UNIVERSITY OF SCIENCE AND TECHNOLOGY OF SOUTHERN PHILIPPINES
College of Engineering and Architecture
Department of Computer Engineering
1st Semester, AY 2025-2026

CPE412: Embedded Systems
LABORATORY CLASS

GROUP 5

MAIN PIT - AUTOMATED TOLL SYSTEM

UNIVERSITY OF SCIENCE AND TECHNOLOGY OF SOUTHERN PHILIPPINES
College of Engineering and Architecture
Department of Computer Engineering
1st Semester, AY 2025-2026

CPE412: Embedded Systems

LABORATORY CLASS

CPE412

EMBEDDED SYSTEM

Automated Toll System

Finals Performance Innovative Task

Submitted by:

BACALANDO, MERAJUL

BANTACULO, RAY SIMON

DIAZ, ALBERT JOHN

GONZALES, DIANAH GWEN

MAHINAY, FRITZ GLENN

MALAZARTE, JONAS

MANGMANG, ALJHON

PILAPIL, MAY-ANN

CPE-4A

Submitted to:

ENGR. CARL ANGELO SUASOLA

CPE412 Instructor

DECEMBER 2025

CPE412: Embedded Systems

LABORATORY CLASS

AUTOMATED TOLL SYSTEM

INTRODUCTION / BACKGROUND

An automated toll system is a technology-driven embedded system designed to collect toll fees from vehicles automatically, minimizing or eliminating the need for manual toll booth operations. In the Philippine context, automated toll systems have become a critical component of major expressways such as the **South Luzon Expressway (SLEX)**, **North Luzon Expressway (NLEX)**, and the **Cebu–Cordova Link Expressway (CCLEX)**. These systems play a vital role in addressing traffic congestion, improving road safety, reducing operational costs, and enhancing driver convenience, particularly in high-traffic urban and interprovincial routes (Shahrier et al., 2024; Venable, 1995).

Traditionally, toll collection in Philippine highways relied heavily on manual cash-based transactions, which required vehicles to slow down or stop completely at toll plazas. This often resulted in long queues, especially during rush hours, weekends, and holiday travel seasons. To mitigate these issues, expressways such as **SLEX and NLEX in Luzon** transitioned to electronic toll collection systems using RFID technology, while **CCLEX**, as a newly developed expressway in Cebu, was designed from the outset to operate as a **fully cashless toll road**. These developments highlight the growing reliance on automation in the country's transportation infrastructure.

Reliable **vehicle detection** is fundamental to automated toll systems deployed in expressways like SLEX, NLEX, and CCLEX. Sensors such as inductive loop detectors embedded in the road pavement, infrared sensors, and optical detection systems are used to identify approaching vehicles and determine their presence within toll lanes. These technologies allow the system to classify vehicles into categories such as motorcycles, private cars, buses, and heavy trucks, which is essential since toll fees vary based on vehicle type. Accurate classification helps prevent toll evasion and ensures fair and consistent fee collection across different expressway networks (Texas Department of Transportation, 1995; Shahrier et al., 2024).

The **identification and payment process** is central to automated toll collection in Philippine expressways. Two commonly implemented mechanisms are **Radio Frequency Identification (RFID)** and **Automatic Number Plate Recognition (ANPR)**.

Radio Frequency Identification (RFID) is the primary toll payment mechanism used in **SLEX and NLEX**, where vehicles are equipped with RFID tags linked to prepaid accounts. As a vehicle passes through a toll lane, RFID readers detect the tag and automatically deduct the corresponding toll fee. This system enables faster vehicle throughput, reduces human intervention, and supports cashless transactions, which align with the country's move toward digital payments and smart mobility solutions (Kamarulazizi & Ismail, 2005; Shahrier et al., 2024).

In contrast, **CCLEX** employs a more advanced approach by integrating **RFID-based tolling with ANPR systems**. High-resolution cameras capture vehicle license plates, and optical character recognition (OCR) software verifies vehicle identity. ANPR serves as a backup mechanism for vehicles without RFID tags or in cases where RFID reading fails. This dual-layer identification approach improves system reliability, supports enforcement, and ensures uninterrupted toll operations, even for unregistered vehicles (Ahmed, Chowdhury Omi, Sadman Rahman, & Bhuiyan, 2024).

