TRIBHUVAN UNIVERSITY (12, Bold, Center Justified)

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Minor Project Report

On

ATM Machine

Submitted By:

Purushottam Raj (THA081BCT026)

Anuj Singh(THA081BCT003)

Rajan Neupane (THA081BCT027)

Raman Rajbanshi (THA081BCT029)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

March, 2025

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Minor Project Report
On
ATM MACHINE

Submitted By: (12, Bold, Center Justified)

[Student Name] ([Student Exam Roll No.]) (12, Center Justified)

[Student Name] ([Student Exam Roll No.])

[Student Name] ([Student Exam Roll No.])

[Student Name] ([Student Exam Roll No.])

Submitted To:

Department of Electronics and Computer Engineering
Thapathali Campus
Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in Computer Engineering.

Under the Supervision of

Prajwol Pakka

March, 2025

DECLARATION

We hereby declare that the report of the project entitled **"ATM Machine"** which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Computer** is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this work.

Anuj Singh (Class Roll No: THA081BCT003) _____

Purushottam Raj (Class Roll No: THA081BCT026) _____

Rajan Neupane (Class Roll No: THA081BCT027) _____

Raman Rajbansi (Class Roll No: THA081BCT029) _____

Date: March,2025

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a minor project work entitled "**ATM Machine**" submitted by **Anuj Singh, Purushottam Raj, Rajan Neupane** and **Raman Rajbansi** in partial fulfillment for the award of Bachelor's Degree in Computer Engineering. The Project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor's degree of Computer Engineering.

_____

Project Supervisor
Prajwol Pakka
Department of Electronics and Computer Engineering, Thapathali Campus

_____

External Examiner
Department of Electronics and Computer Engineering, Pulchowk Campus

_____

Project Co-ordinator
Department of Electronics and Computer Engineering, Thapathali Campus

_____

Mr.Umesh Kant Ghimire
Head of the Department,
Department of Electronics and Computer Engineering, Thapathali Campus

March, 2025

COPYRIGHT

ACKNOWLEDGEMENT

We express our sincere thanks to the Institute of Engineering, Thapathali Campus, forgiving us the chance to improve our knowledge and abilities through this project. We extend our sincere gratitude to the Department of Electronics and Computer Engineering for their ongoing assistance and for providing us with essential resources and information. We are particularly grateful to our supervisors, Er. Prajwol Pakka ,along with Er. Anup Shrestha, for their essential guidance, support, and expert insights throughout this project's duration. Their guidance was vital for the successful fulfillment of our project.

We would also like to convey our gratitude to our colleagues and our supervisors for their support and encouragement. This project was more than just an academic task; it was an experience that enabled us to use our practical skills in System Development, particularly in the C programming language. The educational experience and practical application have been incredibly advantageous, and we appreciate everyone who contributed to this learning journey.

Anuj Singh (Class Roll No.: THA081BCT003)

Purushottam Raj (Class Roll No.: THA081BCT026)

Rajan Neupane (Class Roll No.: THA081BCT027)

Raman Rajbansi (Class Roll No.: THA081BCT029)

ABSTRACT

The swift global transition to online transactions highlights the critical demand for digital financial solutions, particularly in developing nations such as Nepal, where conventional banking methods dominate .But along with that traditional method ATM us also a mid-modern banking service when mob banking doesnot exist. Our initiative presents a prototype aimed at simplifying minor, essential transactions via a bank service, thereby knowing the importance of ATM where there is no bank branches to withdraw. Using the C programming language, which is renowned for its strong system-level features, we created a ATM wallet enabling users to make payments with ease. This effort not only improves the ease of financial transactions but also utilizes educational elements of C, incorporating every concept and method learned throughout our first semester.

Keywords: C-Programming ,ATM ,Withdraw , financial transaction.

Table of Contents

DECLARATION ..... **i**

CERTIFICATE OF APPROVAL ..... **ii**

COPYRIGHT ..... **iii**

ACKNOWLEDGEMENT ..... **iv**

ABSTRACT ..... **v**_Toc14879675

List of Figures ..... **viii**

List of Tables ..... **ix**

List of Abbreviations ..... **x**

1. INTRODUCTION **1**

1.1 Background 1

1.2 Motivation 1

1.3 Problem Definition 1

1.4 Project Objectives 2

1.5 Project Scope and Applications 2

1.6 Report Organization 2

2. LITERATURE REVIEW 4

2.1 Sub-heading 1 .....**Error! Bookmark not defined.**

2.2 Sub-heading 2 and so on.....**Error! Bookmark not defined.**

3. REQUIREMENT ANALYSIS 6

3.1 Sub-heading 1 – Project Requirement (may be Hardware and Software Requirements) .....**Error! Bookmark not defined.**

3.2 Sub-heading 2 – Feasibility Study.....**Error! Bookmark not defined.**

4. SYSTEM ARCHITECTURE AND METHODOLOGY 8

4.1 Sub-heading 1 – Block Diagram/Architecture**Error! Bookmark not defined.**

4.2 Sub-heading 2 – Flowcharts/Algorithms (and other design methods) **Error! Bookmark not defined.**

5. IMPLEMENTATION DETAILS 18

5.1 Sub-heading 1 for e.g. Hardware Components**Error! Bookmark not defined.**

5.2 Sub-heading 2 .....**Error! Bookmark not defined.**

5.2.1 Sub-heading of 5.2 .....**Error! Bookmark not defined.**

6. RESULTS AND ANALYSIS 19

6.1 Sub-heading 1 .....**Error! Bookmark not defined.**

6.1.1 Sub-heading 1 of 6.1 .....**Error! Bookmark not defined.**

6.2 Sub-heading 2 .....**Error! Bookmark not defined.**

7. FUTURE ENHANCEMENT 20

8. CONCLUSION 22

9. APPENDICES 23

Appendix A: Circuit Diagram 23

Appendix B: PCB Diagram 24

Appendix C: Zigbee Module Specification 26

Appendix D: Used Linux Commands 27

References.....30

List of Figures

Figure 4-1: Block Diagram of abc ..... **Error! Bookmark not defined.**
Figure 4-2: Flowchart of xyz ..... **Error! Bookmark not defined.**
Figure 6-1: figure xyz ..... **Error! Bookmark not defined.**
Figure 9-1: Circuit Diagram of AVR with sensors and XBee 23
Figure 9-2: Circuit Diagram of Motor Driver and Relay.....24
Figure 9-3: PCB Diagram of AVR with sensors and Xbee (Top Layer).....24
Figure 9-4: PCB Diagram of AVR with sensors and Xbee (Bottom Layer) 25
Figure 9-5: PCB Diagram of Motor Driver and Relay (Top Layer).....25
Figure 9-6: PCB Diagram of Motor Driver and Relay (Bottom Layer) 26

List of Tables

Table 9-1: XBee S2 Pin Description.....26
Table 9-2: XBee S2 General Features 26

List of Abbreviations

AC	Alternating Current
ADC	Analog to Digital Converter
API	Application Programming Interface
BCM	Broadcom
CMOS	Complementary Metallic Oxide Semiconductor
CMS	Center Monitoring System
CSS	Cascading Style Sheets
DHCP	Dynamic Host Control Protocol
et al.	And Others
FTP	File Transfer Protocol
FTPS	File Transfer Protocol Secure
GIS	Geographic Information System
GPIO	General Purpose Input Output
GPRS	General Packet Radio Service

GPS	Global Positioning System
GPU	Graphics Processor Unit
GUI	Graphical User Interface
HDMI	High Definition Media Interface
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
IDLE	Integrated Development Environment
IP	Internet Protocol
IOT	Internet Of Things
ISP	Internet Service Provider
ISR	Interrupt Service Routine
IT	Information Technology

1. INTRODUCTION

The project contains several C source files (`.c`), header files (`.h`), and an executable (`neo_main.exe`). The project is related to a ***user login system along with financial transaction or an administrative tool*** with functions for handling users (`neo_user.c`), login (`neo_login.c`), administrative actions (`neo_admin.c`), and time-related operations (`neo_time.h`).

