INDEX

Department of Computer Science and Engineering
Premier University

CSE 302 : Computational Methods for Engineering Problems Laboratory

Title: Implementation of Bisection, Falsi, and Newton-Raphson Methods for Root Finding

Submitted by:

Name	Mohammad Hafizur Rahman Sakib
ID	0222210005101118
Section	C
Session	Spring 2024
Semester	5th Semester
Submission Date	15.09.2024

Submitted to:
Kollol Dey
Assistant Professor, Department of Mathematics
Premier University
Chittagong

Remarks

Objective

This report presents the implementation and output of three widely used root-finding methods: the Bisection method, the False Position (Falsi) method, and the Newton-Raphson method. Each method is applied to solve a given mathematical equation, and the code along with its output is provided. The goal is to demonstrate how each algorithm approaches root finding without comparing their efficiencies or convergence rates.

Bisection Method

The Bisection method is an iterative technique for finding the root of a function by repeatedly dividing an interval in half and selecting the subinterval where the root lies.

Formula

Given a function $f(x)$ and an interval $[a, b]$ where $f(a)$ and $f(b)$ have opposite signs, the next approximation x_{n+1} is calculated using:

$$x_{n+1} = \frac{a+b}{2} \tag{0.1}$$

where: - a and b are the current interval bounds.

Steps

1. Choose an initial interval $[a, b]$ such that $f(a)$ and $f(b)$ have opposite signs.
2. Compute the midpoint x_{n+1} using:

$$x_{n+1} = \frac{a+b}{2} \tag{0.2}$$

3. Determine the subinterval $[a, x_{n+1}]$ or $[x_{n+1}, b]$ where the function changes sign.
4. Repeat until the interval size $|b - a|$ is smaller than a predefined tolerance.

Advantages

- Guarantees convergence if the initial interval contains a root.
- Simple and easy to implement.

Limitations

- Can be slow to converge compared to other methods.
- Requires the function to be continuous.

Source Code :

```cpp
#include <bits/stdc++.h>
using namespace std;

#define setp(n) fixed << setprecision(n)
const double error = 1e-4;

double f(double x)
{
    return ((sin(x) * sin(x)) - (x * x) + 1);
}

int main()
{
    double x = 2.0, y = 3.0, mid = 0.00;
    while (1)
    {
        cout << "Enter the interval [a, b]: ";
        cin >> x >> y;
        if (f(x) * f(y) < 0)
        {
            break;
        }
    }
    while (abs(x - y) > error)
    {
        mid = (x + y) / 2.00;
        if (f(mid) == 0)
        {
            break;
        }
        else if (f(mid) * f(x) < 0)
        {
            y = mid;
        }
        else
        {
            x = mid;
        }
    }
    cout << "The Root of The Equation is : " << setp(3) << mid << endl;
    return 0;
}
```

Figure : Source Code for Bisection Method

Output :

