KOCAELİ UNIVERSITY
FACULTY OF ENGINEERING
ELECTRONIC AND TELECOMMUNICATION ENGINEERING

CAR WASHING STATION WITH C#

C# PROJECT

Faculty Member:
Dr. Prof. OĞUZHAN KARAHAN

KOCAELİ, 2023

FOREWORD and ACKNOWLEDGMENTS

Thanks to technology getting better, we've seen big improvements in how computer programs and tiny computing devices (microcontrollers) work together. Programming languages and microcontrollers are super important for making all sorts of cool tech things.

This project entails the creation of an automated car washing station, where the various operational stages are controlled by microcontrollers. The overall orchestration and user interaction are facilitated through a user-friendly C# graphical user interface (GUI). Three distinct microcontroller-driven stages, including water spraying, and drying, are seamlessly coordinated through the centralized control provided by the C# UI. This integrated system aims to optimize the car washing process, enhance user experience, and demonstrate the efficiency of microcontroller-C# synergy in managing diverse tasks within the station.

Appreciation is extended to the esteemed advisor, Dr. Professor OĞUZHAN KARAHAN , whose consistant support and encouragement have been instrumental throughout the various stages of this project.

December 2023, KOCAELİ Furkan ÖZKEROĞLU

CONTENTS

ŞEKİLLER DİZİNİ

ABOUT PROJECT

Furkan ÖZKEROĞLU

Keywords: user interface.

Abstract: In the world of advancing technology, practical projects are popping up to tackle everyday challenges. This project focuses on creating a four-stage automated car washing station. The main goal is to give users control through a simple user interface (UI) made with Visual Studio C#. This UI connects with a microcontroller that oversees how the station works.

Project Overview:

The car washing station has four stages, and users can set up each stage using the easy-to-use UI made with Visual Studio C#. The microcontroller takes care of the nitty-gritty details, with managing sensors runs smoothly.

The idea is to provide a straightforward, tech-powered solution, letting users customize their car washing experience. This project shows how technology, like a microcontroller paired with a C# UI, can make an automated car washing station that's easy for anyone to use.

1. INTRODUCTION

C#:

C# (C-sharp) is a modern, object-oriented programming language developed by Microsoft. It is part of the .NET framework and is commonly used for developing Windows applications, web applications, and services. C# is known for its simplicity, type-safety, and integration with Microsoft technologies.alınabilir.

Python:

Python is a high-level, interpreted programming language that is widely used for its readability and ease of use. It supports multiple paradigms, including procedural, object-oriented, and functional programming. Python is popular for web development, data science, artificial intelligence, and automation. It has a vast ecosystem of libraries and frameworks, making it a versatile choice for various applications.

Raspberry Pi (RPI):

Raspberry Pi is a series of small, affordable, single-board computers that have gained popularity for various DIY (do it yourself) projects. It runs on different operating systems, including Linux, and is known for its versatility and compact size. Raspberry Pi is widely used in projects ranging from home automation to educational purposes.

2. PRESENTATION OF ELECTRONIC MATERIALS USED

2.1 Raspberry Pi 4 8gb

The Raspberry Pi 4 is the 4th generation of the mainline series of Raspberry Pi single-board computers. Developed by Raspberry Pi Trading and released on 24 June 2019, the Pi 4 came with many improvements over its predecessor; the SoC was upgraded to the Broadcom BCM2711, two of the Raspberry Pi's four USB ports were upgraded to USB 3.0, and options were added for RAM capacities larger than the 1 GB standard for the preceding Raspberry Pi 3 series.[1] The Pi 4 also ends the trend of the $35 maximum MSRP that previous Raspberry Pis had adhered to, as the larger RAM capacities added extra cost to the board; however, the base 1 GB model is still sold for $35. On 28 September 2023, the Raspberry Pi 5 was announced as the successor to the Raspberry Pi 4.[2]