1.1 Background

This project is a ***basic ATM simulation*** developed in C, featuring a menu-driven interface for user authentication, account management, and transaction handling. It allows users to log in, register, and perform banking operations, while administrators may have additional control over account settings. The system integrates multiple modules, including login validation, user account handling, and administrative functionalities, providing a structured approach to simulating ATM operations.

1.2 Motivation

The motivation behind this project is to ***simulate a basic ATM system*** that enhances understanding of ***banking transactions, user authentication, and account management*** using C programming. It provides hands-on experience in ***file handling, modular programming, and security measures*** while demonstrating real-world banking operations in a controlled environment.

1.3 Problem Definition

The problem this project addresses is the ***need for a secure and efficient ATM system*** that allows users to perform banking transactions such as deposits, withdrawals, and balance inquiries. Traditional banking processes can be time-consuming and prone to errors, making automation essential. This project aims to ***simulate a functional ATM***, ensuring secure user authentication, seamless transactions, and administrative control, while also serving as a learning tool for ***C programming and financial system simulations***

1.4 Project Objectives

The main objectives of our project are listed below **:**

- **To develop a secure and user-friendly ATM simulation*** that allows users to log in, register, and perform basic banking transactions such as deposits, withdrawals, and balance inquiries.
- **To enhance understanding of C programming concepts*** by implementing modular programming, file handling, and authentication mechanisms in a real-world financial application.

1.5 Project Scope and Applications

Project Scope: - ***User Authentication:*** Secure login and registration system to prevent unauthorized access. - ***Basic Banking Transactions:*** Users can check balances, withdraw, and deposit money. - ***Admin Controls:*** Administrative functionalities for managing user accounts. - ***File Handling:*** Data storage and retrieval using files to maintain transaction records. - ***Menu-Driven Interface:*** Simple and interactive UI for ease of use. ###

Applications: - ***Banking Simulations:*** Can be used for educational purposes to understand ATM operations. - ***Software Development Training:*** Helps students and developers learn C programming concepts like authentication, file handling, and modular programming. - ***Prototype for Real ATM Systems:*** Can serve as a foundation for more advanced banking software.

1.6 Report Organization

Report Organization:*

1. ***Introduction:*** Provides an overview of the project, including its background, motivation, problem definition, objectives, and scope.

2. ***Literature Review:*** Discusses existing ATM systems, related technologies, and previous research or projects in this domain.

3. ***System Design and Methodology:*** Explains the architecture of the ATM system, module interactions, flowcharts, and algorithms used for user authentication and transactions.

4. ***Implementation:*** Details the coding approach, programming language (C), file handling techniques, and key functions of different modules (login, transactions, admin control).

5. ***Testing and Results:*** Describes test cases, system performance, error handling, and validation of functionalities through various user scenarios.

6. ***Conclusion and Future Scope:*** Summarizes project achievements, challenges faced, and potential enhancements such as database integration or security improvements.

2. LITERATURE REVIEW

2.1 Traditional ATM Systems* ### *

2.1.1 What is the Work About?* Traditional ATMs allow users to conduct financial transactions such as cash withdrawals, deposits, balance inquiries, and fund transfers using a physical ATM card and a PIN for authentication. ### *

2.1.2 How It Works?* ATMs use ***magnetic stripe or chip-based cards*** to store account information, which is read by a card reader. The system verifies the user's identity through ***PIN-based authentication*** and processes transactions via a secure banking network, following protocols such as ***ISO 8583*** for communication between ATMs and banking servers. ###

2.1.3 Importance and Applications - Provides ***24/7 access to banking services***, reducing reliance on human tellers. - Enhances ***banking convenience and accessibility***, allowing customers to perform transactions globally. - Facilitates ***quick and secure cash withdrawals*** without the need to visit a bank branch. ###

2.1.4 Drawbacks and Limitations* - ***Security risks***: Traditional ATMs are vulnerable to ***card skimming, shoulder surfing, and PIN theft***. - ***Dependency on physical cards***: Losing a card prevents access to funds and requires replacement. - ***High maintenance costs***: ATMs require ***regular servicing, security monitoring, and cash refilling***. ###

2.1.5 Criticism & Link to Project Motivation Traditional ATMs have significant ***security vulnerabilities***, especially regarding PIN authentication and card skimming. Our project ***removes the dependency on physical cards*** by implementing a ***software-based authentication system***, reducing the risk of fraud. --- ##

2.2 Software-Based ATM Simulations ### *

2.2.1 What is the Work About?* Software-based ATM simulators are programs that replicate ATM functionalities for educational or testing purposes. These simulators allow users to ***log in, check balances, withdraw or deposit funds***, and manage accounts without physical hardware. ### *

2.2.2 How It Works?* These systems are ***developed using programming languages*** like ***C, Python, or Java***. They utilize: - ***File handling or database management*** to store user credentials and transaction details. - ***Menu-driven interfaces*** for user interaction. - ***Basic authentication mechanisms***, such as username-password or PIN-based verification. ### *

2.2.3 Importance and Applications* - Used for ***learning purposes*** in software development and financial computing. - Helps banks ***test new ATM features*** before implementing them in physical machines. - Provides ***a risk-free environment*** for students to practice programming concepts related to banking systems. ### *

2.2.4 Drawbacks and Limitations* - Many simulators ***lack strong security measures***, making them vulnerable to unauthorized access. - ***No real-time banking integration***, limiting their use beyond training purposes. - File-based storage is ***less secure than database-based systems***. ### *

2.2.5 Criticism & Link to Project Motivation* Many ATM simulations ***do not prioritize security***, making them unrealistic compared to real banking systems. Our project aims to ***enhance authentication security***, potentially integrating ***multi-factor authentication or encryption*** in future versions. --- ## *

2.3 Security in ATM Systems* ### *

2.3.1 PIN-Based Authentication* ####
What is the Work About? PIN-based authentication is the standard security mechanism in ATMs, requiring users to enter a ***Personal Identification Number (PIN)*** to verify their identity. ####
How It Works? - The user enters a PIN, which is compared against a ***securely stored encrypted PIN*** in the bank's database.

- If the PIN matches, access is granted; otherwise, multiple failed attempts may result in the account being locked.

Importance and Applications* - ***Simple and easy to implement***, making it the most widely used authentication method.

- Protects accounts from unauthorized access.

Drawbacks and Limitations* - ***Prone to theft*** through methods such as ***shoulder surfing*** and ***keylogging attacks***. - ***Weak PINs can be easily guessed***, leading to potential fraud. - ***Does not provide multi-layer security***, relying solely on a single factor for authentication. #### ***Criticism & Link to Project Motivation*** Given the vulnerabilities of ***PIN-based authentication***, our project could ***explore alternative security mechanisms*** such as ***biometric authentication or OTP-based verification*** to enhance security. --- ### *

2.3.2 File-Based vs. Database-Based User Authentication*

What is the Work About? ATM systems store user credentials and account balances using ***either file-based storage or databases***. **How It Works?*** - ***File-based storage:*** Information is stored in text files and accessed through programming logic. - ***Database-based storage:*** Data is managed using ***SQL or NoSQL databases***, providing structured security measures such as encryption.***Importance and Applications*** - ***File-based authentication*** is simple and easy to implement for small-scale applications. - ***Database-based authentication*** is used in real-world banking systems to ensure ***data integrity, security, and scalability***. ***Drawbacks and Limitations*** - ***File-based storage*** is less secure, as data can be easily modified or accessed without encryption. - ***Databases require additional infrastructure*** and may slow down the system if not optimized properly. ***Criticism & Link to Project Motivation*** Many ATM simulations rely on ***file-based storage***, making them vulnerable to security threats. Our project starts with file handling but could be ***enhanced with database integration and encryption*** for improved security. --- ##

2.4 Conclusion This chapter analyzed existing ATM systems, software-based simulations, authentication methods, and storage techniques. The study revealed ***security weaknesses in traditional ATMs and current simulations***, which our project aims to address by improving authentication mechanisms and storage security.