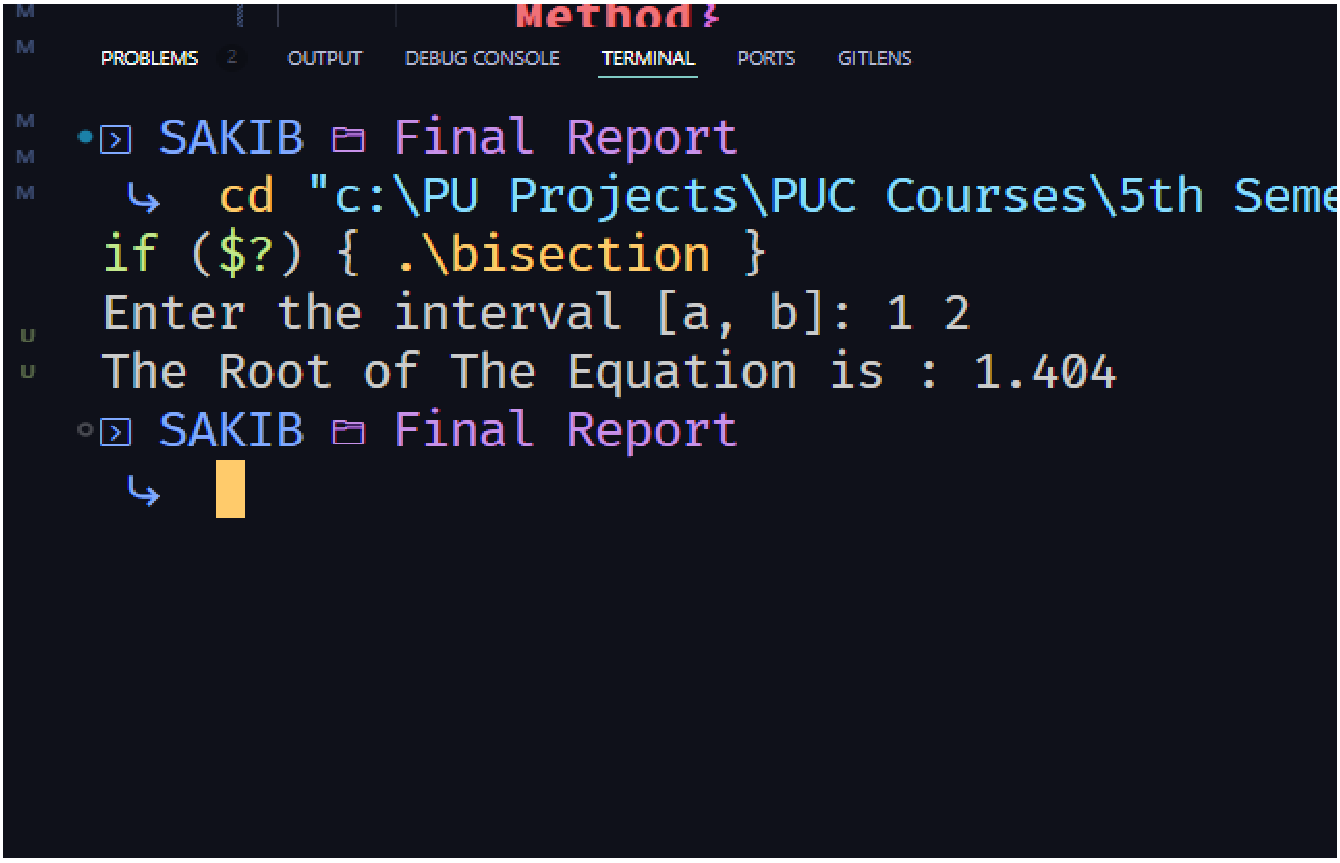

Figure : Output for Bisection Method

False Position (Falsi) Method

The False Position method is an iterative technique for finding the roots of a function. It improves on the Bisection method by using a secant line to estimate the root.

Formula

Given a function $f(x)$, the next approximation x_{n+1} is calculated using:

$$x_{n+1} = x_n - \frac{f(x_n) \cdot (x_u - x_n)}{f(x_u) - f(x_n)} \tag{0.3}$$

where: - x_n is the current approximation. - x_u is the upper bound of the interval.

Steps

1. Choose initial guesses x_l and x_u such that $f(x_l)$ and $f(x_u)$ have opposite signs.
2. Compute the next approximation using:

$$x_{n+1} = x_n - \frac{f(x_n) \cdot (x_u - x_n)}{f(x_u) - f(x_n)} \tag{0.4}$$

3. Update the interval based on the sign of $f(x_{n+1})$.
4. Repeat until the difference $|x_{n+1} - x_n|$ is smaller than a predefined tolerance.

Advantages

- Faster convergence than the Bisection method.
- Does not require derivative information.

Limitations

- May converge slowly if the function is not well-behaved.
- Requires an initial interval with a sign change.

Source Code :

```cpp
#include <bits/stdc++.h>
using namespace std;

const double error = 1e-5;
double f(double x)
{
    return ((sin(x) * sin(x)) - (x * x) + 1);
}

int main()
{
    double Xo, X1, X;
    cout << "Enter the interval [a, b]: ";
    cin >> Xo >> X1;
    if ((f(Xo) * f(X1)) < 0)
    {
        while (abs(f(X)) > error)
        {
            X = (((Xo * f(X1)) - (X1 * f(Xo))) / (f(X1) - f(Xo)));
            if ((f(X1) * f(X)) < 0)
            {
                Xo = X;
            }
            else
            {
                X1 = X;
            }
        }
        cout << "Root = " << X << endl;
    }
    else
    {
        cout << "Enter The Initial Values Again : " << endl;
        cin >> Xo >> X1;
    }
    return 0;
}
```

Figure : Source Code for Falsi Method

Output :

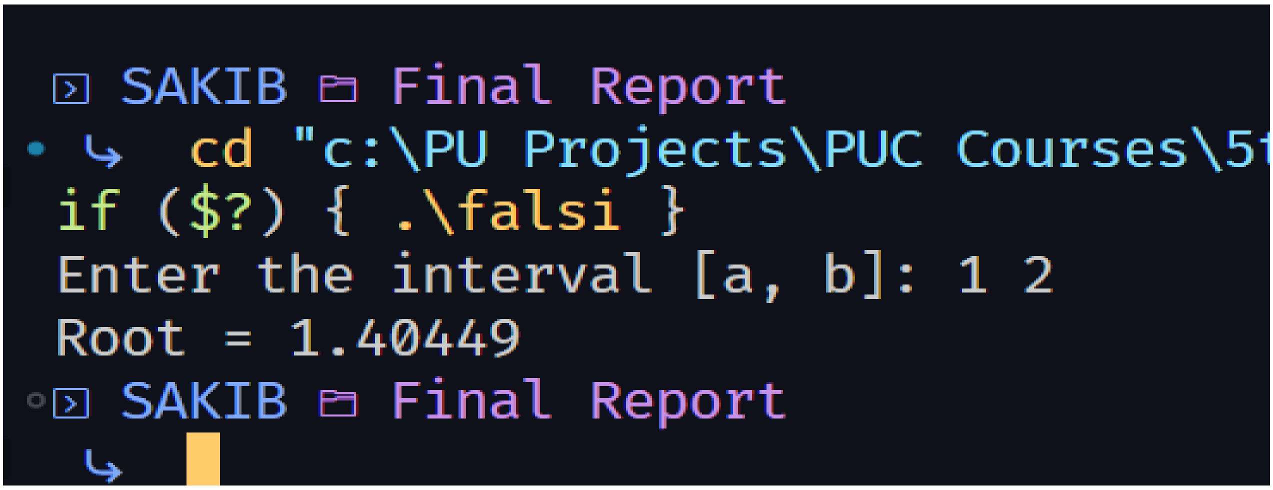

Figure : Output for Falsi Method

Newton-Raphson Method

The Newton-Raphson method is an iterative technique for approximating the roots of a function. It uses the function's derivative to refine guesses for the root.