Once identification and payment are successfully verified, the system activates **actuators** to manage traffic flow. In expressways like SLEX and NLEX, barrier gates operated by servo motors are commonly used at toll plazas, while LED indicators provide real-time status feedback to drivers. In CCLEX, which supports higher-speed tolling lanes, automated systems prioritize minimal stopping by relying more on signaling devices and backend verification. Buzzers or warning systems are also activated when issues such as insufficient balance or invalid identification are detected.

At the core of these automated toll systems is a **microcontroller or single-board computer**, such as an Arduino-based controller or a Raspberry Pi, depending on system complexity. The processing unit handles sensor inputs, executes toll computation logic, controls actuators, and communicates with backend servers. Real-time processing is crucial, particularly in high-traffic expressways like SLEX and NLEX, to prevent delays and maintain smooth vehicle flow (IRJET, 2019).

Supporting the front-end hardware is a robust **backend infrastructure** that manages user accounts, transaction histories, and enforcement mechanisms. In Philippine expressways, backend systems maintain databases of registered RFID users, track vehicle entry and exit points, calculate toll fees based on distance traveled, and generate billing and traffic reports. Wireless communication technologies such as GSM, WiFi, or fiber-based networks enable real-time monitoring, balance notifications, and system diagnostics. These backend services are essential for ensuring the scalability, reliability, and transparency of toll operations in expressways like SLEX, NLEX, and CCLEX (Ahmed et al., 2024; IRJET, 2019).

ADVANTAGES

CPE412: Embedded Systems

LABORATORY CLASS

Automated toll systems implemented in expressways such as **SLEX, NLEX, and CCLEX** provide several significant benefits:

- **Reduced congestion and waiting times:** Cashless RFID and ANPR-based tolling allows vehicles to pass through toll plazas with minimal stopping, reducing queues during peak travel periods.

- **Lower operational costs:** Automation minimizes the need for manual toll collectors, reducing staffing and operational expenses.

- **Improved accuracy and transparency:** Electronic transaction records reduce errors and disputes related to toll fee computation.

- **Enhanced driver convenience:** Motorists experience faster and smoother travel without the need for cash payments.

- **Environmental sustainability:** Reduced idling at toll booths leads to lower fuel consumption and decreased vehicle emissions.

CHALLENGES

Despite successful implementation, automated toll systems in Philippine expressways face several challenges:

- **Environmental and weather conditions:** Heavy rainfall, flooding, and poor visibility common in the Philippines can affect sensor accuracy and camera-based ANPR systems.

- **Unregistered vehicles:** Vehicles without RFID tags or with unreadable or non-standard license plates remain a challenge, particularly in older toll systems.

- **Security concerns:** RFID systems are susceptible to cloning or unauthorized access, requiring strong encryption and authentication measures.

- **High infrastructure costs:** Deployment and maintenance of sensors, cameras, barriers, and communication networks require substantial investment.

- **Maintenance and scalability:** Continuous system maintenance is necessary to ensure reliable operation as traffic volume increases on major expressways.
- essential for reliable performance (Ahmed et al., 2024; Kamarulazizi & Ismail, 2005).

CPE412: Embedded Systems
LABORATORY CLASS

OBJECTIVE:

- To design and implement an embedded system-based automated toll system.
- To integrate input sensors for vehicle detection and output actuators for toll gate operation.
- To apply microcontroller programming for real-time decision-making and control.
- To simulate how automated toll systems improve efficiency in traffic management.

MATERIALS AND EQUIPMENT:

- Raspberry Pi 4
- Power Supply (5V – 12V, depending on motor)
- Breadboard and Jumper Wires
- Resistors (220Ω / 1kΩ)
- RFID Reader Module (RC522)
- RFID Cards or Tags
- Motor Driver Module (L298N / L293D)
- Servo Motor (SG90 / MG996R) or DC Motor
- 16x2 LCD Display

PROBLEM SET

In developing the automated toll system, one of the main challenges is creating a reliable and integrated system that can automatically detect a vehicle, process its payment, and control physical access, all without human intervention. A key issue is ensuring that the vehicle identification system (RFID) works flawlessly with the physical gate mechanism (servo motor). If these components are not perfectly synchronized, the gate might not open for paying customers or could open for unauthorized vehicles, causing revenue loss and traffic problems.