3. REQUIREMENT ANALYSIS

3.1 Project Requirements The project requires both ***hardware and software*** components to ensure smooth execution.

3.1.1 Hardware Requirements Although the project is a ***software-based ATM simulation***, minimal hardware is needed for development and testing: - ***Processor:*** Intel Core i3 or higher (for compiling and running the program efficiently). - ***RAM:*** At least 4GB (to handle code execution without lags). - ***Storage:*** 500MB free space (for storing source code, executables, and transaction data). - ***Keyboard & Monitor:*** Required for user input and program interaction. ### *

3.1.2 Software Requirements* The software requirements include the necessary tools for development, compilation, and execution: - ***Operating System:*** Windows (compatible with Windows.h header), Linux (with minor modifications). - ***Programming Language:*** C (due to its efficiency, low-level system control, and structured programming). - ***Compiler:*** GCC or MinGW for compiling the C program. - ***Code Editor:*** VS Code, Dev-C++, or Code::Blocks for writing and testing the code. The program uses ***file handling for data storage***, so no additional database management software is needed in the initial version. However, future enhancements may incorporate database integration. --- ## *

3.2 Feasibility Study* A feasibility study is conducted to evaluate the project's practicality, efficiency, and implementation challenges. ### ***3.2.1 Technical Feasibility*** The project is ***technically feasible*** as it uses fundamental C programming concepts like ***file handling, modular programming, and authentication mechanisms***. Since it does not require complex hardware or networking, it can be run on ***basic computers with minimal system resources***. ###

3.2.2 Economic Feasibility The project is ***cost-effective*** because:

- It does not require purchasing additional hardware.

- Open-source tools like ***GCC, Code::Blocks, and Dev-C++*** are used, eliminating software costs.

- No real banking network integration is needed, reducing operational expenses.

3.2.3 Operational Feasibility The project provides an ***easy-to-use interface*** where users can log in, register, and perform banking transactions. Its ***menu-driven structure*** ensures accessibility, even for users with limited technical knowledge.

3.2.4 Legal and Security Feasibility Since the project is ***a simulation and does not interact with real financial institutions***, there are no legal concerns regarding banking regulations. However, for real-world applications, ***data encryption and secure authentication methods*** would need to be implemented. --- ##

Conclusion The requirement analysis has established that the project can be successfully developed and executed on basic hardware and software configurations. The feasibility study confirms that the project is ***technically, economically, and operationally viable***, making it a practical educational tool for learning ATM functionalities in C programming.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

1. ***To develop a secure and user-friendly ATM simulation*** that allows users to log in, register, and perform basic banking transactions such as deposits, withdrawals, and balance inquiries. 2. ***To enhance understanding of C programming concepts*** by implementing modular programming, file handling, and authentication mechanisms in a real-world financial application.

Project Scope: - ***User Authentication:*** Secure login and registration system to prevent unauthorized access. - ***Basic Banking Transactions:*** Users can check balances, withdraw, and deposit money. - ***Admin Controls:*** Administrative functionalities for managing user accounts. - ***File Handling:*** Data storage and retrieval using files to maintain transaction records. - ***Menu-Driven Interface:*** Simple and interactive UI for ease of use. ### ***Applications:*** - ***Banking Simulations:*** Can be used for educational purposes to understand ATM operations. - ***Software Development Training:*** Helps students and developers learn C programming concepts like authentication, file handling, and modular programming. - ***Prototype for Real ATM Systems:*** Can serve as a foundation for more advanced banking software.

Report Organization:

1. ***Introduction:*** Provides an overview of the project, including its background, motivation, problem definition, objectives, and scope.

2. ***Literature Review:*** Discusses existing ATM systems, related technologies, and previous research or projects in this domain.

3. ***System Design and Methodology:*** Explains the architecture of the ATM system, module interactions, flowcharts, and algorithms used for user authentication and transactions.

4. ***Implementation:*** Details the coding approach, programming language (C), file handling techniques, and key functions of different modules (login, transactions, admin control).

5. ***Testing and Results:*** Describes test cases, system performance, error handling, and validation of functionalities through various user scenarios.

6. ***Conclusion and Future Scope:*** Summarizes project achievements, challenges faced, and potential enhancements such as database integration or security improvements.

2: LITERATURE REVIEW* This chapter presents an overview of existing works related to ***ATM systems and banking security***, analyzing their methodologies, technologies, applications, and limitations. The criticisms of these systems provide the foundation for the motivation behind this project. --- ## ***2.1 Traditional ATM Systems*** ### ***2.1.1 What is the Work About?*** Traditional ATMs allow users to conduct financial transactions such as cash withdrawals, deposits, balance inquiries, and fund transfers using a physical ATM card and a PIN for authentication. ### ***2.1.2 How It Works?*** ATMs use ***magnetic stripe or chip-based cards*** to store account information, which is read by a card reader. The system verifies the user's

identity through **PIN-based authentication** and processes transactions via a secure banking network, following protocols such as **ISO 8583** for communication between ATMs and banking servers. ### **2.1.3 Importance and Applications** - Provides **24/7 access to banking services**, reducing reliance on human tellers. - Enhances **banking convenience and accessibility**, allowing customers to perform transactions globally. - Facilitates **quick and secure cash withdrawals** without the need to visit a bank branch. ###

2.1.4 Drawbacks and Limitations - **Security risks**: Traditional ATMs are vulnerable to **card skimming, shoulder surfing, and PIN theft**. - **Dependency on physical cards**: Losing a card prevents access to funds and requires replacement. - **High maintenance costs**: ATMs require **regular servicing, security monitoring, and cash refilling**. ### *

2.1.5 Criticism & Link to Project Motivation Traditional ATMs have significant **security vulnerabilities**, especially regarding PIN authentication and card skimming. Our project **removes the dependency on physical cards** by implementing a **software-based authentication system**, reducing the risk of fraud. --- ##

2.2 Software-Based ATM Simulations ###

2.2.1 What is the Work About? Software-based ATM simulators are programs that replicate ATM functionalities for educational or testing purposes. These simulators allow users to **log in, check balances, withdraw or deposit funds**, and manage accounts without physical hardware. ### *

2.2.2 How It Works? These systems are **developed using programming languages** like **C, Python, or Java**. They utilize: - **File handling or database management** to store user credentials and transaction details. - **Menu-driven interfaces** for user interaction. - **Basic authentication**

mechanisms, such as username-password or PIN-based verification. ### **2.2.3 Importance and Applications** - Used for **learning purposes** in software

development and financial computing. - Helps banks ***test new ATM features*** before implementing them in physical machines. - Provides ***a risk-free environment*** for students to practice programming concepts related to banking systems. ### *

2.2.4 Drawbacks and Limitations* - Many simulators ***lack strong security measures***, making them vulnerable to unauthorized access. - ***No real-time banking integration***, limiting their use beyond training purposes. - File-based storage is ***less secure than database-based systems***. ### *

2.2.5 Criticism & Link to Project Motivation* Many ATM simulations ***do not prioritize security***, making them unrealistic compared to real banking systems. Our project aims to ***enhance authentication security***, potentially integrating ***multi-factor authentication or encryption*** in future versions. --- ## *

2.3 Security in ATM Systems* ###

2.3.1 PIN-Based Authentication #### ***What is the Work About?*** PIN-based authentication is the standard security mechanism in ATMs, requiring users to enter a ***Personal Identification Number (PIN)*** to verify their identity. #### ***How It Works?*** - The user enters a PIN, which is compared against a ***securely stored encrypted PIN*** in the bank's database.