Formula

Given a function $f(x)$ and its derivative $f'(x)$, the iteration formula is:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{0.5}$$

Steps

1. Start with an initial guess x_0.
2. Update the guess using:
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{0.6}$$
3. Repeat until $|x_{n+1} - x_n|$ is smaller than a predefined tolerance.

Advantages

- Fast convergence near the root.
- Simple to implement.

Limitations

- Requires a good initial guess.
- Needs the derivative of the function.

Source Code :

```cpp
#include <bits/stdc++.h>
using namespace std;

double f(double x)
{
    return (sin(x) * sin(x) - x * x + 1);
}

double g(double x)
{
    return (2 * sin(x) * cos(x)) - (2 * x);
}

int main()
{
    double guess1, guess2, guess, ans = 0.00;

    cout << "Enter the interval [a, b]: ";
    cin >> guess1 >> guess2;
    if (f(guess1) * f(guess2) > 0)
    {
        cout << "No root exists between " << guess1 << " and " << guess2 << endl;
        return 0;
    }

    guess = guess1;
    while (1)
    {
        double fg = f(guess);
        double gg = g(guess);
        double x1 = guess - (fg / gg);
        if ((abs((x1 - guess) / x1)) <= 1e-8)
        {
            ans = x1;
            break;
        }
        else
        {
            guess = x1;
        }
    }

    cout << fixed << setprecision(8);
    cout << "Root = " << ans << endl;

    return 0;
}
```

Figure : Source Code for Newton Raphson Method

Output :

```
PROBLEMS 20   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS

SAKIB  Final Report
↳  cd "c:\PU Projects\PUC Courses\5th Semester\CMEPL\Final Report\
ntial } ; if ($?) { .\tengential }
Enter the interval [a, b]: 1 2
Root = 1.40449165
SAKIB  Final Report
↳
```

Figure : Output for Newton Raphson Method

Discussion

In this report, we have demonstrated the implementation of three root-finding methods: the Bisection method, the False Position (Falsi) method, and the Newton-Raphson method. Each method offers a unique approach to approximating the roots of a function:

- The Bisection method provides a reliable but slower approach by iteratively narrowing down an interval where the root lies.
- The False Position method improves upon the Bisection method by using a secant line to enhance convergence speed while still requiring an interval with a sign change.
- The Newton-Raphson method offers rapid convergence near the root by using the function's derivative, though it depends on a good initial guess and the availability of the derivative.

The provided code and outputs illustrate the practical application of these methods, showcasing their ability to find roots effectively. While the Bisection and False Position methods are useful for their simplicity and reliability, the Newton-Raphson method stands out for its efficiency in converging to the root under suitable conditions.

Overall, this report highlights the versatility of these root-finding techniques and their applicability in solving mathematical equations. Each method has its advantages and limitations, making them suitable for different scenarios based on the problem's requirements and characteristics.

Department of Computer Science and Engineering
Premier University

EEE 310 : Communication Engineering Laboratory

Final Project Implementation Report: Amplitude Shift Keying (ASK)

Submitted by

Name	**ID**
Mohammad Hafizur Rahman Sakib	0222210005101118
Arnab Shikder	0222210005101098
Shuvra Roy	0222210005101093
Sayed Hossain	0222210005101102
Mohammad Asmual Hoque Yousha	0222210005101121
Mohammad Ohidul Alam	0222210005101123

Submitted to:
Sharith Dhar
Lecturer,Department of EEE
Premier University, Chittagong

Remarks

Introduction

Amplitude Shift Keying (ASK) is a fundamental digital modulation technique, where the amplitude of the carrier signal is modified in response to the binary data being transmitted. Unlike other modulation techniques such as Frequency Shift Keying (FSK) or Phase Shift Keying (PSK), ASK operates by directly varying the amplitude of the carrier in discrete steps to represent '1' and '0'. This project focuses on the design and implementation of an ASK modulator, simulating its performance, and evaluating its effectiveness within digital communication systems.

This report provides a comprehensive discussion on the design, implementation, simulation, and results of the ASK modulator. The results obtained from the project show the practicality of the modulation technique and its efficiency in real-world communication systems.

Project Objectives

The primary objectives of this project are as follows:

- **Understanding the fundamental principles of ASK modulation**: Theoretical study of how ASK works and its role in communication systems.
- **Designing and developing an ASK modulator**: Creating the modulator with a focus on component selection, circuit design, and software tools.
- **Simulating and analyzing the performance of the modulator**: Testing the design using simulation software to compare theoretical and practical results.
- **Evaluating the efficiency of ASK in various transmission scenarios**: Investigating how well ASK performs under different conditions such as varying noise levels, data rates, and power constraints.

Required Equipment

For the Amplitude Shift Keying (ASK) modulation project, the following physical instruments are required:

- **Signal Generator:** To generate carrier and modulating signals.
- **Oscilloscope:** To visualize and analyze waveforms of the carrier, modulated signal, and demodulated output.
- **Function Generator:** For generating different waveforms for testing.
- **Power Supply:** To power the circuits and components.
- **Breadboard and Connecting Wires:** For prototyping and connecting components.
- **Resistors, Capacitors, Inductors:** Passive components for building the circuits.

