

WI-FI DRIVER DEVELOPMENT



BY

Ahmed Ali

&

Seher Ishtiaq

(Reg. No. 2021-UOK-04518)

&

(Reg. No. 2021-UOK-04511)

Session 2021-2025

Department of Data Science

Faculty of Computing & Engineering

University of Kotli Azad Jammu and Kashmir

PROJECT REPORT FOR WI-FI DRIVER DEVELOPMENT

BY

Ahmed Ali

&

Seher Ishtiaq

(Reg. No. 2021-UOK-04518)

&

(Reg. No. 2021-UOK-04511)

A Report

Submitted in partial fulfillment of the requirements for the 5th semester project of

Operating System

In

Data Science

Session 2021 -25

Department of Data Science

Faculty of Computing & Engineering

University of Kotli Azad Jammu and Kashmir

Contents

1.	Introduction.....	1
1.1.	Background	1
1.2.	Objectives.....	1
1.3.	Project Scope.....	1
1.4.	Target Platform	2
2.	Methodology	2
2.1.	Overview of Development Process	2
2.2	User Mode Approach	3
2.3	Kernel Mode Approach.....	3
2.4	Tools and Technologies.....	4
2.5	Requirements Analysis.....	4
3.	Implementation	6
3.1.	Environment Setup.....	6
3.2.	Main Functions Implemented In Code	6
3.3.	Challenges Faced.....	7
3.4.	Testing Methodologies	8
3.5	Testing On Target System.....	9
3.6	Target System Deployment	10
3.7	Host System Configuration	11
4.	Deployment.....	12
5.	Conclusion and Reference	16
5.1.	Lessons Learned.....	16
5.2.	References	16

1. Introduction

Ever rage-quit a video call because your Wi-Fi decided to take a vacation? Yeah, us too. That's why we build a super-charged Wi-Fi driver for Windows 10. This isn't your average driver, it's about to become your new best friend!

Imagine browsing the web without those annoying buffering circles, streaming movies without any lag, and working on cloud documents with zero hiccups. That's the kind of smooth, reliable connection we're aiming for. Our goal is to make sure this driver works perfectly with all sorts of devices and gives everyone on Windows 10 the best possible Wi-Fi experience. Basically, we're saying goodbye to Wi-Fi woes and hello to a frustration-free, connected life!

1.1. Background

With the increasing reliance on wireless networks for both personal and professional activities, the demand for optimized Wi-Fi drivers has grown significantly. Traditional Wi-Fi drivers often face challenges such as connectivity issues, compatibility issues with different hardware configurations, and suboptimal performance. These issues can lead to frustrating user experiences, including slow internet speeds, dropped connections, and limited range.

1.2. Objectives

Specifically designed for the Windows 10 operating system. Enhancing the reliability and stability of wireless connections across a variety of devices and network environments. Optimizing data transmission rates to improve browsing speeds, streaming quality, and overall network performance. Ensuring compatibility with a wide range of hardware configurations to maximize user accessibility. Providing comprehensive documentation and support resources to facilitate easy installation and troubleshooting for end-users.

We've rigorously tested and fine-tuned the driver to ensure it is reliable and secure. Plus, we'll provide easy-to-follow instructions and support resources to make setup a breeze.

1.3. Project Scope

This project focuses on developing a high-performance Wi-Fi driver specifically designed for Windows 10. Here's a breakdown of the scope:

1.4. Target Platform

Target platform is the Windows 10 operating system for which we have developed our driver. Windows 10 (both 32-bit and 64-bit versions). This ensures broad compatibility with most modern Windows 10 machines. The driver will be designed to enhance the reliability, stability, and overall performance of wireless connections for Windows 10 users. This means focusing on improvements in areas like Minimizing dropouts and disconnects. Optimizing speeds for browsing, streaming, and downloads. Working seamlessly with a wide range of Wi-Fi adapters. The project will not focus on developing drivers for other operating systems like Windows 7, 8, or 11. It will not include hardware modifications to Wi-Fi adapters themselves.