- If the PIN matches, access is granted; otherwise, multiple failed attempts may result in the account being locked.

Importance and Applications - ***Simple and easy to implement***, making it the most widely used authentication method.

- Protects accounts from unauthorized access.

Drawbacks and Limitations - ***Prone to theft*** through methods such as ***shoulder surfing*** and ***keylogging attacks***. - ***Weak PINs can be easily guessed***, leading to potential fraud. - ***Does not provide multi-layer security***, relying solely on a single factor for authentication. #### ***Criticism & Link to Project Motivation*** Given the vulnerabilities of ***PIN-based authentication***, our

project could ***explore alternative security mechanisms*** such as ***biometric authentication or OTP-based verification*** to enhance security. --- ###

2.3.2 File-Based vs. Database-Based User Authentication #### ***What is the Work About?*** ATM systems store user credentials and account balances using ***either file-based storage or databases***. #### ***How It Works?*** - ***File-based storage:*** Information is stored in text files and accessed through programming logic. - ***Database-based storage:*** Data is managed using ***SQL or NoSQL databases***, providing structured security measures such as encryption. #### ***Importance and Applications*** - ***File-based authentication*** is simple and easy to implement for small-scale applications. - ***Database-based authentication*** is used in real-world banking systems to ensure ***data integrity, security, and scalability***. #### ***Drawbacks and Limitations*** - ***File-based storage*** is less secure, as data can be easily modified or accessed without encryption. - ***Databases require additional infrastructure*** and may slow down the system if not optimized properly. #### ***Criticism & Link to Project Motivation*** Many ATM simulations rely on ***file-based storage***, making them vulnerable to security threats. Our project starts with file handling but could be ***enhanced with database integration and encryption*** for improved security. --- ##

2.4 Conclusion This chapter analyzed existing ATM systems, software-based simulations, authentication methods, and storage techniques. The study revealed ***security weaknesses in traditional ATMs and current simulations***, which our project aims to address by improving authentication mechanisms and storage security. Would you like ***proper citations*** to be added to this section?

3: REQUIREMENT ANALYSIS* This chapter outlines the necessary ***hardware and software*** components for developing the ATM simulation system. It also includes a ***feasibility study*** to determine the practicality and effectiveness of the project in real-world applications. --- ## ***3.1 Project Requirements*** The project requires both ***hardware and software*** components to ensure smooth execution. ### ***3.1.1 Hardware Requirements*** Although the project is a ***software-based ATM simulation***, minimal hardware is needed for development and testing: -

Processor: Intel Core i3 or higher (for compiling and running the program efficiently). - **RAM:** At least 4GB (to handle code execution without lags). - **Storage:** 500MB free space (for storing source code, executables, and transaction data). - **Keyboard & Monitor:** Required for user input and program interaction. ### **3.1.2 Software Requirements** The software requirements include the necessary tools for development, compilation, and execution: - **Operating System:** Windows (compatible with Windows.h header), Linux (with minor modifications). - **Programming Language:** C (due to its efficiency, low-level system control, and structured programming). - **Compiler:** GCC or MinGW for compiling the C program. - **Code Editor:** VS Code, Dev-C++, or Code::Blocks for writing and testing the code. The program uses **file handling for data storage**, so no additional database management software is needed in the initial version. However, future enhancements may incorporate database integration. --- ## **3.2 Feasibility Study** A feasibility study is conducted to evaluate the project's practicality, efficiency, and implementation challenges. ### **3.2.1 Technical Feasibility** The project is **technically feasible** as it uses fundamental C programming concepts like **file handling, modular programming, and authentication mechanisms**. Since it does not require complex hardware or networking, it can be run on **basic computers with minimal system resources**. ### **3.2.2 Economic Feasibility** The project is **cost-effective** because:

- It does not require purchasing additional hardware.

- Open-source tools like **GCC, Code::Blocks, and Dev-C++** are used, eliminating software costs.

- No real banking network integration is needed, reducing operational expenses.

3.2.3 Operational Feasibility The project provides an **easy-to-use interface** where users can log in, register, and perform banking transactions. Its **menu-driven structure** ensures accessibility, even for users with limited technical knowledge. ### **3.2.4 Legal and Security Feasibility** Since the project is **a simulation and does not interact with real financial institutions**, there are no legal concerns regarding banking regulations. However, for real-world applications, **data encryption and secure authentication methods** would need to be implemented. ---

Conclusion The requirement analysis has established that the project can be successfully developed and executed on basic hardware and software configurations. The feasibility study confirms that the project is ***technically, economically, and operationally viable***, making it a practical educational tool for learning ATM functionalities in C programming.

4: SYSTEM ARCHITECTURE AND METHODOLOGY

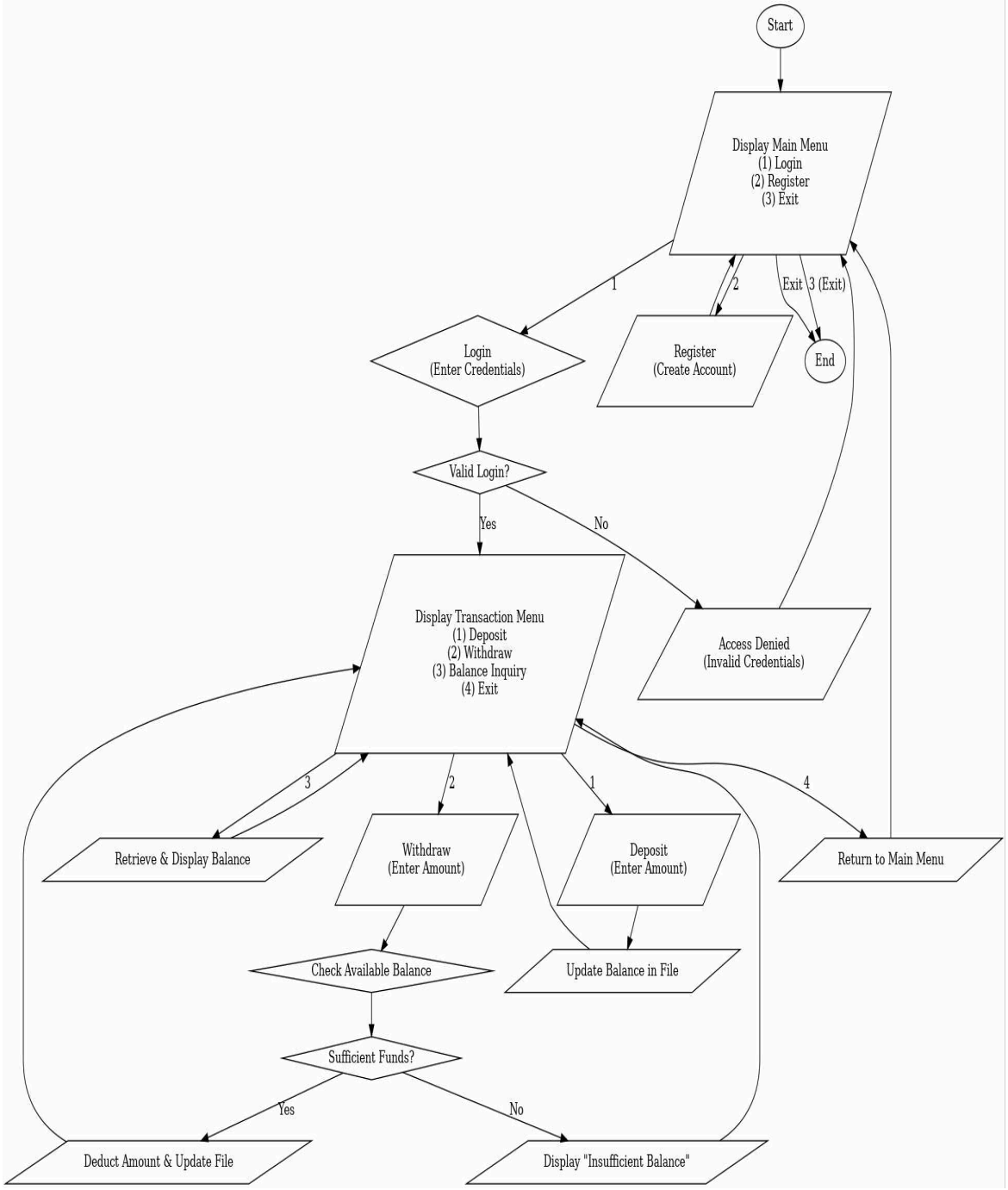
4.1 Block Diagram / System Architecture* The system is designed as a ***modular structure***, where different components handle specific functionalities such as ***user authentication, transactions, and administrative controls***. ###

