dt} \qquad (5)$$

D. Integral Term K_i

The integral term in a control system exerts a crucial influence in eliminating steady state errors by progressively driving the error towards zero over time. Multiplying the accumulated error by a constant called the integral gain (Ki) is calculated.

The integral gain is a parameter in charge of controlling the speed with which the accumulated error of the system is corrected. A higher integral gain results in a faster control output for a given accumulated error, leading to a faster reduction of the error. However, the use of a higher integral gain can also lead to e instability, so it is essential to select a proper value of the system.

In essence, the integral term serves as a process to continuously add up the error accumulated over time by gradually eliminating the steady-state errors until zero value is reached. For this purpose, it is of primary importance to reset the integral term at the specified time, otherwise instability will occur.

The equation representing the integral term is responsible for accumulating the error over time by multiplying it by the integral gain Ki, correcting the persistent errors and reducing the steady-state error, and is represented below.

$$u(t) = K_i \int e(\tau) d(\tau) \qquad (6)$$

As a consequence during the design stage, the values of Kp, Ki and Kd were set to 0.015, 0.003 and 10, respectively, through a trial and error process. It was detected that a very low value of Kd causes oscillations in the robot's movements. Also, Kp was adjusted to ensure stable and smooth movement, while Ki was set to improve line tracking and minimise steady-state error.

IV. RESULTS AND DISCUSSION

This project aims to design and implement a line follower and adaptive cruise control robot. First of all, the main task of the robot is to follow a predefined path and complete the course by maintaining a certain distance from the obstacles that may come in front of it. To provide this functionality, the control algorithm of the robot plays a critical role in ensuring that the robot follows the line without errors, maintains the distance and exhibits stable performance.

In the software part of this system, all environmental configuration is done at the record level. This code represents the microcontroller-based software of a line follower and adaptive cruise control robot using PID control. When the written code is briefly explained, the main code is responsible for the PID algorithm to perform two basic functions of the robot. First, it ensures that the robot goes on the recognized line by reading the data from the sensor correctly and adjusts the motor speed using PID. Second, it processes the distance data received from the HC-SR04 ultrasonic sensor with these set conditions and controls the distance to a moving surface with PID. This code organizes sensor data reading, PID calculations and motor control to provide real-time adjustments to adjust line and distance tracking. Input data from QTR-8RC sensor is read via GPIOA pins and the robot's position is dynamically calculated. For this calculation, it provides a binary signal (1 or 0) that is processed to calculate the weighted average of active sensors. Based on the calculated error, PID algorithm is to make it follow a predefined line and control two DC motors for movement. At the same time, the distance of the robot to a moving surface is continuously measured with the HC-SR04 sensor and this distance is maintained with the help of PID algorithms. The sensor works with a 10 microsecond trigger signal sent from the TRIG pin. This signal is sent to surrounding objects and the sound waves reflected from the object are detected via the ECHO pin. The robot calculates the distance by measuring the duration of this reflection.

To facilitate accurate sensor readings and timing operations, the delay functions generate PWM signals using Timer2 and Timer3 configured for microsecond precision; These signals are used to generate PWM outputs on the left motor Timer2 (PB0, PB1) and on the right motor Timer3 (PB10, PB11). The generated PWM signals determine the duty cycle for each motor, which directly controls their speed. The PID algorithm processes the deviation values from the sensors for line tracking and optimizes the motor speeds based on proportional, integral and derivative components. Similarly, Timer2 is used for the measurements of the ultrasonic sensor in the distance control, and delays with microsecond precision are provided. The measured distance is compared with the target distance, and the error value is calculated. This error value is processed by the PID algorithm, and the forward speeds of the motors are adjusted to reduce this error. The system clock is configured for microsecond precision based on the 72 MHz system clock. It ensures efficient processing and synchronization of all components. In addition, a buzzer is integrated in the code and the logic controlling this pin gives an audible warning to the user when the distance limits are exceeded.