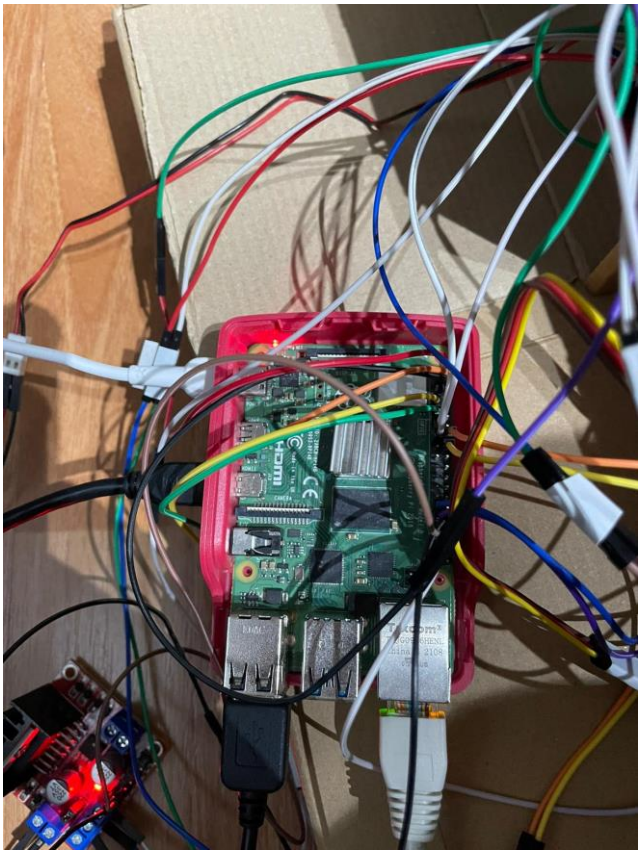

Figure 2. 1: Raspberry PI 8 gb

2.2 DISTANCE SENSOR

The HC-SR04 is an affordable and easy to use distance measuring sensor which has a range from 2cm to 400cm (about an inch to 13 feet). The sensor is composed of two ultrasonic transducers. One is transmitter which outputs ultrasonic sound pulses and the other is receiver which listens for reflected waves.

Figure 2. 2: HC-SR04 Ultrasonic Sensor

2.3 SG5010 (SERVO MOTOR):

Servo motors or “servos”, as they are known, are electronic devices and rotary or linear actuators that rotate and push parts of a machine with precision. Servos are mainly used on angular or linear position and for specific velocity, and acceleration.

Figure 2. 3: SG5010 (SERVO MOTOR)

2.4 12V fan

Figure 2. 4: 12V fan

2.5 L298N motor driver

The L298N is a dual H-Bridge motor driver which allows speed and direction control of two DC motors at the same time. The module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A.

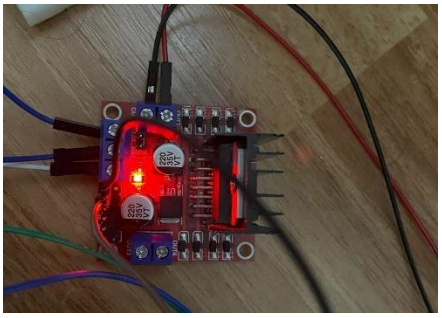

Figure 2. 5: L298N motor driver

3 PROJECT OVERVIEW AND HIERARCHICAL LEVELS

3.1 Introduction

In the realm of automated car wash systems, the integration of advanced Project technologies has become paramount to enhance user experience and operational efficiency. This project aims to develop a sophisticated user interface (UI) using Visual Studio C# to interact seamlessly with a Raspberry Pi 4 (8GB) microcontroller. The UI will guide users through various stages of the car wash process, providing real-time alerts and customization options.