2. Methodology

The methodology involves comprehensive research into Windows Driver Kit (WDK) and Wi-Fi driver development guidelines. Development proceeds with iterative testing and debugging cycles to ensure compatibility and reliability with Windows 10.

2.1. Overview of Development Process

Building a robust Wi-Fi driver involves a series of crucial steps. Here's a roadmap to guide you through the process:

2.1.1 Microsoft Visual Studio

This powerful IDE (Integrated Development Environment) will be your coding hub. It allows you to write, compile, and debug your driver code.

2.1.2 Windows Driver Kit (WDK)

Consider this your Wi-Fi driver toolkit. It provides essential tools, libraries, and documentation specifically designed for Windows driver development.

2.1.3 Mastering the Language

Delve into the world of C++ programming, focusing on kernel-mode development. Kernel-mode drivers operate at the core of the operating system, interacting directly with hardware.

2.1.4 Wi-Fi Driver Development

Grasp the core concepts and principles behind Wi-Fi driver development. This includes understanding how the driver interacts with Wi-Fi adapters and manages wireless communication.

2.1.5 Learning from Examples

Don't reinvent the wheel! Explore the treasure trove of sample Wi-Fi driver projects offered by Microsoft in the Windows Driver Samples repository.

2.1.6 Testing and Certification

Rigorous testing is paramount. You'll need to create a comprehensive testing plan that evaluates your driver's functionality, compatibility, and performance across various scenarios.

2.1.7 Windows Hardware Certification Kit (HCK)

To ensure broad compatibility and user confidence, consider using the Windows Hardware Certification Kit (HCK).

2.1.8 Staying Ahead of the Curve

The world of technology is ever-evolving. Actively stay updated with the latest resources and best practices by following the Windows Hardware Dev Centre.

2.2 User Mode Approach

In the user mode approach, the study area revolves around creating a Wi-Fi driver that operates within the user space of the operating system. This involves developing software that interacts with the Wi-Fi input and communicates with the operating system through user-mode APIs (Application Programming Interfaces).

2.3 Kernel Mode Approach

In the kernel mode approach, the study area focuses on creating a keyboard driver that operates within the kernel space of the operating system. This involves developing software that has direct access to hardware resources and system functions, providing enhanced performance and control. I have choose kernel mode for my project.

2.4 Tools and Technologies

Tools	Specifications
Visual Studio	64-bit VS Studio with desktop development with C++ and x64 latest spectre mitigated components.
Windows driver kit (WDK)	Latest for windows 10 & 11
SDK	Windows 10 & 11
Development System	8 GB RAM, 40 GB storage
Target System	WDK installed, Target Operating system
Languages	C, C++

Table 1 Tools and specification

2.5 Requirements Analysis

Requirement analysis involves identifying essential functionalities like network discovery, connection management, and security protocols. Additionally, non-functional requirements such as performance benchmarks and compatibility with various hardware configurations are crucial considerations. This phase aims to define clear specifications and criteria to guide the development process effectively.

2.5.1 Functional Requirements

Functional requirements are necessary for the project development without them project will not work properly.

Wireless Connectivity:

The driver should facilitate seamless wireless connectivity to Wi-Fi networks.

Network Discovery:

Ability to scan for and identify available Wi-Fi networks within range, including hidden networks.

Authentication and Encryption:

Support for authentication methods (e.g., WPA2-PSK, WPA3) and encryption protocols (e.g., AES) to ensure secure communication. It is one of important

functionalities.

Configuration Options:

Enable users to configure network settings such as SSID, password, security type, and IP addressing (static or dynamic).

Roaming Support:

Smooth transition between different access points within the same network to maintain uninterrupted connectivity while moving between coverage areas.

Power Management:

Implement power-saving features to optimize battery life when operating on mobile devices, including idle and sleep modes.

Quality of Service:

Prioritize network traffic to ensure smooth performance for bandwidth-intensive applications like video streaming and online gaming.

Diagnostic Tools:

Include diagnostic capabilities to troubleshoot connectivity issues, such as signal strength indicators, packet loss detection, and connection status reports.