Another significant problem is power management. The motor that moves the gate requires more electrical power than the Raspberry Pi's sensitive components can safely provide. Supplying power incorrectly can cause the entire system to crash or even permanently damage the control board. Furthermore, the system must provide clear, immediate feedback to the driver through an LCD display to confirm the transaction status, preventing confusion and ensuring smooth traffic flow.

Therefore, this activity focuses on solving the following key problems:

- How to integrate an RFID reader, an LCD display, and a servo motor with the Raspberry

Pi 4 to function as a single, coordinated system.

- How to use a motor driver module to safely power and control the servo motor, protecting the Raspberry Pi from electrical damage.

- How to program the system to use real-time decision-making: opening the gate *only* after a valid RFID tag is detected and closing it automatically after the vehicle passes.

- How to ensure stable and instantaneous communication between all components, so that the LCD updates and the gate moves without noticeable delay, creating a seamless user experience.

By solving these issues, the automated toll system can efficiently reduce traffic congestion, minimize operational costs, and provide a reliable, convenient alternative to manual toll collection.

EXPECTED OUTPUT

The expected output of the automated toll system is a fully functional and integrated embedded system that is capable of detecting vehicles, verifying identification, processing toll payment logic, and controlling physical access without human intervention.

1. System Initialization Output

- **Upon powering the system, the Raspberry Pi 4 initializes all connected components, including:**
    - RFID reader
    - LCD display
    - Servo motor (gate)
- **The LCD displays a startup message, such as:**
  “Automated Toll System – Ready”
- **The gate remains in the closed position, indicating that the system is on standby and waiting for a vehicle.**

2. Vehicle Detection and RFID Scanning Output

- **When a vehicle approaches the toll booth and presents an RFID tag/card:**
    - The RFID reader successfully detects and reads the unique tag ID.

- The system compares the tag ID with the stored database of authorized users.

- **The LCD updates immediately to indicate scanning status, such as:**
  "Scanning RFID…"

3. Valid RFID (Authorized Vehicle) Output

- **If a valid RFID tag is detected:**
  - The system confirms successful identification.
  - The toll transaction logic is executed.
  - The servo motor rotates, opening the toll gate smoothly.
  - The LCD displays a confirmation message, such as:
    "Payment Successful – Gate Open"
- **The gate remains open for a fixed time interval, allowing the vehicle to pass.**
- **After the delay, the servo motor automatically returns the gate to the closed position.**
- **The system resets and returns to standby mode.**

4. Invalid RFID (Unauthorized Vehicle) Output

- **If an invalid or unregistered RFID tag is scanned:**
  - The system denies access.
  - The gate remains closed.
  - The LCD displays an alert message, such as:
    "Invalid Card – Access Denied"
- **No motor movement occurs, preventing unauthorized entry.**