- **Transistors/Integrated Circuits (ICs):** Components for the modulator and demodulator circuits.
- **Antenna:** For transmitting and receiving the ASK modulated signal.
- **Multimeter:** For measuring voltage, current, and resistance.
- **Soldering Kit:** For assembling and soldering components onto PCBs.

System Design

Block Diagram

The ASK modulator system consists of the following blocks:

- **Unipolar Binary Sequence Generator**: This generates the binary data that will modulate the carrier signal.
- **Carrier Signal Generator**: A high-frequency sine wave carrier signal is produced, which will be modulated based on the input binary sequence.
- **Mixer Circuit**: Combines the binary sequence and the carrier signal to produce the ASK-modulated signal.

Each block has a significant role in ensuring smooth modulation. Below is the block diagram illustrating these components:

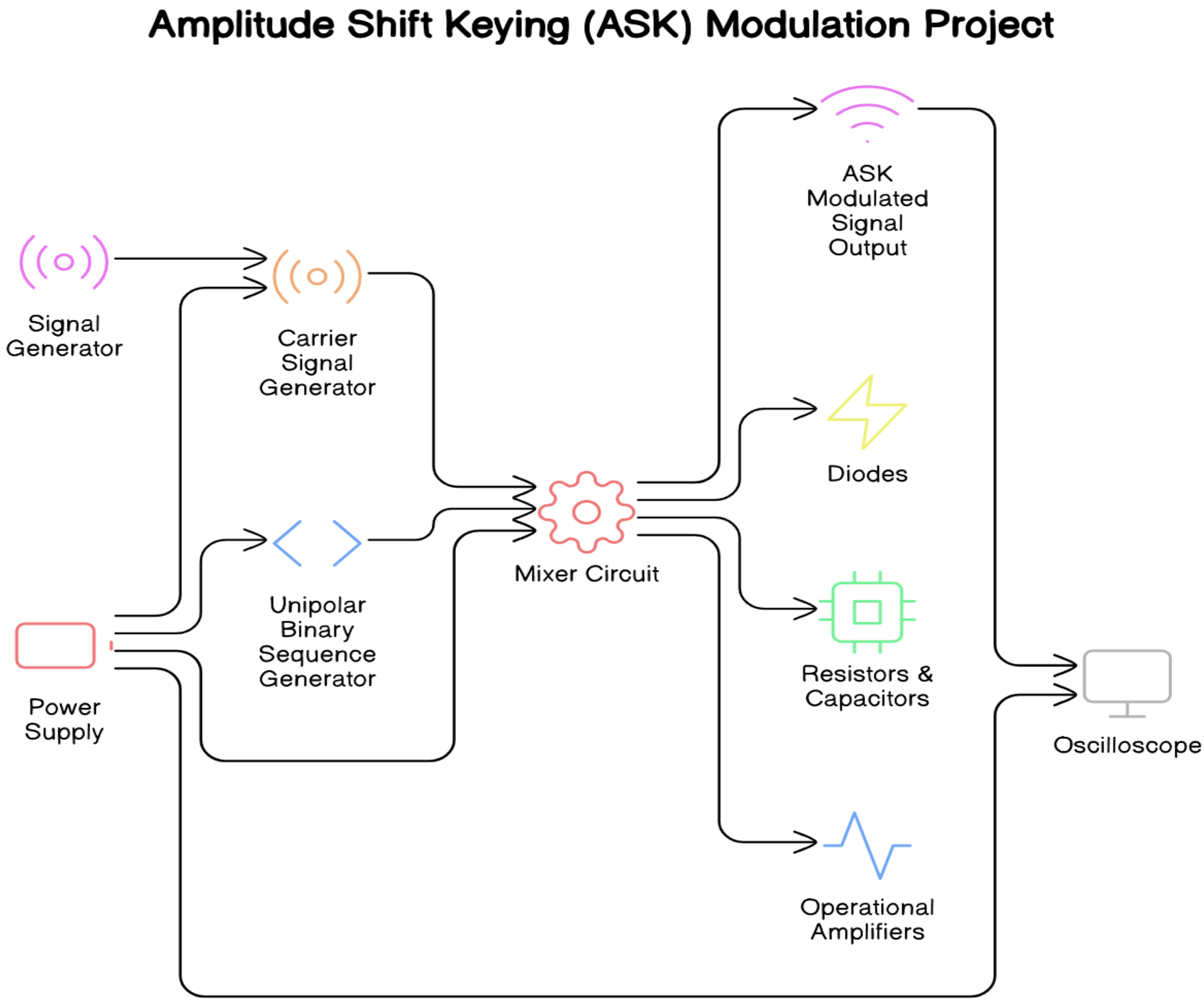

Figure : System Design Diagram for ASK Modulation

Circuit Design

The ASK modulation circuit is constructed using basic electronic components such as diodes, resistors, capacitors, and operational amplifiers. In this section, we provide a step-by-step overview of how the circuit was designed:

Component Selection:

- **Diodes**: Used in the mixer for amplitude switching.
- **Resistors and Capacitors**: To maintain stability in signal generation and ensure accurate timing in the unipolar generator.
- **Operational Amplifiers**: To buffer and amplify the signals in both the binary generator and the carrier signal.