3.2 Explanation

As the vehicle approaches the barrier of the automated car wash, the user interface prominently displays pertinent information, enabling the user to make decisions regarding the speed at which the entrance barrier opens. Subsequently, the interface facilitates a seamless transition to the washing area. Upon reaching the Water Spray Section, the user interface apprises the user of the available options for water drainage, affording choices such as 30%, 60%, or 100% drainage from the tank. This interactive feature empowers users to exert control over water consumption based on their individual preferences. Furthermore, the intelligent interface issues a notification in the event of the presence of another vehicle within the machine, signaling the arrival of a new vehicle and prompting appropriate alerts. During the transition from the washing area to the drying zone, the smart interface prompts the user to specify the desired fan speed for drying. The user can tailor the fan speed according to personal preference. Upon cessation of the fan, the machine permits the user to exit, ensuring a user-centric and customizable car wash experience..

3.3 Conclusion

This comprehensive automation system, integrating Visual Studio C# UI with the Raspberry Pi 4 microcontroller, not only streamlines the car wash process but also enhances user engagement and control. The project underscores the potential of

marrying software and hardware to create a user-friendly, interactive, and efficient car wash experience. As technology continues to advance, such integrations pave the way for innovative solutions in various domains, setting new standards for automation and user interface design.

3.4 Unveiling the Structure and Components of a Python Implementation

3.4.1 Servos

In this study, we present an integrated servo-control system designed for smart station gate and water tank management. The system incorporates two servo motors, each serving a distinct purpose. The first servo motor is dedicated to controlling the station gate, allowing users to specify opening and closing speeds based on the information provided. Through user-friendly interfaces, the system dynamically adjusts the gate's movements, providing flexibility and customization for different scenarios. The second servo motor focuses on regulating the rotation of the water tank. Users input information to determine the desired rotation angle, influencing the water dispensation process accordingly. This integrated approach facilitates a seamless user experience, where both the station gate and water tank movements are tailored to user preferences. Figure 3.1 illustrates the dynamic function representing the speed of the station gate, allowing users to customize opening and closing velocities. In Figure 3.2, the wash tank function is presented, showcasing the servo-controlled rotation based on user-defined input.

```
def doorSpeed(speed_level):
    try:
            if speed_level == "Slow":
                sleep_duration = 0.02
            elif speed_level == "Normal":
                sleep_duration = 0.01
            elif speed_level == "Fast":
                sleep_duration = 0.005
            else:
                print("Geçersiz hız seviyesi!")
                return

            # Hareket (0 ile 180 derece arasında)
            for i in range(90, 270, 1):
                aciAyarla(i)
                time.sleep(sleep_duration)

    except KeyboardInterrupt:
        # İstisna durumunda temizlik işlemlerini gerçekleştirin
        GPIO.cleanup()
        pwm.stop()
```

Figure 3.1: DoorSpeed

```
def washTank(angle):
    try:
        if angle == "%30":
            for i in range(2, 62, 1):
                aci2Ayarla(i)
                time.sleep(0.02)
        elif angle == "%60":
            for i in range(2, 102, 1):
                aci2Ayarla(i)
                time.sleep(0.02)
        elif angle == "%100":
            for i in range(2, 182, 1):
                aci2Ayarla(i)
                time.sleep(0.02)
        else:
            print("Geçersiz açı!")
            return

    except KeyboardInterrupt:

        GPIO.cleanup()
        pwm2.stop()
```

Figure 3.2: WashTank

3.4.2 Ultrasonic Distance Sensor

In this study, three distinct ultrasonic distance sensors are employed to enhance the adaptability of smart station components. The first sensor caters to approaching vehicles at the station gate, while the second sensor is dedicated to vehicles nearing the washing station. The third sensor focuses on vehicles approaching the drying station. These sensors play a pivotal role in creating adaptive functions, as illustrated in Figure 3.3. Each function corresponds to specific station components, allowing for dynamic responses to varying proximity scenarios. This research underscores the importance of sensor-driven adaptability in optimizing the performance of smart station features, contributing to the evolution of efficient and responsive automated systems.

```
def distance():
    GPIO.output(TRIG, False)
    time.sleep(0.1)

    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    pulse_start = time.time()
    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()

    pulse_end = time.time()
    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150
    distance = round(distance, 2)
    return distance
```