2.5.2 Non-Functional Requirements

Non-functional requirements are optional but important for long-lasting projects.

Performance:

The driver should deliver high-speed data transmission rates with low latency, even under heavy network traffic conditions.

Reliability:

Ensure stable and consistent connectivity with minimal connection drops or interruptions.

Compatibility:

Compatibility with a wide range of hardware configurations, including different Wi-Fi chipsets and device manufacturers.

Security:

Implement robust security measures to protect against unauthorized access, data breaches, and cyber threats.

Scalability:

Scalable architecture to accommodate future updates and enhancements,

ensuring long-term viability and support.

Ease of Use:

User-friendly interface and installation process, with clear documentation and support resources for end-users.

Resource Efficiency:

Optimize resource utilization, including memory and CPU usage, to minimize system impact and maximize performance.

Regulatory Compliance:

Adherence to industry standards and regulations for wireless communication devices and software.

3. Implementation

The implementation phase involves writing code to fulfill the specified requirements, including device interaction, data processing, and error handling. It encompasses translating design specifications into executable code, adhering to coding standards and best practices, and conducting rigorous testing to ensure functionality and reliability. Additionally, documentation and version control are essential aspects of this phase to facilitate maintenance and future enhancements.

3.1. Environment Setup

Starting off with the first step, download and installation of Visual Studio. You can download it from the official website. After downloading, execute the setup file and choose "desktop development with C++" from workloads. From individual components in Visual Studio, install "64 latest Spectre".

You can download SDK and WDK from the official website of Microsoft. Setup Visual Studio for driver development. Open Visual Studio > Create a new project > Select appropriate template depending on user mode or kernel mode.

Right-click on the project name in Solution Explorer and click "Add" > "New Item". Under properties, you can find all settings related to the project.

3.2. Main Functions Implemented In Code

Driver Entry Function:

Initializes the driver and creates a device object. Assigns dispatch routines for

major IRP functions such as create, close, and device control. Handles custom IRP functions for scanning devices and connecting to a network.

Driver Unload:

Cleans up resources when the driver is unloaded.

Driver Create and Driver Close:

Dispatch routines for handling create and close requests from user-mode applications.

Driver Device Control:

Dispatch routine for handling device control requests, including custom IOCTL codes.

Driver Scan Devices and Driver Connect Network:

Dispatch routines for handling custom IRP functions, namely scanning devices and connecting to a network.

Monitor Internet Speed:

Placeholder function for monitoring internet speed. It should be replaced with actual implementation logic.

3.3. Challenges Faced

Here's a breakdown of the challenges encountered and the solutions implemented:

Spectre Mitigated Library Error:

Encountered an error related to Spectre mitigations likely within a library being used.

Solution:

Successfully addressed the issue by adding the latest x64 Spectre-mitigated library from the components being used to ensure leveraging the latest security patches.

.inf File Error:

Challenge: Faced an error related to the .inf file, which plays a critical role in driver installation on Windows.

Solution:

Identified and fixed the specific error within the .inf file itself, ensuring proper driver installation.

Pinging the Target Machine:

Establishing communication (pinging) between the development machine and the target machine with the Wi-Fi adapter being developed for.

Solution:

Ensured correct network configuration, checked for firewall issues, and ensured driver functionality on the target machine.

Deployment:

Utilized tools provided by Microsoft or the development environment to automate the deployment process.

3.4. Testing Methodologies

Rigorous testing is paramount for any Wi-Fi driver to ensure it functions correctly and delivers optimal performance across various scenarios. Here are some key testing methodologies employed during Wi-Fi driver development:

3.4.1 Unit Testing

Focuses on testing individual components and functionalities of the driver in isolation. This helps identify bugs early in the development process, leading to faster debugging and code improvement. Unit testing frameworks like Microsoft Driver Verifier (Verifier.exe) can be used to simulate various hardware and software conditions, stressing the driver and uncovering potential issues.

