

SECURE BARCODE GENERATION AND DECODING WITH ENCRYPTION TECHNIQUES

P.V.N.SIRISHA(192210231)
B.VARSHITHA(192210420)



ABSTRACT

- This project focuses on the development of a secure barcode generation and reading system that utilizes Caesar cipher encryption to ensure data confidentiality.
- By integrating sensor-based hardware, the system can capture and decode barcodes.
- System also protects sensitive information by encrypting it before embedding into the barcode.
- This method is especially beneficial for applications requiring secured data transmission, such as in logistics, healthcare, and secure identification processes.
- A secure and efficient barcode system that ensures confidentiality in automated identification.

INTRODUCTION

- In today's digital world, the need for secure data handling within automated identification systems is greater than ever
- This project addresses these security concerns by introducing an innovative method to enhance the protection of barcode data through encryption.
- At the core of this system is the **Caesar cipher**, a classical encryption technique that shifts each character in the data by a fixed number of positions.
- After the data is encrypted, it is transformed into a barcode format, making it easy to scan and compatible with a wide range of hardware.
- The barcode is then read by a specially configured device capable of scanning

OBJECTIVES

Generate barcodes that encapsulate secure information using Caesar cipher encryption.

Develop a hardware-integrated system for reading and decoding encrypted barcodes

Implement Caesar cipher encryption to restrict unauthorized access to barcode data.

Evaluate the effectiveness of the encryption method in ensuring data confidentiality

Environmental Parameter Configuration

Barcode Size:

Ensures the readability and accuracy of generated barcodes in various lighting and environmental conditions.

Contrast Settings:

Adjusts hardware settings to improve barcode scanning accuracy, especially under different lighting conditions

Distance and Alignment: Calibrates distance and alignment for optimal barcode capture, critical in hardware setup.

Sensor Integration and Data Capture

Sensor Selection:

Choose a reliable barcode scanner compatible with standard barcode formats and Caesar cipher encoding.

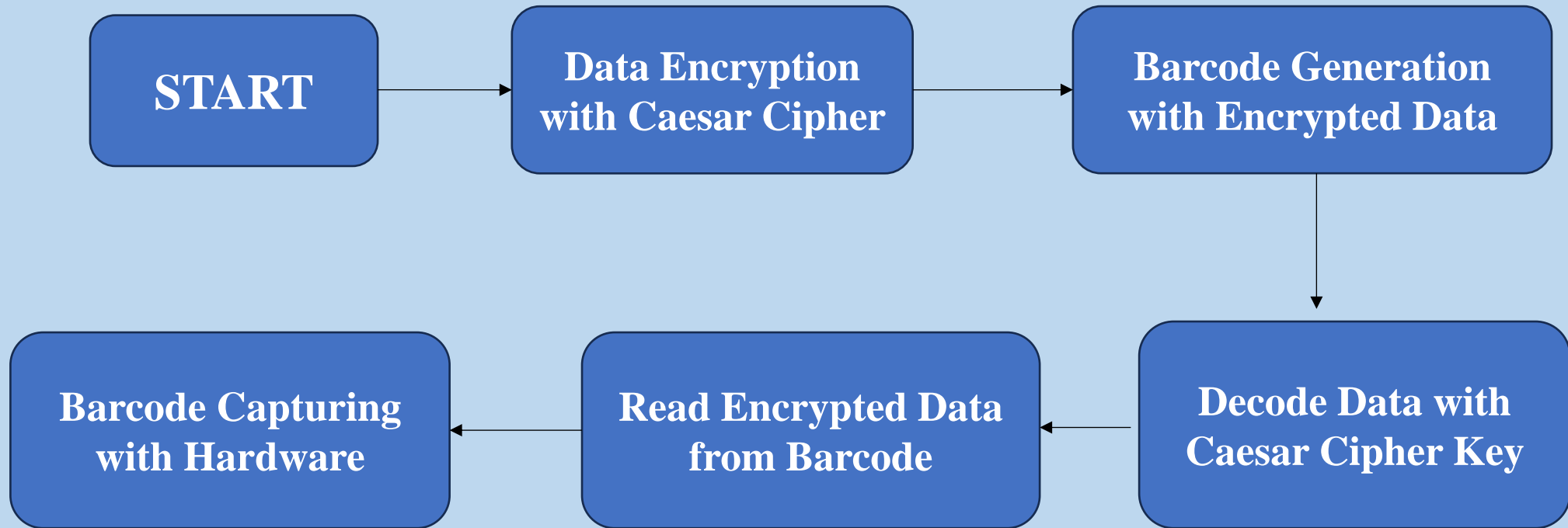
Data Capture:

Integrate the sensor with a microcontroller to read and transmit barcode data to a connected system.