4.1.1 System Overview The ATM simulation consists of the following key components: - ***User Interface (UI):*** A ***menu-driven system*** that allows users to select operations like login, registration, withdrawals, deposits, and balance inquiries. - ***Authentication Module:*** Verifies user credentials during login and ensures only authorized access. - ***Transaction Processing:*** Handles user operations like withdrawing money, checking balances, and updating account details. - ***File Management System:*** Stores and retrieves user account information and transaction history. - ***Admin Module:*** Provides control functions for managing users and monitoring transactions. ### *

4.1.2 System Block Diagram* Below is the conceptual block diagram of the system: ***(Insert Center Justified Figure)* *Figure 4.1: Block Diagram of ATM Simulation*** #### ***Description of Blocks:*** - ***User Input:*** Users interact with the system via a menu-based interface. - ***Authentication System:*** Verifies login credentials stored in files. - ***Transaction Handling:*** Executes deposits, withdrawals, and balance checks. - ***File Storage:*** Updates and retrieves account details. - ***Admin Controls:*** Provides access for managing users and overseeing transactions. Each component interacts ***sequentially***, ensuring a structured flow of operations. --- ##

4.2 Flowcharts / Algorithms The system follows ***structured programming*** using ***C*** to ensure modularity, security, and efficiency. The key processes are defined using ***flowcharts and algorithms***. ### *

4.2.1 User Login & Registration Flowchart* This process ensures that users can securely ***log in*** or ***register a new account*** before performing transactions.



: Flowchart of User Authentication* #### ***Algorithm:*** 1. ***Start*** 2. Display menu: ***(1) Login (2) Register (3) Exit*** 3. If ***Login:*** - Prompt user for ***username and password*** - Validate credentials from ***file storage*** - If valid, proceed to ***main menu*** - Else, display ***"Invalid credentials"*** message 4. If ***Register:*** - Prompt user for ***new account details*** - Store details securely in the file - Display

*"**Registration successful**"* message 5. If *Exit:* Terminate program 6. *End* --- ### *

4.2.2 Transaction Flowchart* This process handles user actions like ***depositing, withdrawing, and checking balances***. ***(Insert Center Justified Figure)*** ***Figure 4.3: Flowchart of Transactions*** #### ***Algorithm:*** 1. ***Start***

2. Display transaction menu:

- (1) Deposit - (2) Withdraw - (3) Balance Inquiry - (4) Exit 3. If ***Deposit:*** - Prompt user for deposit amount - Update balance in ***file storage*** 4. If ***Withdraw:*** - Check available balance - If sufficient, deduct amount and update file - Else, display *"**Insufficient balance**"* message 5. If ***Balance Inquiry:*** - Retrieve and display balance from file 6. If ***Exit:*** Return to main menu 7. ***End*** --- ###

4.2.3 File Handling for Data Storage Instead of using databases, the system employs ***file handling*** to store user details securely. #### ***Methods Used:*** - ***fopen(), fclose()*** → Open and close files. - ***fprintf(), fscanf()*** → Read/write user details. - ***fseek()*** → Locate and modify user records. --- ## ***Conclusion*** The system architecture is designed to be ***modular, secure, and efficient***, ensuring ***smooth user interactions and reliable transaction handling***. The use of ***flowcharts and algorithms*** ensures a clear structure for execution.

5. IMPLEMENTATION DETAILS

5.1 Hardware Components Since this project is a *software-based ATM simulation*, there are minimal hardware requirements. However, for real-world implementation, the following hardware components would be required:

5.1.1 Processor & Memory - *CPU:* A standard computer processor (Intel Core i3 or higher) is required to *run the software smoothly*. - *RAM:* At least 4GB of RAM ensures that the program executes efficiently without lags. ### *

5.1.2 Input/Output Devices* - *Monitor:* Displays the ATM interface (menu-driven system). - *Keyboard:* Used for user input (account number, PIN, transaction choices). ### *

5.1.3 Storage System* - In this project, user details and transactions are stored in *files* (using C file handling). - In a real-world ATM, a *secure database (MySQL, PostgreSQL)* would store customer data securely. --- ## *

5.2 Software Implementation* The project is implemented using *C programming* due to its efficiency, security, and control over system resources. ###

5.2.1 Key Software Components #### *
1. User Authentication Module* - Users can *log in* or *register* for an account. - The system checks credentials stored in a file and grants access upon *successful authentication*. #### *
2. Transaction Handling Module* - Users can *deposit, withdraw, and check their balance*.

- The system updates the file storage accordingly after each transaction.

3. File Management System

- User details and transactions are stored securely in files.

- Functions like *fopen(), fclose(), fprintf(), fscanf()* are used for file handling.

4. Menu-Driven Interface - The system provides an *easy-to-navigate menu* for users. - The program executes in a *loop* until the user decides to exit. --- ## *5.2.2 Interfacing & Protocols* Since the project is a *standalone software*, no external communication protocols are required. However, if integrated into an actual banking system, it would require: - *Secure Communication (SSL/TLS):* To protect sensitive user data. - *Database Integration:* To replace file storage with *SQL databases*. - *Banking APIs:* To connect with real financial systems. --- ## *

Conclusion* This chapter explains how the ***ATM simulation is implemented using C programming***, focusing on ***user authentication, transactions, and file management***. While currently file-based, it can be enhanced with ***databases and security protocols*** for real-world applications.

6. RESULTS AND ANALYSIS

6.1 Simulation Outputs This section presents the actual ***execution results*** of the project, including user interactions, transaction processing, and file updates. ### ***6.1.1 User Login & Registration Output*** #### ***Expected Behavior:*** - Users should be able to ***register*** a new account successfully.

- The system should validate login credentials before granting access.