The code has a modular structure and seamlessly integrates sensor reading, PID control, motor control and feedback mechanisms, enabling the robot to adapt to dynamic conditions. This structure ensures that the robot operates precisely, stably and efficiently.

```
/* --- PWM INIT ON TIMER2 (PB0=CH3, PB1=CH4) --- */
void initPWM_Timer2(void)
{
  RCC->APB1ENR |= (1<<0);        // Enable TIM2 clock
  AFIO->MAPR   |= (3<<8);        // Full remap of TIM2 => PB0, PB1 => CH3, CH4
  TIM2->PSC     = 72 - 1;        // Prescaler => 72 => Timer freq=1 MHz
  TIM2->ARR     = 50000;         // Period => 50000 => ~20 Hz (example)

  // CH3 => OC3M=110 (PWM1 mode), OC3PE=1
  TIM2->CCMR2 |= (6<<4) | (1<<3);
  // CH4 => OC4M=110, OC4PE=1
  TIM2->CCMR2 |= (6<<12) | (1<<11);
  // Enable CH3, CH4 outputs
  TIM2->CCER  |= (1<<8) | (1<<12);
  // ARPE=1
  TIM2->CR1   |= (1<<7);
  // Enable TIM2
  TIM2->CR1   |= (1<<0);
}

/* --- PWM INIT ON TIMER3 (PB10=CH3, PB11=CH4) --- */
void initPWM_Timer3(void)
{
  RCC->APB1ENR |= (1<<1);      // Enable TIM3 clock
  TIM3->PSC     = 72 - 1;      // Prescaler => 72 => 1 MHz
  TIM3->ARR     = 2000;        // Period => 2000 => 500 Hz (example)

  // CH3 => OC3M=110 (PWM1), OC3PE=1
  TIM3->CCMR2  |= (6<<4) | (1<<3);
  // CH4 => OC4M=110 (PWM1), OC4PE=1
  TIM3->CCMR2  |= (6<<12) | (1<<11);
  // Enable CH3, CH4
  TIM3->CCER   |= (1<<8) | (1<<12);
  // ARPE=1
  TIM3->CR1    |= (1<<7);
  // Enable TIM3
  TIM3->CR1    |= (1<<0);
}
```

Figure 4.1 (Motor Speed Control)

```
/* --- MEASURE DISTANCE (HC-SR04) --- */
float measureDistance(void)
{
  GPIOB->ODR &= ~(1<<12);   // TRIG=0
  delay_us(2);              // Wait 2 us
  GPIOB->ODR |= (1<<12);    // TRIG=1
  delay_us(10);             // High for 10 us
  GPIOB->ODR &= ~(1<<12);   // TRIG=0

  while((GPIOB->IDR & (1<<13)) == 0);  // Wait ECHO=1
  uint32_t start = TIM2->CNT;          // Record start
  while((GPIOB->IDR & (1<<13)) != 0);  // Wait ECHO=0
  uint32_t end   = TIM2->CNT;          // Record end

  uint32_t pulseWidth = (end > start) ? (end - start) : 0;
  float dist = (float)pulseWidth / 58.0f; // approximate => 1us ~ 58 => 1 cm
  return dist;
}
/* --- BUZZER CONTROL --- */
void buzzerControl(float dist)
{
  // Example: beep if near setpoint or if out of range
  if(dist > 58.0f && dist < 62.0f)
  {
    GPIOB->BSRR = (1<<8);     // Buzzer ON
    delay_ms(100);            // 100 ms beep
    GPIOB->BSRR = (1<<(8+16));// Buzzer OFF
  }
  else if(dist < 40.0f || dist > 80.0f)
  {
    GPIOB->BSRR = (1<<8);     // Continuous beep ON
  }
  else
  {
    GPIOB->BSRR = (1<<(8+16));// Buzzer OFF
  }
}
```

Figure 4.2 (Buzzer Control)

After completing all the necessary configurations, the line follower robot was ready to navigate along the track. It can be found the codes of line follower with detailed explanation and demonstration video in the following links.