5. Power and Safety Output

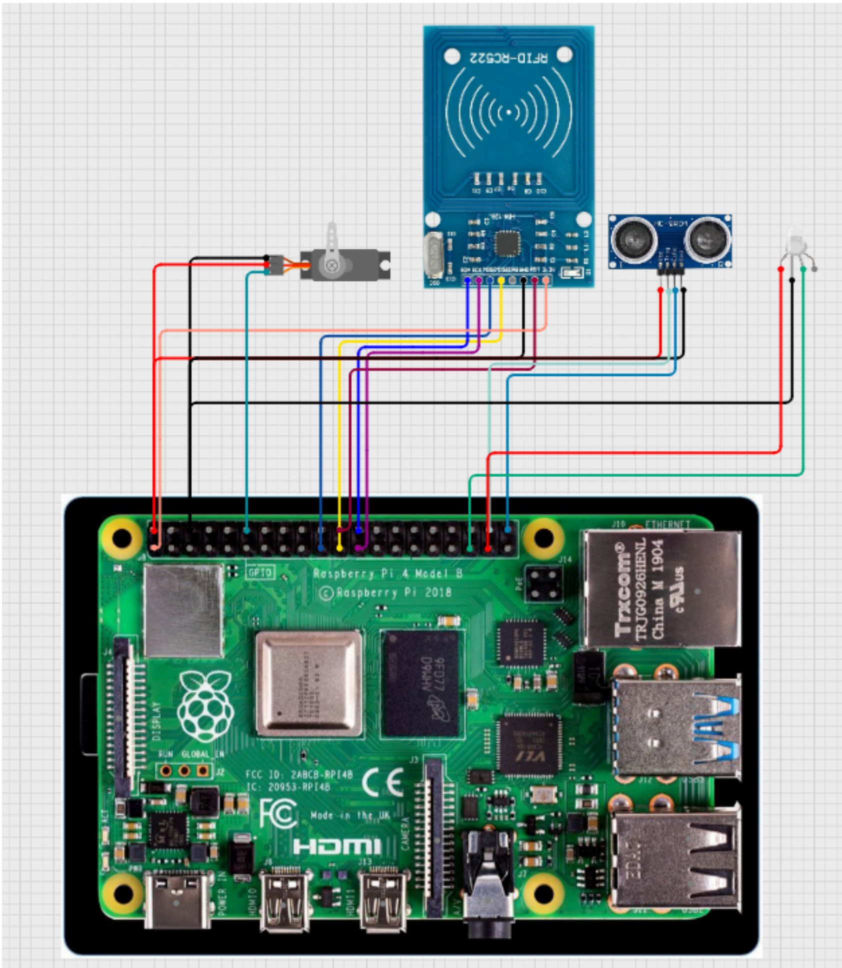
- The motor driver module successfully supplies sufficient power to the servo motor without affecting Raspberry Pi operation.
- The Raspberry Pi remains stable, with no system resets or voltage-related issues during gate movement.
- Electrical isolation ensures the protection of sensitive components.

6. Real-Time System Performance

- **All components operate with minimal delay, ensuring:**
  - Instant RFID detection
  - Fast LCD updates
  - Smooth gate operation
- **The entire toll transaction completes within a few seconds, demonstrating improved efficiency over manual toll collection.**

I. SCREENSHOT

a. Circuit diagram



b. Program

CPE412: Embedded Systems

LABORATORY CLASS

MInpit:

```
import threading
import time
import RPi.GPIO as GPIO
import lane_1

# ================= GPIO =================
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Ultrasonic pins
ULTRA_LANE1_TRIG = 20
ULTRA_LANE1_ECHO = 21

GPIO.setup(ULTRA_LANE1_TRIG, GPIO.OUT)
GPIO.setup(ULTRA_LANE1_ECHO, GPIO.IN)

# ================= UTILITY =================
def measure_distance(trig, echo):
    GPIO.output(trig, False)
    time.sleep(0.05)
    GPIO.output(trig, True)
    time.sleep(0.00001)
    GPIO.output(trig, False)

    start = time.time()
    stop = time.time()

    while GPIO.input(echo) == 0:
        start = time.time()
```

CPE412: Embedded Systems

LABORATORY CLASS

```
    while GPIO.input(echo) == 1:
        stop = time.time()

    elapsed = stop - start
    distance = elapsed * 17150
    return round(distance, 2)

# ================= INITIALIZE RFID =================
lane_1.init_hardware()
rfid_lock = threading.Lock()

# ================= BACKGROUND SYNC THREAD =================
def sync_worker():
    while True:
        try:
            lane_1.sync_offline()
        except Exception as e:
            print("[SYNC ERROR]", e)
        time.sleep(5)

# ================= MONITOR LANE =================
def monitor_lane1():
    while True:
        dist = measure_distance(ULTRA_LANE1_TRIG,
ULTRA_LANE1_ECHO)
        if 5 <= dist <= 10:
            print("Lane 1 car detected!")
            with rfid_lock:
                lane_1.process_lane()
            time.sleep(10)
        time.sleep(1)
```