3.4.2 Hardware Testing

Involves testing the driver on various Wi-Fi adapter hardware configurations to ensure compatibility and proper functionality across different models and vendors. This can be achieved using testing on a variety of real-world Wi-Fi adapters to ensure broad compatibility.

3.4.3 Functionality Testing:

Evaluates core functionalities of the Wi-Fi driver like, Connecting to access points with different security protocols (WEP, WPA, WPA2, etc.). Verifying successful authentication with access points. Testing data transmission and reception capabilities under various network conditions (wired and wireless). Evaluating the driver's ability to manage power consumption of the Wi-Fi adapter.

3.4.4 Performance Testing

Assesses the driver's performance metrics like measuring the speed of data transfer under different network loads and protocols. Evaluating the time it takes for data packets to travel between the device and the network. Testing the driver's ability to maintain a strong and stable connection under varying signal conditions.

3.4.5 Stress Testing

Pushes the driver to its limits by simulating extreme network conditions and heavy workloads. This helps identify potential bottlenecks and ensure the driver can handle peak usage scenarios without crashing or malfunctioning.

3.4.6 Certification Testing

Consider using the Windows Hardware Certification Kit (HCK) to ensure your driver meets Microsoft's compatibility and performance standards. This can increase user confidence and marketability for your driver.

3.5 Testing On Target System

Testing the driver on the target system to assess its working.

3.5.1 Provisioning Target Machine:

After creating a basic structure of the Wi-Fi driver, the next step was to test and deploy. I found three methods for this:

Develop and test on the same machine:

This method involves developing and testing the driver on the same machine where the development environment is set up.

Use Virtual Machine:

Another approach is to use a virtual machine (VM) for testing purposes. However, there might be compatibility issues, especially if the VM has to run on the same system with processor-related issues.

Use another target machine:

I chose the third option, which is to use a separate target machine for testing. This option best suited me as the Windows Driver Kit (WDK), a crucial component in testing and deploying, was having issues with my processor. Using a third-party target machine allows for efficient testing and error handling.

Requirements for Target System

One of the requirements is that since I was creating this driver for Windows,

the target machine should have a Windows operating system for efficient testing, error handling, and monitoring the performance of the Wi-Fi driver on the Windows operating system.

3.6 Target System Deployment

To prepare the target computer for provisioning, follow these steps:

1. Install the WDK:

Install the Windows Driver Kit (WDK) on the target computer. You only need to install Visual Studio if you plan on doing driver development on the target computer.

2. Disable Secure Boot (if enabled):

If Secure Boot is enabled on the target computer, disable it. This option is typically found in the boot menu under advanced or security features. For information about Unified Extensible Firmware Interface (UEFI) and Secure Boot, refer to the UEFI Firmware documentation.

3. Enable Test Signing:

If you will be deploying a test driver on the target machine, enable test signing from an elevated command prompt with the command `bcdedit /set testsigning on`.

4. Run WDK Test Target Setup MSI:

On the target computer, run the WDK Test Target Setup MSI that matches the platform of the target computer. You can typically find this MSI file at `C:\Program Files (x86)\Windows Kits\10\Remote`.

5. Verify Installation:

Once the installation is complete, verify that the WDK Test Target environment is set up correctly by checking for any installed components or utilities related to driver testing.

6. Restart the Target Computer (if required):

Some installations may require a restart to apply changes fully. If prompted, restart the target computer to ensure that the WDK Test Target environment is ready for use.

7. Ping Test:

Ping both laptops by name or IP address to ensure network connectivity. You can do this by executing the command `ping Target PC name` or `ping Target`

PC IP address.

3.7 Host System Configuration

To deploy the driver onto the target system, follow these steps:

1. On the host computer, open Visual Studio.
2. Select the Extensions menu.
3. Point to Driver, then point to Test, and select Configure Devices.
4. In the Configure Devices dialog, select Add new device.
5. For Network host name, enter the name or local IP address of your target computer.
6. Add the new device and in the placeholder, write the testing PC name and IP address.
7. The provisioning process takes several minutes and might automatically reboot the target computer once or twice.
8. When provisioning is complete, select Finish.