Figure 3.3: DistanceFunc

3.4.3 Fan

This study incorporates a 12-volt fan with user-configurable speed control to optimize ventilation within the smart station. Figure 3.4 demonstrates the dynamic function of the fan, enabling users to select from slow, medium, and fast speed settings. This customizable fan speed feature enhances user comfort and contributes to energy efficiency. The research underscores the significance of user-driven control in tailoring environmental conditions to individual preferences, showcasing the potential of smart systems to

prioritize user experience and sustainability.

```
elif message == "Low":
    pwm1.ChangeDutyCycle(30)
elif message == "Medium":
    pwm1.ChangeDutyCycle(65)
elif message == "High":
    pwm1.ChangeDutyCycle(100)
elif message == "CloseCooling":
    pwm1.ChangeDutyCycle(0)
```

Figure 3.4: Fan

3.4.4 From C# to RPI

This study focuses on the seamless integration of a Python codebase running on a Raspberry Pi (RPI) with a C# application through TCP communication. The Python code, responsible for controlling various components of a smart station, communicates with the C# application to enable real-time monitoring and user interaction. The TCP communication establishes a robust connection between the RPI and the C# interface, facilitating efficient data exchange and command execution. This research highlights the effectiveness of cross-language communication for enhanced control and monitoring capabilities in smart systems.Figure 3.5.

```
if __name__ == "__main__":
    tcp_receive_thread = threading.Thread(target=tcp_receive_thread)
    main_thread = threading.Thread(target=main_thread)

    # İş parçacıklarını başlat
    tcp_receive_thread.start()
    #udp_receive_thread.start()
    main_thread.start()

    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        GPIO.cleanup()
        pwm.stop()
        udp_receive_thread.join()
        main_thread.join()
```

```python
def tcp_receive_thread():
    server2_ip = "192.168.0.59"
    server2_port = 5555

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((server2_ip, server2_port))
    server_socket.listen(1)  # 1 bağlantıyı dinle

    print(f"TCP Sunucu dinleniyor: {server2_ip}:{server2_port}")

    try:
        while True:
            conn, addr = server_socket.accept()
            print(f"Bağlantı alındı from {addr[0]}:{addr[1]}")

            data = conn.recv(1024)
            message = data.decode("utf-8")
            print(f"Gelen mesaj: {message}")
```

Figure 3.5: Tcp

3.4.5 From RPI to C#

This study delves into the integration of a Raspberry Pi (RPI) and a C# application through UDP communication, presenting an efficient means of inter-device interaction in a smart system context. The Python code running on the RPI communicates seamlessly with the C# application, enabling real-time data exchange and control commands. The implementation of UDP ensures a reliable and lightweight communication protocol for effective coordination between the devices. This research showcases the advantages of utilizing UDP communication to enhance connectivity and collaboration in the development of smart systems.Figure 3.6 Udp

```python
def send_udp_message(message, server_ip="192.168.1.2", server_port=5555):
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    client_socket.sendto(message.encode("utf-8"), (server_ip, server_port))
    client_socket.close()
```

Figure 3.6: Udp

3.5 Communication Technology Used

3.5.1 TCP (Transmission Control Protocol):

The Transmission Control Protocol (TCP) is a reliable, connection-oriented communication protocol. TCP divides data into packets and establishes a connection to securely deliver these packets to the recipient. Upon arrival, the recipient ensures that these packets are received in the correct order and without any omissions. TCP incorporates a series of mechanisms to ensure reliability by managing packet loss, duplicate transmissions, and irregular

developments. Due to these characteristics, TCP is commonly utilized in applications such as file transfers and web page displays where data integrity and sequencing are crucial.