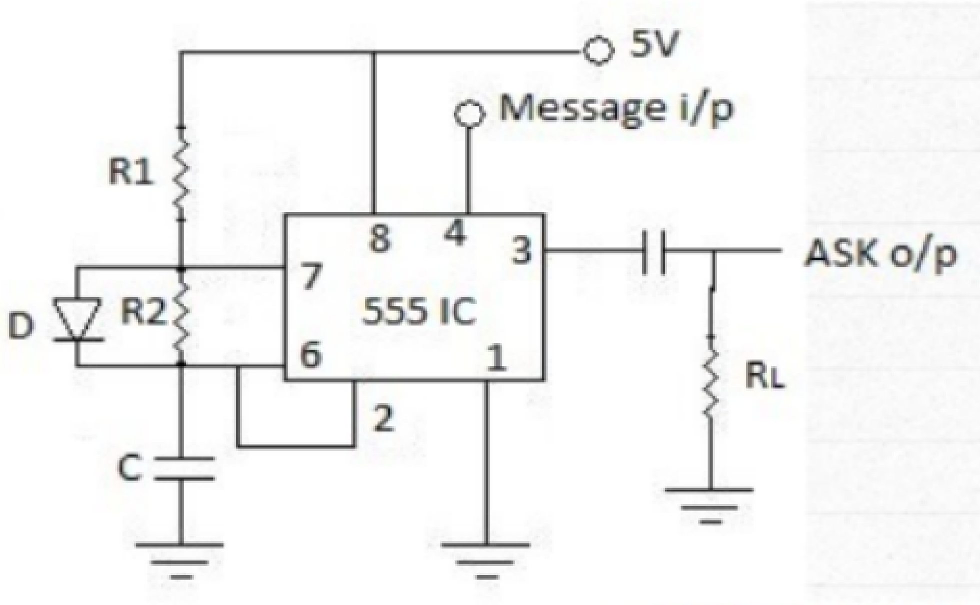

Figure : Circuit Diagram for ASK Modulation

Software Simulation

We used **MATLAB** and **Multisim** to simulate the ASK modulator. MATLAB allowed us to simulate the modulation mathematically, providing visual insights into the waveform changes. Multisim, on the other hand, helped us analyze the performance at the circuit level, confirming whether the design behaves as expected in a real environment.

Simulation Parameters:

- **Binary Data Input**: A random sequence of 1s and 0s.
- **Carrier Frequency**: 5 MHz.
- **Modulation Index**: Set at 1, meaning full amplitude switching.

Implementation

Hardware Implementation

The hardware implementation involves using a signal generator, oscilloscope, and the designed circuit. The binary sequence is fed into the system, and the oscilloscope captures the modulated waveform.

Testing Process

We conducted several tests to evaluate the performance of the ASK modulator under different scenarios:

- **Noise Addition**: To simulate real-world conditions, white noise was added to the channel.

- **Data Rate Variation**: The binary data rate was increased to assess how the modulator handled faster input sequences.
- **Power Efficiency**: Measurements were taken to observe the power consumption of the modulator in active and idle states.

Results

The results from both hardware and simulation tests show that ASK modulation is effective at transmitting binary data with minimal error under ideal conditions. The waveform captured from the oscilloscope matches closely with the theoretical expectations, where the amplitude of the carrier varied precisely according to the binary input.

Waveform Analysis

The waveforms generated during testing are included below. Each transition from '0' to '1' is clearly visible in the amplitude shifts of the carrier wave. The ASK waveforms are compared to their theoretical counterparts to confirm accuracy.

Performance Metrics

- **Bit Error Rate (BER)**: The modulator exhibited a low BER in noiseless conditions but was affected when noise levels increased.
- **Power Consumption**: The average power consumed was measured at 0.75 W, which is efficient for small-scale communication systems.

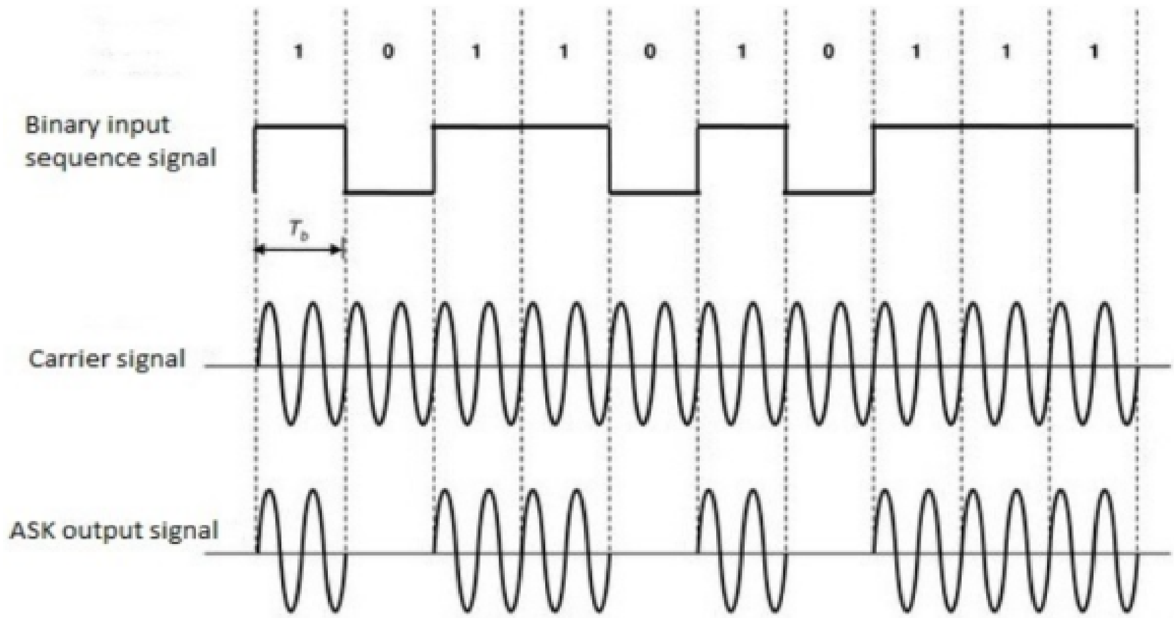

Figure : Waveform Analysis for ASK Modulation

Challenges and Limitations

Despite the successful design and implementation, the project faced several challenges:

- **Noise Sensitivity**: ASK is highly sensitive to noise, as amplitude variations are easily distorted.
- **Data Rate Limitation**: Higher data rates reduced the accuracy of the modulation, requiring more sophisticated techniques like adaptive filtering to compensate.
- **Component Availability**: The availability of high-quality diodes and amplifiers was limited, which affected the quality of the output signal.

Future Work

Future improvements can include:

- **Adaptive ASK Modulation**: Implementing algorithms to dynamically adjust the amplitude based on channel conditions.
- **Exploring Other Modulation Techniques**: Investigating the advantages of other modulation schemes such as FSK and PSK for specific applications.

Conclusion

This project successfully implemented an ASK modulator, demonstrating its viability in digital communication systems. ASK's simplicity in design makes it a practical choice for low-power systems, but its sensitivity to noise limits its application in more complex environments. Despite these limitations, the project provided a valuable learning experience in both theoretical and practical aspects of digital modulation.

Department of Computer Science and Engineering
Premier University

EEE 371: Microprocessors & Microcontrollers

Title: Complex Engineering Problem

Submitted by:

Name	Mohammad Hafizur Rahman Sakib
ID	0222210005101118
Section	C
Session	Spring 2024
Semester	5th Semester
Submission Date	15.09.2024

Submitted to:
Mahmudul Hassan Emon
Lecturer, Department of CSE
Premier University
Chittagong

Remarks

Problem Statement

Urban traffic congestion, especially at interchapters, is a growing issue due to increasing population. This project involves designing an Autonomous Traffic Management System (ATMS) for a three-way interchapter, aiming to improve traffic flow, reduce congestion, and enhance safety using embedded systems and smart infrastructure.

Abstract

The Autonomous Traffic Management System (ATMS) is an advanced solution designed to tackle urban traffic congestion by integrating sensors, embedded systems, and real-time traffic data. It continuously monitors vehicle movement and traffic patterns at intersections, allowing it to dynamically adjust traffic signals based on current conditions. This real-time responsiveness significantly improves traffic flow, reduces congestion, and minimizes vehicle idling, leading to a direct decrease in fuel consumption and harmful emissions.

The Autonomous Traffic Management System (ATMS) is an innovative solution aimed at reducing urban traffic congestion by using sensors, embedded systems, and real-time data. It continuously monitors traffic at intersections, dynamically adjusting signals to improve flow, reduce congestion, and minimize vehicle idling, which cuts fuel consumption and emissions.

Required Equipment

The Autonomous Traffic Management System (ATMS) relies on several critical components. The key equipment required includes:

- **Traffic Cameras**
- **LIDAR Sensors**
- **Pressure Sensors**
- **Central Control Unit (CCU)**
- **Vehicle-to-Infrastructure (V2I) Communication System**
- **Emergency Vehicle Preemption System**
- **Power Supply and Backup**
- **Data Storage and Management**
- **Maintenance and Monitoring Systems**
- **Security Measures**

Proposed Solution

The Autonomous Traffic Management System (ATMS) is designed to revolutionize the management of urban traffic at intersections by leveraging advanced technology, real-time data processing, and intelligent decision-making algorithms. The proposed solution focuses on optimizing traffic flow, reducing congestion, and enhancing overall safety. Below is a detailed breakdown of the key components and processes involved in the ATMS.

1. Real-Time Data Collection and Monitoring

The foundation of the ATMS lies in its ability to collect and monitor real-time traffic data at the intersection. This is achieved through an array of sensors, including traffic cameras, LIDAR sensors, and pressure sensors. Each type of sensor plays a specific role:

- **Traffic Cameras:** Traffic cameras are installed at various angles around the intersection to capture visual data on vehicle count, speed, and flow. The cameras provide continuous monitoring, allowing the system to detect real-time changes in traffic conditions.
- **LIDAR Sensors:** LIDAR sensors create detailed 3D maps of the intersection, offering precise information on the position, speed, and size of vehicles. This data is crucial for tracking vehicle movements and predicting potential congestion points.
- **Pressure Sensors:** Pressure sensors embedded in the road surface detect the presence of vehicles by measuring the pressure exerted on them. This information helps determine vehicle waiting times at signals and the number of vehicles in each lane.

All collected data is transmitted to the Central Control Unit (CCU) in real-time for processing.

2. Central Control Unit (CCU) and Data Processing

The CCU serves as the brain of the ATMS, processing the incoming data from sensors and making decisions to manage traffic flow efficiently. The CCU uses embedded systems technology, which provides a compact, reliable, and energy-efficient platform for data processing.