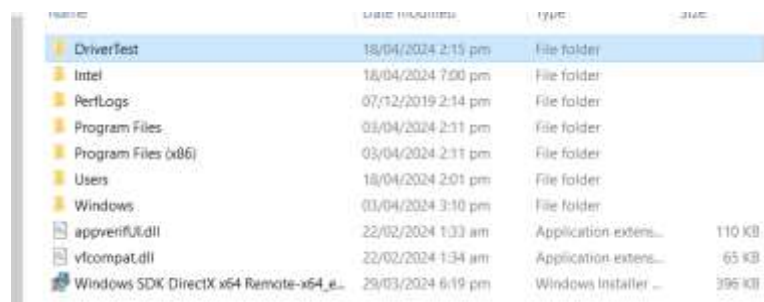
This process ensures that the driver is deployed onto the target system for testing purposes.

4. Deployment

After successful deployment, a new folder named DriverTest will be created in the C drive. This folder contains all the necessary files of the newly deployed driver.

1. Driver Test Folder Creation

After deployment, a folder named DriverTest is created in the C drive.

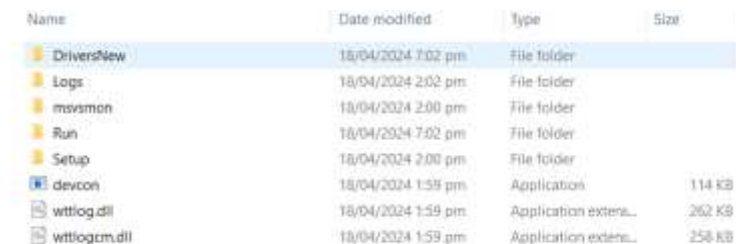


Name	Date modified	Type	Size
DriverTest	18/04/2024 2:13 pm	File folder	
Intel	18/04/2024 7:00 pm	File folder	
PerfLogs	07/12/2019 2:14 pm	File folder	
Program Files	03/04/2024 2:11 pm	File folder	
Program Files (x86)	03/04/2024 2:11 pm	File folder	
Users	18/04/2024 2:01 pm	File folder	
Windows	03/04/2024 3:10 pm	File folder	
appverifx.dll	22/02/2024 1:33 am	Application extension	110 KB
vlcompat.dll	22/02/2024 1:34 am	Application extension	65 KB
Windows SDK DirectX x64 Remote-x64_e...	29/03/2024 6:19 pm	Windows Installer package	396 KB

Figure 1 Test Driver Folder

2. Contents of Driver Test Folder:

Inside the DriverTest folder, there are other folders related to the driver test and installation.

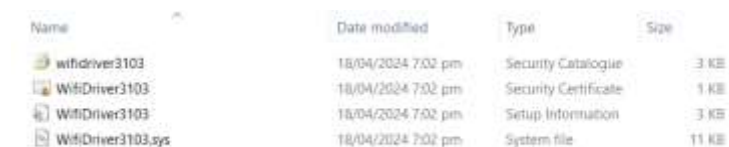


Name	Date modified	Type	Size
DriversNew	18/04/2024 7:02 pm	File folder	
Logs	18/04/2024 2:02 pm	File folder	
msvsmon	18/04/2024 2:00 pm	File folder	
Run	18/04/2024 7:02 pm	File folder	
Setup	18/04/2024 2:00 pm	File folder	
devcon	18/04/2024 1:59 pm	Application	114 KB
wttlog.dll	18/04/2024 1:59 pm	Application extension	262 KB
wttlogcm.dll	18/04/2024 1:59 pm	Application extension	258 KB

Figure 2 files inside driver test folder

3. Important Files:

The most important files in this folder are the .inf and .sys files, which are essential for the driver to work.