3.5.2 UDP (User Datagram Protocol):

The User Datagram Protocol (UDP) is an unreliable, connectionless, and unordered communication protocol. UDP breaks down data into packets and directly delivers them to the recipient without establishing a connection. However, UDP lacks mechanisms for ensuring reliability or preserving packet sequence. Consequently, UDP is preferred in applications where swift and low-latency communication is imperative. Applications such as real-time voice transmission, video streaming, and online gaming often leverage UDP due to the critical importance of speed and instantaneous response, even at the expense of guaranteed reliability.

3.6 C# Code

3.6.1 UI

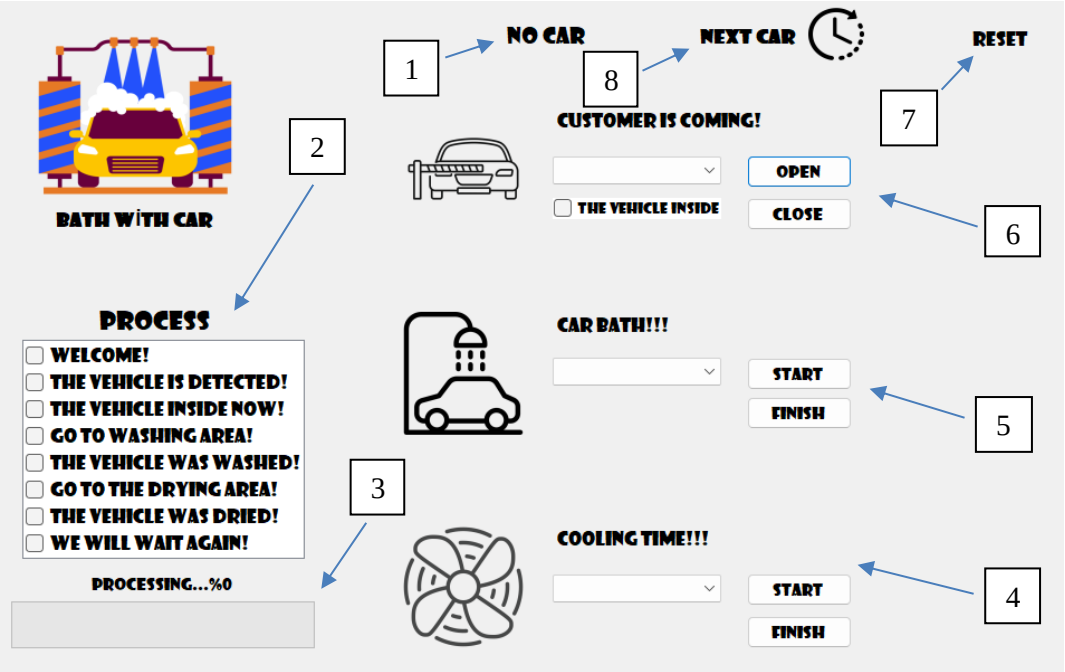

Figure 3.7: UI

3.6.1.1 “NO CAR”

In the realm of automated car wash systems, the initiation point is marked by a pivotal element known as "No Car." This distinctive feature serves as the preliminary status indicator, discerning the presence or absence of an automobile within the machine. The system's user interface deftly responds to the absence of a car, providing a clear and unequivocal declaration through the designation "No Car." This signal not only signifies the absence of a vehicle but also serves as a trigger for a visual transformation, wherein the color representation dynamically adjusts to underscore the absence of an automobile. The nuanced color change acts as a visual cue, efficiently communicating the system's status and facilitating a seamless user experience. This intricate interplay between the "No Car" status and the corresponding color modulation embodies a foundational aspect of our automated car wash machine, contributing to its user-centric design and operational transparency.