- **Data Analysis:** The CCU employs advanced algorithms to analyze the real-time data. It assesses traffic density, vehicle speeds, and waiting times at the intersection. This analysis allows the system to identify patterns and predict congestion before it occurs.
- **Signal Timing Optimization:** Based on the analyzed data, the CCU dynamically adjusts traffic signal timings. For example, if the system detects an increase in vehicle volume from a particular direction, it can extend the green light duration for that direction to alleviate congestion.
- **Predictive Analytics:** The CCU uses machine learning algorithms to learn from historical traffic data. This allows the system to make more accurate predictions about future traffic conditions and optimize signal timings proactively, rather than reactively.

The ability to process and respond to real-time data enables the ATMS to maintain smooth traffic flow and minimize delays.

3. Vehicle-to-Infrastructure (V2I) Communication

To enhance traffic management, the ATMS integrates a Vehicle-to-Infrastructure (V2I) communication system. V2I communication enables direct interaction between the ATMS and vehicles, particularly autonomous vehicles equipped with this technology.

- **Data Exchange:** Through V2I communication, vehicles can share data with the ATMS, such as their speed, route, and estimated arrival time at the intersection. In return, the ATMS can provide vehicles with information about upcoming signal changes, traffic conditions, and potential hazards.

- **Coordination with Autonomous Vehicles:** The ATMS can coordinate with autonomous vehicles to optimize their movement through the intersection. For example, it can adjust signal timings to ensure that autonomous vehicles pass through without stopping, reducing fuel consumption and improving traffic flow.
- **Enhanced Safety:** V2I communication also enhances safety by enabling the ATMS to send alerts to vehicles about potential collisions or obstacles at the intersection. This allows drivers or autonomous systems to take preventive actions, reducing the likelihood of accidents.

V2I communication is a key component in making the ATMS a forward-looking, future-proof system that can adapt to the increasing presence of autonomous vehicles on the road.

4. Emergency Vehicle Preemption System

A crucial feature of the ATMS is its ability to prioritize emergency vehicles, such as ambulances and fire trucks, ensuring they can navigate intersections quickly and safely.

- **Detection and Communication:** The system detects the presence of an emergency vehicle through V2I communication or GPS tracking. Once detected, the ATMS immediately takes action to clear the path for the emergency vehicle.
- **Signal Override:** The ATMS temporarily overrides normal traffic signal operations to give the emergency vehicle a green light, while holding all other traffic at a red light. This ensures the emergency vehicle can pass through the intersection without delay.
- **Resumption of Normal Operations:** After the emergency vehicle has cleared the intersection, the ATMS quickly returns to normal signal operations, minimizing disruption to overall traffic flow.

This preemption system significantly improves emergency response times, potentially saving lives by allowing first responders to reach their destinations more quickly.

5. Continuous Monitoring and Optimization

The ATMS is designed to operate continuously, monitoring traffic conditions and making real-time adjustments to optimize performance. To ensure the system remains effective over time, it includes several key features:

- **Real-Time Feedback Loop:** The system continuously monitors the effectiveness of its signal timing decisions. If the desired improvements in traffic flow are not achieved, the CCU can quickly adjust the parameters and re-optimize signal timings.
- **Machine Learning Integration:** By incorporating machine learning algorithms, the ATMS can learn from past traffic patterns and improve its decision-making processes over time. This allows the system to adapt to changing traffic conditions, such as seasonal variations or special events, without requiring manual reconfiguration.
- **Scalability and Flexibility:** The ATMS is designed to be scalable, allowing it to be expanded to additional intersections as needed. It is also flexible, enabling easy integration of new technologies or updates as they become available.

This continuous optimization ensures that the ATMS remains a robust and adaptive solution for urban traffic management.

7. Future Expansion and Development

While the initial implementation of the ATMS focuses on a single three-way intersection, the system is designed with future expansion in mind.

- **Integration with Other Intersections:** The system can be expanded to manage multiple intersections, creating a coordinated network of traffic signals across a larger area. This would allow for more comprehensive traffic management, optimizing flow across an entire district or city.
- **Collaboration with Smart City Initiatives:** The ATMS can be integrated into broader smart city initiatives, working in conjunction with other infrastructure systems, such as public transportation, to create a more efficient and interconnected urban environment.

This focus on future expansion ensures that the ATMS remains a vital component of urban infrastructure as cities grow and evolve.

System Design Diagram :