Name	Date modified	Type	Size
wifidriver3103	18/04/2024 7:02 pm	Security Catalogue	3 KB
WifiDriver3103	18/04/2024 7:02 pm	Security Certificate	1 KB
WifiDriver3103	18/04/2024 7:02 pm	Setup Information	3 KB
WifiDriver3103.sys	18/04/2024 7:02 pm	System file	11 KB

Figure 3 Important files of driver

4. Integration into System:

To integrate the driver into the system, open Device Manager, click on Action, then

select Add legacy hardware, and add the newly created device driver from the DriverTest folder.

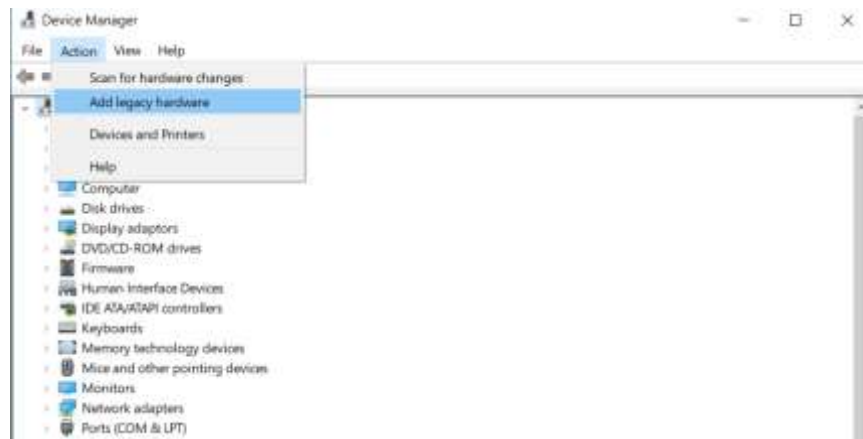


Figure 4 Device Manager

5. Selecting the Driver .inf File:

Choose the .inf file for your driver. For example, if the driver name is “WifiDriver3103”, select its .inf file. Add the driver in the list below or choose custom options.

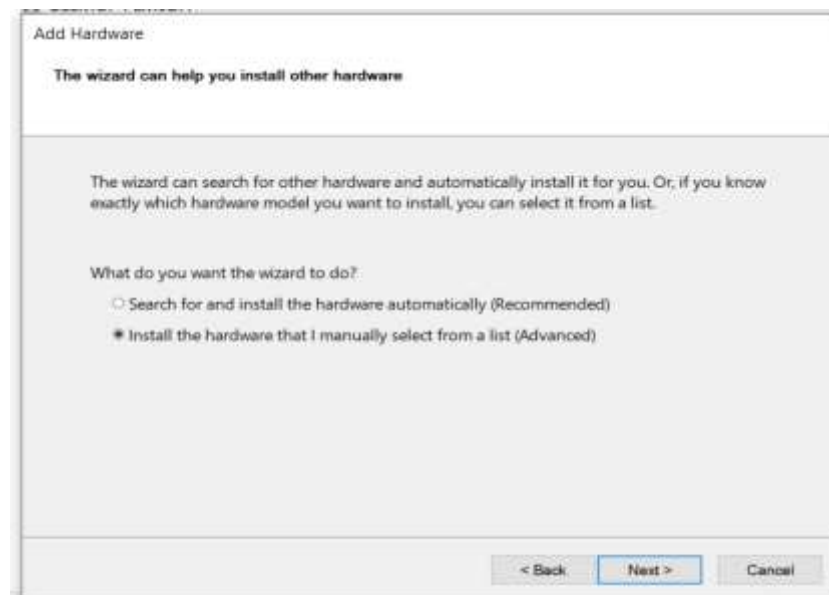


Figure 5 Select Installation type

6. Using Have Disk Option:

Select the Have Disk option.

7. Choosing Path for the Driver:

Choose the path for your driver or browse to locate it.

8. Driver Selection:

- The driver has been successfully selected for installation.