Error Handling:

Implement error correction for reading inconsistencies, ensuring accurate capture and decoding

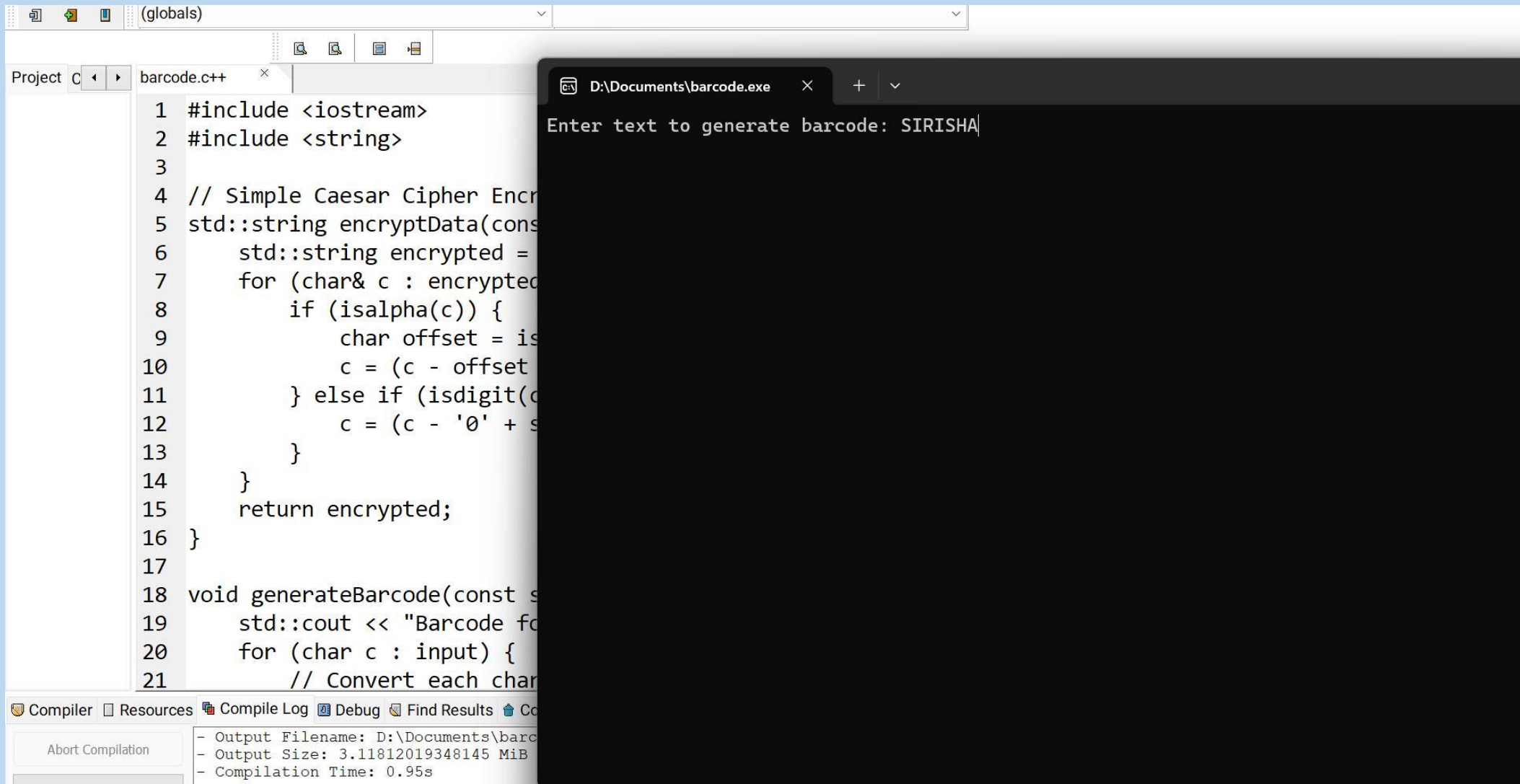
METHODOLOGY



Code Explanation

1. Library Inclusion: Including essential libraries for input/output and string handling.
2. Encryption Function Definition: Creating the encryptData function to apply Caesar cipher encryption.
3. Shifting Alphabet Characters : Applying Caesar cipher logic to shift alphabetic characters within their respective ranges.
4. Shifting Numeric Characters: Shifting numeric characters (0-9) and handling wrap-around to maintain valid digit values.
5. Barcode Generation Function: Defining generateBarcode to convert characters into a visual binary representation for a simple barcode.
6. Main Program Execution: Prompting user input, encrypting text, and displaying encrypted output
- .7. Generating and Displaying the Barcode: Visualizing the encrypted data as a binary-style barcode and displaying it to the user.

Output



The image shows a screenshot of a C++ IDE. The left pane displays the source code for `barcode.cpp`. The right pane shows the execution output of the program, which prompts the user to enter text to generate a barcode. The input "SIRISHA" is shown, and the program is in the process of generating the barcode.