3.6.1.2 “PROCESS Check List Box”

The integration of a sophisticated "Process Checklist Box" within our automated car wash and drying system represents a strategic enhancement geared towards optimizing user interaction and fostering operational transparency. This innovative feature, positioned prominently on the user interface, serves as an intuitive visual guide, systematically communicating the current stage of the car wash and drying process while presenting forthcoming steps in a comprehensible format. Designed as a user-friendly interface element, the checklist box succinctly encapsulates completed or pending tasks, providing users with an immediate and coherent overview of ongoing procedures. This dynamic checklist not only serves as a status indicator for the car wash and drying phases but also visually progresses to denote the advancement of each stage, assuring users of the system's current state. With its anticipatory display of upcoming steps, visual confirmation of completed processes, and clear user guidance, the Process Checklist Box exemplifies our commitment to operational clarity and user-centricity in the domain of automated vehicle maintenance, thereby

enhancing the overall car care experience.

3.6.1.3 “PROGRESS BAR”

The incorporation of a Progress Bar within our automated car wash and drying system offers users a succinct and visual representation of the ongoing process percentage. This graphical element, positioned prominently on the user interface, provides real-time insight into the current status of the car wash and drying sequences, enhancing operational transparency and user experience.

3.6.1.4 “COOLING AREA”

Upon the entrance of the vehicle into the designated drying section of the system, a noteworthy visual cue manifests on the interface. The drying indicator undergoes a distinct color change and initiates a conspicuous flashing pattern, providing a clear and immediate signal to the user regarding the commencement of the drying phase. Subsequently, users are afforded the flexibility to engage with the system through the List Box interface, enabling them to select a preferred drying mode. Upon making their selection, users can initiate the drying process by activating the Start button, thus setting the operation in motion. Crucially, the control to cease the operation is tactfully vested in the hands of the user, reinforcing a user-centric approach to the automated car wash and drying system. This design not only enhances operational transparency but also underscores our commitment to empowering users with control over their experience.

3.6.1.5 “WASHING AREA”

Upon the vehicle's arrival in this designated section of the system, a seamless detection process ensues, triggering a notification to be seamlessly transmitted to the interface. In response, the title on the interface undergoes a conspicuous color alteration and initiates a flashing sequence, serving as an unambiguous signal to the user regarding the system's awareness of the vehicle's presence. Empowering users with further customization, the List Box interface facilitates the user's input of their preferred water level. Upon the user's activation of the Start button, the system adeptly dispenses the

designated volume of water onto the vehicle, aligning with the user's specified preferences. The culmination of the process is initiated by the user's interaction with the Finish button, prompting the water tank to seamlessly revert to its original position, thus concluding the water dispensing operation.

3.6.1.6 “BARRIER AREA”

Upon the approach of a vehicle, a dynamic response unfolds within the system, signified by a change in the title color accompanied by a pulsating pattern, providing a vivid indication to the user of the system's awareness of the approaching vehicle. Following this, the user is presented with a set of three distinct barrier options, each offering varied operational parameters. The user, empowered with the freedom to select an option, triggers the opening of the barrier. Upon successful entry, the system reaffirms the user's passage by signaling a title color change. This alteration serves as a visual cue to the user, prompting them with the knowledge that the entry has been registered. Consequently, the user gains the agency to close the barrier at their discretion, thereby ensuring a seamless and user-controlled experience within the automated car wash system.

3.6.1.7 “RESET”

The Reset button within the system serves as a pivotal tool, providing assistance in the process of resetting and restoring default parameters. This feature, strategically integrated into the interface, enables users to swiftly and efficiently revert various settings and configurations to their initial states. The Reset button acts as a safeguard, affording users the convenience of rectifying unintended changes or modifications.

3.6.1.8 “NEXT CAR”

The dynamic feature embedded within the system dictates that, when a new vehicle arrives at the starting point while another is already within the system, the "Next Car" text undergoes an instantaneous transformation. Simultaneously, the color palette associated with this text undergoes a discernible alteration, providing an immediate and

visually distinctive cue to signify the introduction of a new vehicle into the automated system. This visual and informational change serves as a responsive element, ensuring that users and system operators are promptly notified of the presence of an additional vehicle, thereby facilitating a seamless transition between successive vehicle processes.