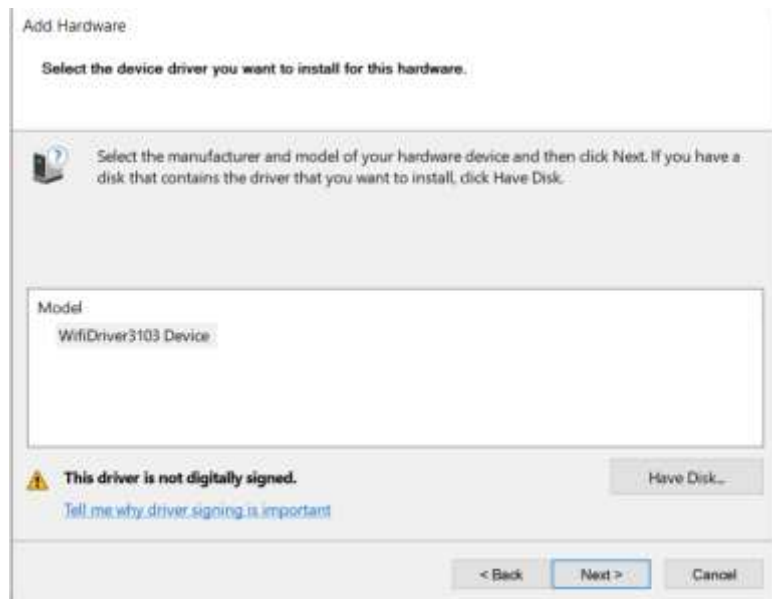


Figure 6 Driver loaded

9. Driver Installation:

- The driver is installed successfully.

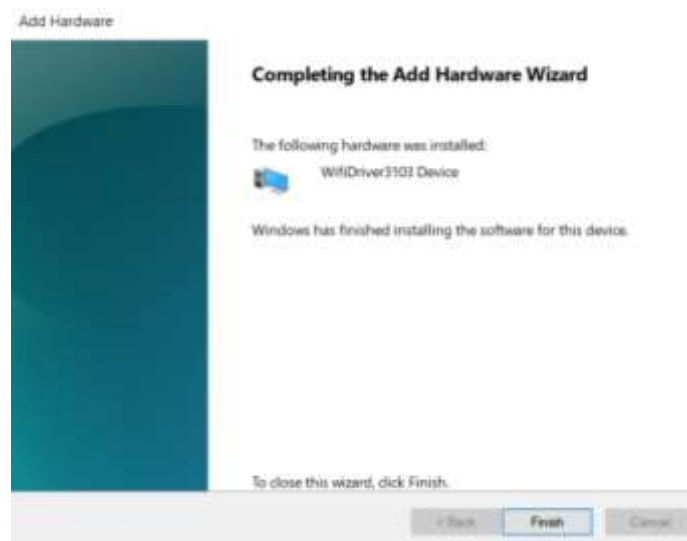


Figure 7 Successful Integration

10. Verify Installation:

- Verify the installation by checking in the System Devices dropdown.

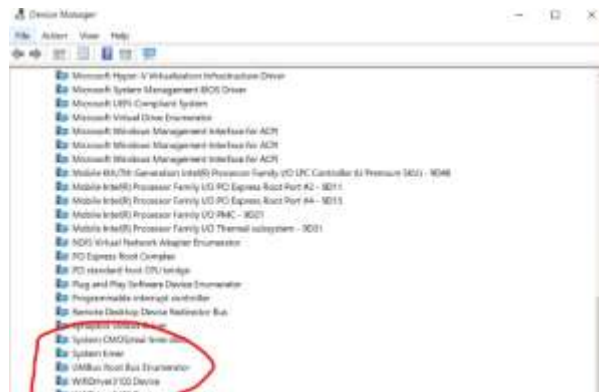


Figure 8 Verification of insatll

11. Successful Installation:

- The driver is successfully installed and ready for use.

12. Driver Properties:

- Check the properties of the installed driver for further verification.