Observed Output: When a user registers, the system prompts:

```
Enter your username: JohnDoe Enter your PIN: **** Registration successful!
```

When the user logs in:

```
Enter your username: JohnDoe Enter your PIN: **** Login successful!
```

If the PIN is incorrect:

```
Invalid credentials. Please try again.
```

(Insert Center Justified Figure)* *Figure 6.1: User Login & Registration Output --- ## ***6.2 Transaction Processing Output*** This section presents how deposits, withdrawals, and balance inquiries are executed. ### ***6.2.1 Deposit & Withdraw Outputs*** #### ***Deposit Output:***

```
Enter deposit amount: 500 Deposit successful! Your new balance is: $1500
```

Withdrawal Output:

```
Enter withdrawal amount: 700 Withdrawal successful! Your remaining balance is: $800
```

If funds are insufficient:

```
Insufficient balance. Transaction failed.
```

6.2.2 Balance Inquiry Output

```
Your current balance is: $800
```

(Insert Center Justified Figure)* *Figure 6.2: Transaction Processing Output --- ## ***6.3 Error Analysis & Validation*** ### ***6.3.1 Error Sources:*** - ***File Handling Errors:*** Data corruption may occur if files are not properly closed. - ***Incorrect User Input:*** If the user enters invalid inputs, the program may fail unless error handling is properly implemented.

7. FUTURE ENHANCEMENT

While the current ATM simulation project successfully handles ***user authentication, transactions, and file-based storage***, there are several technical improvements that can be implemented to enhance its functionality, security, and scalability. --- ## ***1. Database Integration* *Current System:*** Uses ***file handling*** to store user data and transactions. ***Enhancement:*** Replace file-based storage with a ***database (MySQL, PostgreSQL, or SQLite)*** for:

- Faster and more reliable data retrieval.
- Secure storage with encryption for user credentials.
- Support for multiple concurrent users.

2. GUI-Based ATM Interface* *Current System: Uses a ***text-based menu*** for user interaction. ***Enhancement:*** Implement a ***Graphical User Interface (GUI)*** using: - ***Python (Tkinter/PyQt), Java (JavaFX), or C# (WinForms/WPF)*** for an intuitive interface. - A ***touchscreen-compatible*** design for better user experience. --- ##

3. Online Banking & Mobile App Integration* *Enhancement: Extend the ATM functionality to support: - ***Online banking portals*** to check balances and transactions remotely. - ***Mobile applications*** for account management and virtual ATM transactions. - ***API integration*** to connect with real banking systems securely. --- ##

4. Multi-Factor Authentication (MFA) for Security* *Current System: Only uses a ***username and PIN*** for authentication. ***Enhancement:*** Implement ***MFA*** for better security: - ***OTP (One-Time Password) via SMS or email.* - *Biometric Authentication (Fingerprint or Face Recognition).* - *Security Questions for Additional Verification.*** --- ##

5. Blockchain-Based Transactions* *Enhancement: Implement ***blockchain technology*** for: - ***Immutable transaction records.*** - ***Secure decentralized transaction storage.*** - ***Smart contracts*** for automated banking operations. --- ##

6. AI-Powered Fraud Detection System* *Enhancement: Use ***AI and Machine Learning*** to: - Detect ***suspicious withdrawal patterns.*** - Prevent ***fraudulent transactions.*** - ***Flag unusual activities*** (e.g., multiple failed login attempts). --- ## *****

7. Cardless ATM Transactions* *Enhancement:* Implement a ***cardless withdrawal system*** where users can: - Use ***QR codes*** on their mobile apps to withdraw cash. - Authenticate via ***NFC (Near-Field Communication)*** instead of an ATM card. --- ## ***Conclusion*** By integrating ***databases, AI, biometrics, and online banking***, the project can evolve into a ***fully functional and secure ATM system***. These enhancements will improve ***user experience, security, and banking efficiency***.

8. CONCLUSION

The ATM simulation project successfully demonstrates the core functionalities of an ***automated teller machine***, including ***user authentication, deposits, withdrawals, and balance inquiries***. Implemented in ***C programming***, the project uses ***file handling*** for data storage and a ***menu-driven interface*** for user interactions. The system ensures basic transaction security, handles input validation, and provides a functional simulation of real-world ATM operations. However, the project has limitations, such as ***lack of database integration, absence of GUI, and limited security features***. Future improvements, including ***database storage, multi-factor authentication, AI-powered fraud detection, and online banking integration***, can enhance its scalability and real-world applicability. Overall, the project provides a ***strong foundation*** for understanding ATM operations and can be further extended into a ***fully functional banking system*** with additional enhancements.

9. APPENDICES

It may contains the additional topics or data sheets or reference sheets or even user manual. The appendix name should be given as fillows.

Appendix A: Circuit Diagram

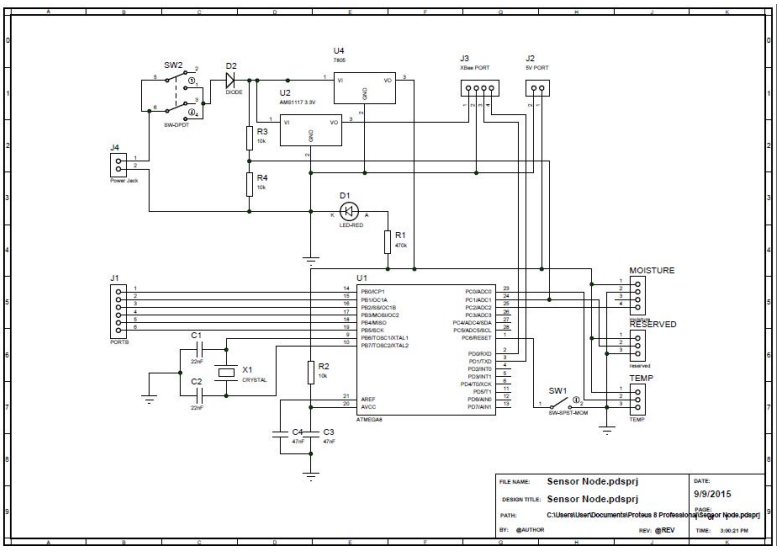


Figure 9-1: Circuit Diagram of AVR with sensors and XBee

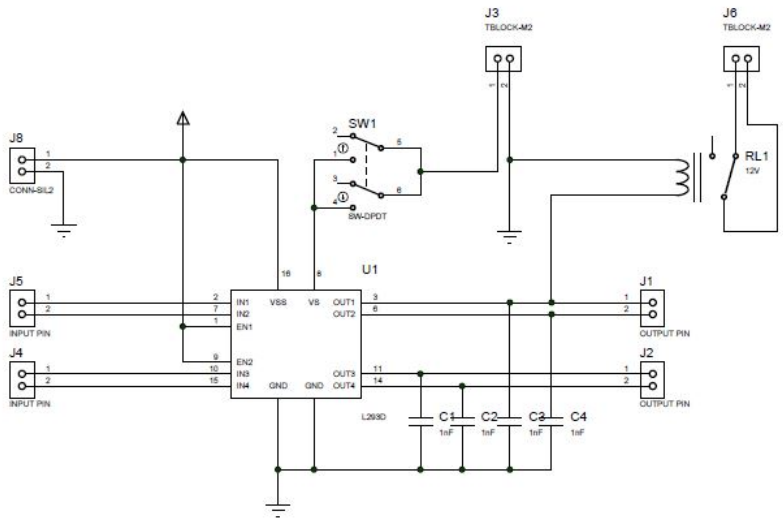


Figure 9-2: Circuit Diagram of Motor Driver and Relay

Appendix B: PCB Diagram

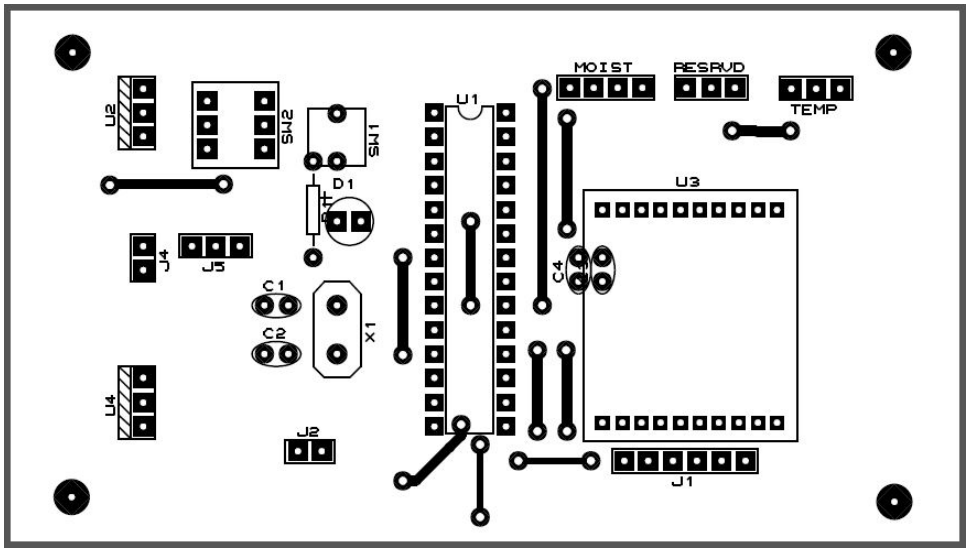


Figure 9-3: PCB Diagram of AVR with sensors and Xbee (Top Layer)