3.6.2 Functions

3.6.2.1 “From RPI to C#”

This C# code segment establishes a UDP listener for an automated car wash system, listening on port 5555. The “UdpListenerThread” continuously monitors incoming messages, decoding them to trigger specific actions in the user interface. Messages like "YesCar" initiate changes such as label color modifications, blinking control activation, progress bar updates, and “CheckedListBox” item checks. Other messages, like "PassedCar," signify a car passing a checkpoint, adjusting flags and halting blinking. Exception handling is in place to log potential errors during the UDP listening process. Overall, this code exemplifies real-time communication integration within the car wash system, orchestrating UI responses based on received messages.

```
1 başvuru
private void InitializeUdpListener()
{
    // UdpClient'ı başlat
    int serverPort = 5555;
    udpServer = new UdpClient(serverPort);

    // Arka planda dinleme işlemini başlat
    Thread udpListenerThread = new Thread(new ThreadStart(UdpListenerThread));
    udpListenerThread.Start();
}

1 başvuru
private void UdpListenerThread()
{
    try
    {
        while (true)
        {
            byte[] data = udpServer.Receive(ref clientEndpoint);
            string message = Encoding.UTF8.GetString(data);

            // Gelen mesajı işle
```

Figure 3.8: Udp for C#

3.6.2.2 “From C# to RPI”

This C# code segment establishes a TCP connection to send a selected option to a Raspberry Pi with IP address "192.168.0.59" on port 5555. It utilizes a “TcpClient” to connect to the specified IP and port,

creating a network stream for data transmission. The selected option, provided as a parameter to the method, is converted to ASCII-encoded bytes and sent over the established TCP connection. The console displays a message confirming the sent option. Exception handling is implemented to catch and log any errors that may occur during the connection process. The "tcpClient" is closed in the finally block to ensure proper resource management.

```
6 başvuru
private void SendSelectedOption(string selectedOption)
{
    TcpClient tcpClient = new TcpClient();

    try
    {
        IPAddress serverIP = IPAddress.Parse("192.168.0.59");  // Raspberry Pi'nin IP
        int port = 5555;

        tcpClient.Connect(serverIP, port);

        NetworkStream stream = tcpClient.GetStream(); // Veri gönderimi için bir akış al

        string message = selectedOption;
        byte[] data = Encoding.ASCII.GetBytes(message);

        stream.Write(data, 0, data.Length);

        Console.WriteLine("Gönderilen mesaj : {0} ", selectedOption);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.ToString());
    }
    finally
    {
        tcpClient.Close();
    }
}
```

Figure 3.9: Tcp for C#

3.6.2.3 "RESET FUNCTION"

The Clean method in the C# application systematically resets critical system parameters and visual indicators. It unchecks user selections in CheckedListBox controls, resets progress representations, and manages boolean flags (isBlinking, carInside, carCheck, carCheck2, carCheck3). Notably, carInside indicates the vehicle's presence, and carCheck tracks its processing stage within the car wash. The method ensures a neutral state for subsequent operations by uniformly adjusting visual cues and stopping ongoing alerts. This systematic reset guarantees the restoration of the system's operational state, ready for subsequent user interactions within the automated car wash environment.

```
private void Clean()
{
    if (checkedListBox1.Items.Count > 0)
    {
        checkedListBox1.SetItemChecked(0, false);
        checkedListBox1.SetItemChecked(1, false);
        checkedListBox1.SetItemChecked(2, false);
        checkedListBox1.SetItemChecked(3, false);
        checkedListBox1.SetItemChecked(4, false);
        checkedListBox1.SetItemChecked(5, false);
        checkedListBox1.SetItemChecked(6, false);
        checkedListBox1.SetItemChecked(7, false);
    }
    if (checkedListBox2.Items.Count > 0)
    {
        checkedListBox2.SetItemChecked(0, false);
    }
    label8.Text = "Processing...%0";
    progressBar1.Value = 0;

    isBlinking = false;
    carInside = false;
    carCheck = false;
    carCheck2 = false;
    carCheck3 = false;

    label3.ForeColor = System.Drawing.Color.Black;
    label5.ForeColor = System.Drawing.Color.Black;
    label6.ForeColor = System.Drawing.Color.Black;
    label2.ForeColor = System.Drawing.Color.Black;
    label4.ForeColor = System.Drawing.Color.Black;

    StopBlinking(pictureBox2);
    StopBlinking(pictureBox3);
    StopBlinking(pictureBox4);
}

1 başvuru
private void label9_Click(object sender, EventArgs e)
{
    Clean();
}
```