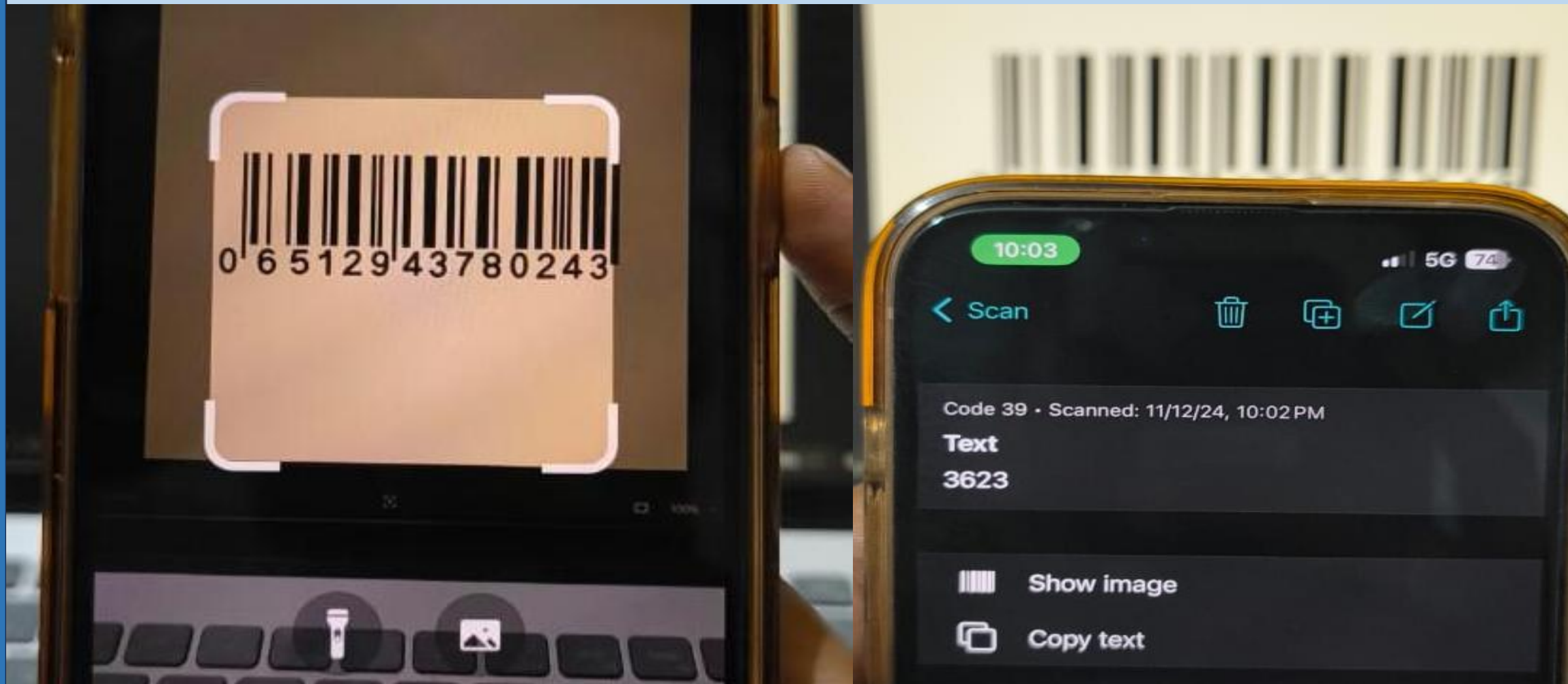
```
1 #include <iostream>
2 #include <string>
3
4 // Simple Caesar Cipher Encr
5 std::string encryptData(const
6     std::string encrypted =
7     for (char& c : encrypted
8         if (isalpha(c)) {
9             char offset = is
10             c = (c - offset
11         } else if (isdigit(c
12             c = (c - '0' + s
13         }
14     }
15     return encrypted;
16 }
17
18 void generateBarcode(const s
19     std::cout << "Barcode fo
20     for (char c : input) {
21         // Convert each char
```

Output: Enter text to generate barcode: SIRISHA

Compiler Output:

- Output Filename: D:\Documents\barcode.exe
- Output Size: 3.11812019348145 MiB
- Compilation Time: 0.95s

RESULTS



Future Scope

- The future scope of this project includes integrating advanced encryption algorithms like AES or RSA for stronger security.
 - Expanding to dynamic barcode formats such as QR codes or Data Matrix for greater data capacity is possible.
 - Real-time barcode scanning and mobile/web integration could enhance usability.
 - Adding error detection and correction mechanisms will improve reliability.
- Multi-layer security measures, such as digital signatures, could further strengthen data protection.

Conclusion

- This project demonstrates the process of generating and reading barcodes while securing the encoded data using the Caesar cipher encryption technique.
- By combining basic encryption with barcode technology, it offers a simple yet effective solution for securing data in environments where confidentiality is important.
- The barcode system allows easy encoding of encrypted information and provides a visual representation for scanning and decoding.
- While the current implementation is functional, future improvements can focus on enhancing security, scalability, and usability, making the system more robust and adaptable to diverse applications.



THANK YOU

The image features a central white rectangular area containing the text "THANK YOU" in a blue, sans-serif font. This central area is flanked by two horizontal bands of a blue geometric pattern composed of various-sized triangles. The entire composition is set against a solid dark blue background.