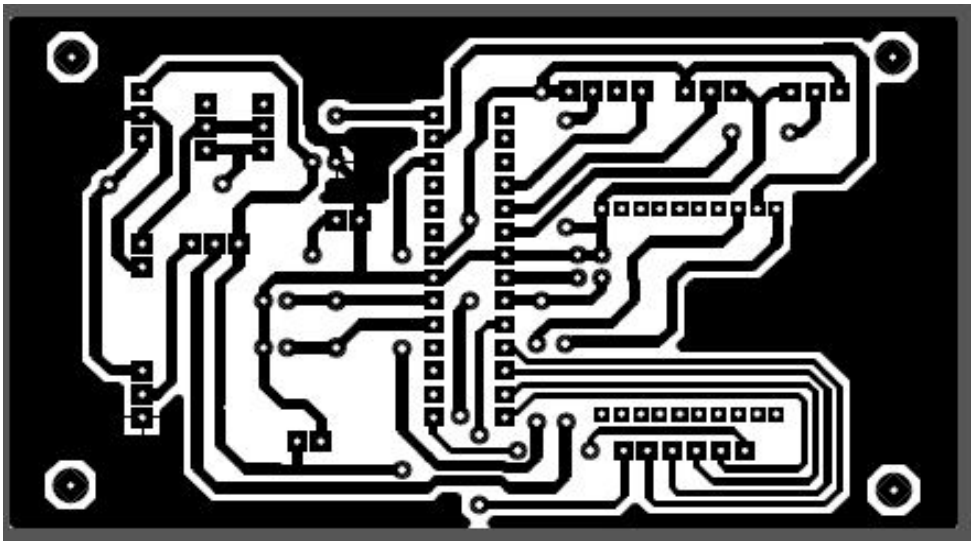


Figure 9-4: PCB Diagram of AVR with sensors and Xbee (Bottom Layer)

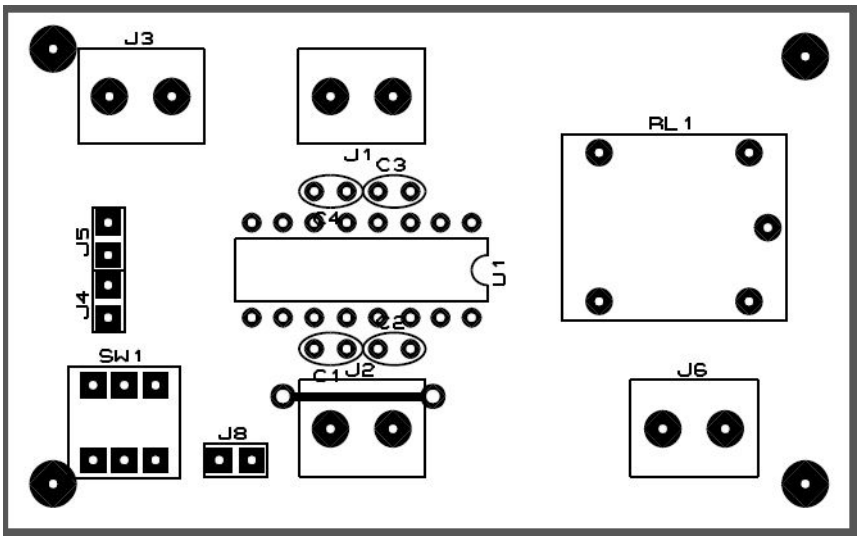


Figure 9-5: PCB Diagram of Motor Driver and Relay (Top Layer)

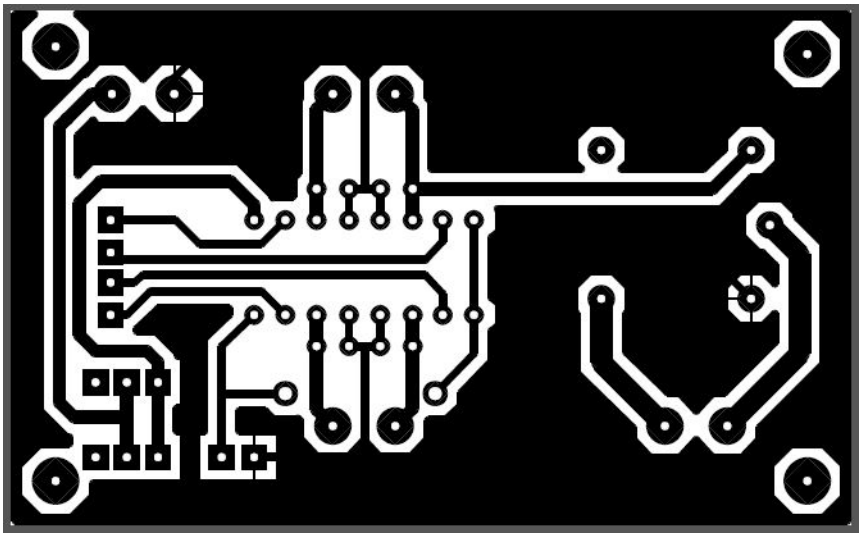


Figure 9-6: PCB Diagram of Motor Driver and Relay (Bottom Layer)

Appendix C: Zigbee Module Specification

Table 9-1: XBee S2 Pin Description [4]

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / CONFIG	Input	UART Data In
4	DIO8*	Output	Digital Output 8
5	RESET	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	DTR / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	CTS / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / SLEEP	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	RTS / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

* Function is not supported at the time of this release

Table 9-2: XBee S2 General Features [4]

Specification	XBee
Performance	
Indoor/Urban Range	Up to 100 ft (30 m)
Outdoor RF line-of-sight Range	Up to 300 ft (90 m)
Transmit Power Output (software selectable)	1mW (0 dBm)
RF Data Rate	250,000 bps
Serial Interface Data Rate (software selectable)	1200 bps - 250 kbps (non-standard baud rates also supported)
Receiver Sensitivity	-92 dBm (1% packet error rate)
Power Requirements	
Supply Voltage	2.8 – 3.4 V
Transmit Current (typical)	45mA (@ 3.3 V)
Idle / Receive Current (typical)	50mA (@ 3.3 V)
Power-down Current	< 10 µA
General	
Operating Frequency	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)
Operating Temperature	-40 to 85° C (industrial)
Antenna Options	Integrated Whip, Chip or U.FL Connector, RPSMA Connector

Appendix D: Used Linux Commands [3]

General Commands

- *apt-get update*: Updates your version of Raspbian.
- *apt-get upgrade*: Upgrades all of the software packages you have installed.
- *clear*: Clears the terminal screen of previously run commands and text.
- *Data*: Prints the current date.
- *date -s "3 Sep 2015 10:18:00":* Reset time and date
- */etc/init.d/ntp start*: restart npt.
- *find / -name example.txt*: Searches the whole system for the file example.txt and outputs a list of all directories that contain the file.
- *nano example.txt*: Opens the file example.txt in “Nano”, the Linux text editor.
- *poweroff*: To shutdown immediately.
- *raspi-config*: Opens the configuration settings menu.
- *reboot*: To reboot immediately.
- *shutdown -h now*: To shutdown immediately.
- *shutdown -h 01:22*: To shutdown at 1:22 AM.
- *startx*: Opens the GUI (Graphical User Interface).