Figure 3.10: Reset Func

3.6.2.4 “Other Functions”

Following the completion of each phase in the automated car wash system, the associated checklist box undergoes periodic updates, reflecting the progress and status of the ongoing operation. Simultaneously, the progress bar incrementally advances, providing users with a visual depiction of the evolving processes. This continuous feedback mechanism ensures users are consistently apprised of the system's status throughout various stages, enriching the overall user experience.

In the event of potential user input errors during these stages, the system integrates robust error-handling mechanisms. Whether stemming from incomplete inputs or unexpected issues, the system employs exception handling to log and manage errors effectively, safeguarding the continuous operation of the car wash system.

Furthermore, user-selected options are seamlessly relayed to the hardware subsystem for further processing. Through a streamlined process, the chosen options are encoded and transmitted to the hardware component, facilitating their execution. By establishing this cohesive link between the user interface and the hardware layer, the

system ensures the synchronized implementation of user preferences, fortifying the reliability and responsiveness of the automated car wash system. This integrated approach, combining real-time feedback, error management, and hardware communication, underscores the system's resilience and user-centric design principles.

4 RESULTS

In conclusion, the developed car wash system successfully integrates sensors, fans, and servos controlled by a Raspberry Pi (RPi) along with a C# Windows Form application. The system effectively detects approaching vehicles, adjusts barrier speed based on user input, prevents entry when another vehicle is inside, and indicates waiting status on the user interface (UI). The process seamlessly progresses to the washing section, where water is dispensed from a tank at user-specified intensity. Subsequently, the system transitions to the drying phase, activating a fan upon detection. The entire process concludes when the drying phase completes.

The integration of hardware components and software application demonstrates a functional and user-friendly car wash system. The responsive nature of the barrier, water dispensing, and drying mechanisms ensures efficient operation. The user interface provides clear feedback to the user throughout the process, enhancing the overall user experience. The modular design allows for flexibility in system customization and scalability for potential future enhancements.

This project contributes to the field by showcasing an automated car wash system that combines the power of Raspberry Pi, sensor technology, and a user-friendly interface. The successful implementation and functionality validate the feasibility of such systems for practical applications. Future work could focus on optimizing resource usage, exploring additional features, and considering environmental sustainability in the design and operation of automated car wash systems. Overall, this project serves as a foundation for further advancements in automated vehicle care systems.

KAYNAKLAR

[1] Raspberry Pi Datasheets. 21 June 2019. Retrieved 3 October 2023.

[2] Upton, Eben (28 September 2023). "Introducing: Raspberry Pi 5!". Raspberry Pi. Retrieved 4 October 2023.).

[3] Iyanda, J. Connection-Oriented TCP vs Connectionless UDP: A Comparative Analysis of Network Protocols.

[4] KARAHAN, O., & BEDER, B. (2023). Design of Smart Bin based on C# Through Raspberry PI. *Eskişehir Türk Dünyası Uygulama ve Araştırma Merkezi Bilişim Dergisi*, *4*(1), 1-7.