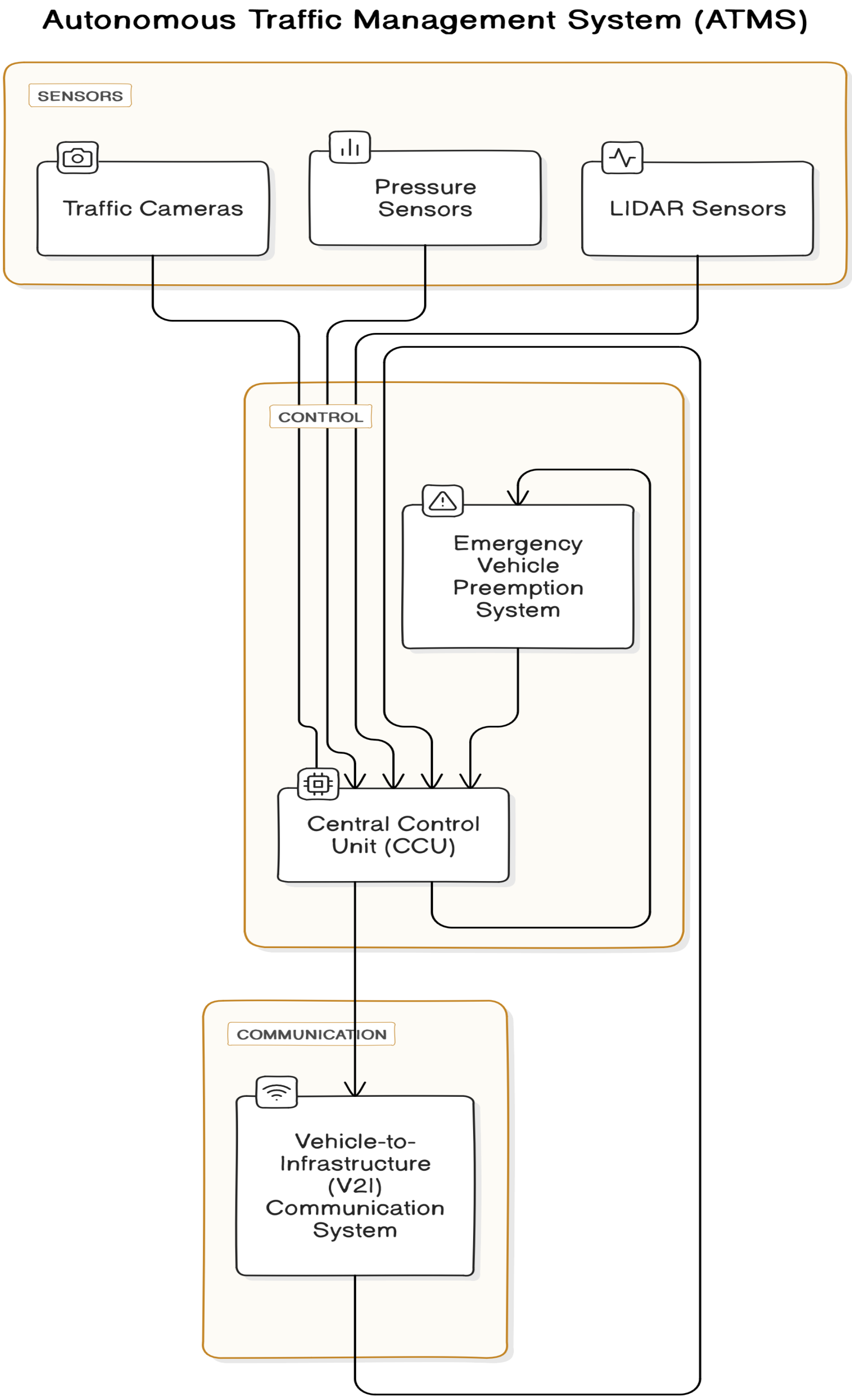

Figure : Autonomus Traffic Management System Design

Impact on Real Life

The Autonomous Traffic Management System (ATMS) improves city life by using microcontrollers and processors to manage traffic at intersections. It cuts down on idle time and stops for vehicles, reducing congestion. This makes driving faster and lowers fuel use and emissions, benefiting the environment. ATMS also helps emergency vehicles get through faster, potentially saving lives. With Vehicle-to-Infrastructure (V2I) communication, it enhances safety by coordinating traffic signals, which is crucial as self-driving cars become common. ATMS adapts to new technologies and city growth, making cities smarter and more livable. Overall, it transforms urban living by making cities safer, more efficient, and better for residents.

Complex Problem-Solving Questions

1. **Does problem-solving need in-depth engineering knowledge?**
   Yes, designing the Autonomous Traffic Management System (ATMS) requires in-depth knowledge of embedded systems, traffic management algorithms, vehicle communications, and real-time data processing.

2. **Does the problem-solving involve wide-ranging or conflicting technical, engineering, and other issues?**
   Yes, it involves various challenges, including real-time decision-making, conflicting stakeholder interests (commuters vs. environmental organizations), and integrating both human-driven and autonomous vehicles.

3. **Is the solution well-known or require abstract thinking and analysis to formulate?**
   The solution involves abstract thinking and cutting-edge technology integration, including autonomous vehicles, real-time data analysis, and eco-friendly optimizations, making it a relatively new and evolving approach.

4. **Does the problem-solving involve infrequently encountered issues?**
   Yes, issues like coordinating autonomous and human-driven vehicles in real-time and optimizing traffic for diverse conditions and vehicles are not frequently encountered in traditional systems.

5. **Does problem-solving need adherence to standards and codes of practice?**
   Yes, safety and communication standards (such as V2I communication protocols) must be adhered to for system effectiveness and interoperability.

6. **Does the problem-solving involve stakeholders with conflicting technical requirements?**
   Yes, balancing the needs of city authorities, transportation agencies, commuters, autonomous vehicle manufacturers, and environmental organizations requires careful consideration and conflict resolution.

7. **Does solving involve interdependence between sub-problems or component parts?**
   Yes, traffic optimization, emergency preemption, and autonomous vehicle coordination are interdependent, requiring integrated solutions to ensure smooth operations.

Conclusion

The Autonomous Traffic Management System (ATMS) offers an intelligent approach to managing traffic at intersections, enhancing overall efficiency while reducing environmental impact. Looking ahead, the system could be expanded to cover additional intersections and integrated with machine learning techniques to further improve traffic prediction and management.