File/Directory Commands

- *cat example.txt*: Displays the contents of the file example.txt.
- *cd /abc/xyz*: Changes the current directory to the /abc/xyz directory.
- *cp XXX*: Copies the file or directory XXX and pastes it to a specified location; i.e. *cp examplefile.txt /home/pi/office/* copies examplefile.txt in the current directory and pastes it into the /home/pi/ directory. If the file is not in the current directory, add the path of the file's location (i.e. *cp /home/pi/documents/examplefile.txt /home/pi/office/* copies the file from the documents directory to the office directory).
- *ls -l*: Lists files in the current directory, along with file size, date modified, and permissions.
- *mkdir example_directory*: Creates a new directory named example_directory inside the current directory.
- *mv XXX*: Moves the file or directory named XXX to a specified location. For example, mv *examplefile.txt /home/pi/office/* moves examplefile.txt in the current directory to the /home/pi/office directory. If the file is not in the current directory, add the path of the file's location (i.e. *cp /home/pi/documents/examplefile.txt /home/pi/office/* moves the file from the documents directory to the office directory). This command can also be used to rename files (but only within the

same directory). For example, *mv examplefile.txt newfile*.txt renames examplefile.txt to newfile.txt, and keeps it in the same directory.

- *rm example.txt*: Deletes the file example.txt.
- *rmdir example_directory*: Deletes the directory example_directory (only if it is empty).
- *scp user@10.0.0.32:/some/path/file.txt*: Copies a file over SSH. Can be used to download a file from a desktop/laptop to the Raspberry Pi.
- *touch*: Creates a new, empty file in the current directory.

Networking/Internet Commands

- *ifconfig:* To check the status of the wireless connection you are using (to see if wlan0 has acquired an IP address).
- *iwconfig*: To check which network the wireless adapter is using.
- *iwlist wlan0 scan*: Prints a list of the currently available wireless networks.
- *iwlist wlan0 scan | grep ESSID*: Use grep along with the name of a field to list only the fields you need (for example to just list the ESSIDs).
- *nmap*: Scans your network and lists connected devices, port number, protocol, state (open or closed) operating system, MAC addresses, and other information.
- *ping*: Tests connectivity between two devices connected on a network. For example, ping 10.0.0.32 will send a packet to the device at IP 10.0.0.32 and wait for a response. It also works with website addresses.
- *wget http://www.website.com/example.txt*: Downloads the file example.txt from the web and saves it to the current directory.

System Information Commands

- *cat /proc/meminfo*: Shows details about your memory.
- *cat /proc/partitions*: Shows the size and number of partitions on your SD card or hard drive.

- *cat /proc/version*: Shows you which version of the Raspberry Pi you are using.
- *df -h*: Shows information about the available disk space.
- *df /*: Shows how much free disk space is available.
- *dpkg –get-selections | grep XXX*: Shows all of the installed packages that are related to XXX.
- *dpkg –get-selections*: Shows all of your installed packages.
- *free*: Shows how much free memory is available.
- *hostname -I*: Shows the IP address of your Raspberry Pi.
- *lsusb*: Lists USB hardware connected to your Raspberry Pi.
- *vcgencmd measure_temp*: Shows the temperature of the CPU.
- *vcgencmd get_mem arm && vcgencmd get_mem gpu*: Shows the memory split between the CPU and GPU.

References

(Use IEEE format as follows. Note: This is auto generated reference, do not try to type manually. Should be left justified.)

[1] B. A., "Wireless Sensor Networks in Precision Agriculture," June 2005.

[2] C. Ayday and S. Safak, "Application of Wireless Sensor Networks with GIS on the Soil Moisture Distribution Mapping," January 2009.

[3] Circuitbasics.com, "Useful Raspberry Pi Commands," [Online]. Available: http://www.circuitbasics.com/useful-raspberry-pi-commands/. [Accessed Aug 2015].

[4] Digi.com, "Documentation for XBee ZB (S2B) Modules," [Online]. Available: http://ftp1.digi.com/support/documentation/90000976_W.pdf. [Accessed Jun 2015].

[5] P. Wood, "Raspberry Pi and Custard for Schools," [Online]. Available: http://www.rs-online.com/designspark/electronics/blog/raspberry-pi-and-custard-

for-schools. [Accessed 02 Sept 2015].

[6] B. Corporation, "Broadcam BCM2835 ARM Peripherals," Feb 2012. [Online]. Available: http://www.farnell.com/datasheets/1521578.pdf. [Accessed 2015 Jun 05].

[7] Raspberrypi.org, "GPIO: RASPBERRY PI MODELS A AND B," [Online]. Available: https://www.raspberrypi.org/documentation/usage/gpio/. [Accessed Jun 2015].

[8] J. E. K. B. a. S. K. Vongsagon Boonsawat, "XBee Wireless Sensor Networks for Temperature," 2010.

GUIDELINES

- Language must be English Language.
- You must use **"Times New Roman"** font with **"Bold"** and *"Italic"* variants whenever needs (except in exported image/diagrams, screenshot of tables etc.).
- For Cover page (First page of this document) and Title page (Second page of this document), use the font size and style as indicated in First page and Second page.
- From page number 'i' to end of the report, use proper headings and paragraph styles.
- For HEADING 1, use font size 12, style **BOLD**, UPPERCASE, Spacing (before: 0, after: 12, Line Spacing: 1.5).
- For HEADING 2, use font size 12, style **BOLD**, Spacing (before: 0, after: 10, Line Spacing: 1.5).
- For HEADING 3, use font size 12, style **BOLD**, Spacing (before: 0, after: 8, Line Spacing: 1.5).
- For HEADING 4, use font size 12, style **BOLD**, Spacing (before: 0, after: 6, Line Spacing: 1.5).

- For HEADING 5, use font size 12, style **BOLD**, Spacing (before: 0, after: 6, Line Spacing: 1.5) **and so on.**
- For Normal Paragraph, use font size 12, Spacing (before: 0, after: 18, Line Spacing: 1.5).
- You can use font size 11 or 12 for labeling the figure and table.
- All Headings must be Left Justified.
- All normal paragraphs must be left and right justified that is fully justified.
- All the tables and figures must be centered justified.
- Page size must be standard A4 size.
- Page margin must be 1.5" at left and all other of 1".
- Page number must be 0.5" from the bottom and centered at the bottom of the page
- Prefatory pages (before INTRODUCTION) should be numbered in Arabic numbers (i, ii, iii…) and body part should be numbered in style (1, 2, 3…)
- You can start the numbering from title page but **must not number the cover page**.
- For extensive table of contents and list of figures, you can use 1 line spacing instead of 1.5 (**For extensive Table of contents and list of figures only, otherwise use 1.5 line spacing as normal paragraph)**
- All the figures, tables, charts, equations etc. used in the reports must be numbered using insert caption under reference of MS Word.
- Reference must be used in ascending order (It means if you use [1] reference in page 5 and reference [2] in page 6 then you must not use reference [3] in page 4) using IEEE format.
- Reference should not be numbered as chapter.