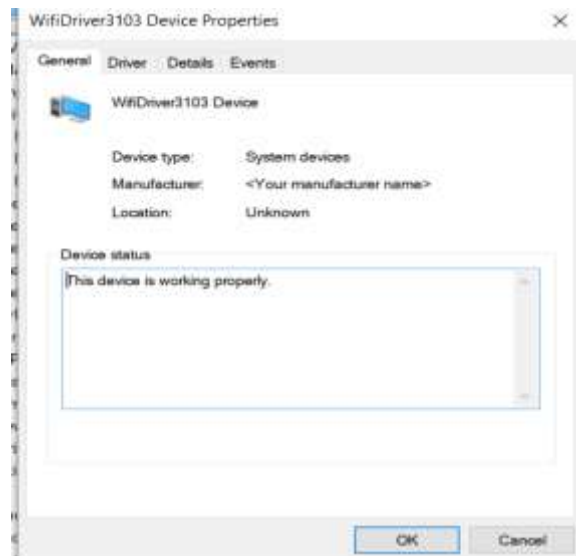


Figure 9 properties

By following these steps, the newly created driver can be deployed and integrated into the system effectively.

5. Conclusion and Reference

The successful development of a robust Wi-Fi driver with comprehensive functionalities tailored for Windows 10 marks a significant achievement. The implementation of Wi-Fi driver deployment procedures ensures successful integration with target devices, enhancing user experience.

5.1. Lessons Learned

Meticulous planning and preparation are crucial in Wi-Fi driver development to ensure project success. Effective utilization of debugging tools and troubleshooting techniques enhances development efficiency. Code organization, documentation, and code review are essential for maintaining code quality and adherence to standards.

5.2.1 Insights Gained:

Deepened understanding of Wi-Fi driver development concepts, frameworks, and industry best practices. Enhanced knowledge of Wi-Fi technology, protocols, and driver implementation strategies enriches technical expertise. Improved problem-solving skills and systematic challenge addressing ability foster continuous learning and growth.

5.2. References

1. Microsoft Developer Network (MSDN) - Official documentation for Windows Driver Kit (WDK). <https://docs.microsoft.com/en-us/windows-hardware/drivers/>
2. Windows Dev Center - Resources and documentation for driver development on the Windows platform. <https://developer.microsoft.com/en-us/windows>
3. NDIS Programming Reference - Documentation for Network Driver Interface Specification (NDIS). <https://docs.microsoft.com/en-us/windows-hardware/drivers/network/>
4. Wi-Fi Alliance - Official website for Wi-Fi technology standards and specifications. <https://www.wi-fi.org/>
5. Stack Overflow - Online community for developers, valuable for problem-solving and troubleshooting. <https://stackoverflow.com/>
6. GitHub - Open-source code hosting platform providing access to Wi-Fi

driver repositories and development resources. <https://github.com/>

7. IEEE Standards Association - Organization responsible for developing Wi-Fi technology standards. <https://standards.ieee.org/>
8. "Developing Drivers with the Windows Driver Foundation" - Book by Penny Orwick and Guy Smith, a comprehensive reference for Windows operating system internals and driver development. <https://www.amazon.com/Developing-Drivers-Foundation-Developer-Reference/dp/0735623740>

This project has not only resulted in a functional Wi-Fi driver but has also provided valuable insights and learning experiences for future endeavors.

Closing Remarks and Gratitude

Dear Sir,

I extend my heartfelt gratitude to you for your unwavering support and guidance throughout this project journey. Your valuable insights and encouragement have been instrumental in our success.

Your dedication to excellence and passion for innovation have truly inspired us. Your mentorship has not only enriched our project but also contributed to our personal and professional growth.

I am immensely grateful for the trust you placed in our team and for the opportunities you provided for us to learn and excel. Your belief in our abilities has motivated us to strive for excellence and push the boundaries of what we thought possible.

As we embark on the next phase of our journey, I carry with me the invaluable lessons and experiences gained under your guidance. Your mentorship will continue to guide me in my future endeavors, and I am profoundly grateful for the impact you have had on my development.

Once again, thank you for everything. Your support has made all the difference, and I am honored to have had the opportunity to work alongside you.

With warm regards,

Ahmed Ali

&

Seher Ishtiaq