Link for the codes:

- https://drive.google.com/drive/folders/1o6B_dDebFYhTOptOExP8KIK0iiSkXDdT?usp=sharing
- https://github.com/alasulu/STM32_LINE_FOLLOWER

Link for the video:

- https://youtu.be/eMMg-Zr_-wU?si=v-y6RqtGeFxpkQkk

There were some difficulties and situations encountered while completing this project. It was difficult to obtain precise readings from QTR-8RC reflection sensors and HC-SR04 ultrasonic sensor. Due to the lack of materials and the inefficient operation of the sensor and materials used, the robot's distance and line tracking perception was difficult. Some negative situations occurred in some cases while testing the robot. Tests conducted in different environments, irregular surfaces, changing friction levels and external factors such as ambient noise negatively affected the performance of the ultrasonic sensor. It was observed that dust and dirt on the test track and wheels negatively affected the stability. When the track was dirty, it was observed that the robot sometimes slipped off the track and could not complete. While testing the robot, it was observed that changes in the battery level affected the speed of the motor and sensor readings from time to time as a result of long-term trials. However, addressing and solving these problems increased the robustness, adaptability and overall performance of the robot on the server and ensured a successful result.

V. CONCLUSION

This project successfully demonstrates the construction of a line following and adaptive cruise control robot using PID control algorithms. By integrating QTR-8RC reflection sensors for line following and HC-SR04 ultrasonic sensor for distance measurement, the robot successfully completed predefined paths while maintaining a safe distance from obstacles. In addition, a buzzer was added to make the system more adaptable to real world conditions, providing audible feedback when distance thresholds are exceeded. All the code written in this project did not use any library, all codes were written at the register level. In addition, PID controllers were successfully integrated into embedded systems as an important software element. Multiple problems were encountered during the completion of the project. Therefore, the Ssensor data was dynamically processed to calculate position and distance errors, which were then corrected in real time using PID algorithms. The line and distance were effectively maintained by the proportional, integral, and derivative components by minimizing mistakes in spite of challenges such abrupt turns, shifting surfaces, and environmental changes. As a result, the project met its objectives and showed how modular, scalable robotic systems might be used in practical settings. In summary, the robot successfully met all the desired conditions, followed the specified line as required and successfully maintained the specified 60 cm distance from the moving board. In addition, the added buzzer provided real-time feedback, allowing the system to be warned audibly in undesirable situations.

REFERENCES:

[1] Toroman, A., & Vojic, S. (2021, November). *Adaptive car control system based on a predictive model*. **IOP Conference Series: Materials Science and Engineering**, 1208(1), 012040. https://doi.org/10.1088/1757-899X/1208/1/012040

[2] Abidoye, O. M., Danjuma, U. A., Jibrin, M. S., Mshelia, M. B., Abubakar, M. A., & Suleiman, U. Y. (2015).*A mechatronics design of a line tracker robot using Ziegler Nichols control technique for P, PI and PID controllers*. 10.13140/RG.2.1.4107.4722

[3] Mahfouz, A. A., Aly, A. A., & Salem, F. A. (2013). *Mechatronics design of a mobile robot system*. **International Journal of Intelligent Systems and Applications**, 5(3), 23–36.

https://doi.org/10.5815/ijisa.2013.03.03

[4] Parikh, P. A., Shah, H., & Sheth, S. M. (2014, June). *A mechatronics design of a line tracker robot using Ziegler Nichols control technique for P, PI and PID controllers*.
In *Proceedings of the International Mechanical Engineering Congress (IMEC - 2014)* (Vol. 1, pp. 529–532).https://doi.org/10.13140/RG.2.1.4107.4722

[5]https://en.wikipedia.org/wiki/Proportional%E2%80%93integral%E2%80%93derivative_controller

[6] Pakdaman, M., Sanaatiyan, M. M., & Rezaei, M. (2010, March).*A line follower robot from design to implementation: Technical issues and problems*.In *2010 2nd International Conference on Computer and Automation Engineering (ICCAE)* (Vol. 1). IEEE. https://doi.org/10.1109/ICCAE.2010.5451881

[7] Taştan, Ö. (2017, December). *Arduino Line Follower Robot*. Bayburt University.