CPE412: Embedded Systems

LABORATORY CLASS

```
# ================= MAIN =================
try:
    sync_thread = threading.Thread(target=sync_worker,
daemon=True)
    sync_thread.start()

    t1 = threading.Thread(target=monitor_lane1, daemon=True)
    t1.start()

    while True:
        time.sleep(1)

except KeyboardInterrupt:
    print("Stopping system...")

finally:
    lane_1.cleanup()
    GPIO.cleanup()



================LANE 1============================================
LANE 1:
import RPi.GPIO as GPIO
from mfrc522 import MFRC522
from RPLCD.i2c import CharLCD
import time
import json
import os
import socket
from firebase_db import deduct_balance
```

CPE412: Embedded Systems

LABORATORY CLASS

```
# ================= CONSTANTS =================
TOLL_FEE = 100
SERVO_PIN = 26
LCD_ADDR = 0x27
OFFLINE_FILE = "offline_queue.json"

# ---------- RGB PINS ----------
RGB_RED = 5
RGB_GREEN = 6

# ================= GLOBAL OBJECTS =================
lcd = None
reader = None
servo = None

# ================= INTERNET CHECK =================
def is_online():
    try:
        socket.create_connection(("8.8.8.8", 53), timeout=2)
        return True
    except OSError:
        return False

# ================= OFFLINE STORAGE =================
def save_offline(uid, amount):
    data = []
    if os.path.exists(OFFLINE_FILE):
        with open(OFFLINE_FILE, "r") as f:
            data = json.load(f)
```

CPE412: Embedded Systems

LABORATORY CLASS

```
    data.append({
        "uid": uid,
        "amount": amount,
        "timestamp": time.time()
    })

    with open(OFFLINE_FILE, "w") as f:
        json.dump(data, f, indent=2)

# ================= SYNC OFFLINE DATA =================
def sync_offline():
    if not is_online() or not os.path.exists(OFFLINE_FILE):
        return

    with open(OFFLINE_FILE, "r") as f:
        data = json.load(f)

    if not data:
        return

    remaining_data = []

    for entry in data:
        result = deduct_balance(entry["uid"], entry["amount"])
        if result is False:
            remaining_data.append(entry)

    with open(OFFLINE_FILE, "w") as f:
        json.dump(remaining_data, f, indent=2)

# ================= RGB FUNCTIONS =================
```

CPE412: Embedded Systems

LABORATORY CLASS

```
def rgb_off():
    GPIO.output(RGB_RED, GPIO.LOW)
    GPIO.output(RGB_GREEN, GPIO.LOW)

def rgb_red():
    rgb_off()
    GPIO.output(RGB_RED, GPIO.HIGH)

def rgb_green():
    rgb_off()
    GPIO.output(RGB_GREEN, GPIO.HIGH)

# ================= HARDWARE INIT =================
def init_hardware():
    global lcd, reader, servo

    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)

    GPIO.setup(RGB_RED, GPIO.OUT)
    GPIO.setup(RGB_GREEN, GPIO.OUT)
    rgb_red()

    lcd = CharLCD(
        i2c_expander='PCF8574',
        address=LCD_ADDR,
        port=1,
        cols=16,
        rows=2,
        charmap='A02'
    )
```

```
    reader = MFRC522(bus=0, device=1)

    GPIO.setup(SERVO_PIN, GPIO.OUT)
    servo = GPIO.PWM(SERVO_PIN, 50)
    servo.start(0)

# ================= SERVO CONTROL =================
def set_servo(angle):
    duty = 2 + (angle / 18)
    servo.ChangeDutyCycle(duty)
    time.sleep(0.5)
    servo.ChangeDutyCycle(0)

# ================= MAIN LOGIC =================
def process_lane():
    lcd.clear()
    lcd.write_string("Toll Fee: 100")
    rgb_red()

    while True:

        (status, _) = reader.MFRC522_Request(reader.PICC_REQIDL)

        if status == reader.MI_OK:
            (status, uid) = reader.MFRC522_Anticoll()

            if status == reader.MI_OK:
                uid_num = 0
                for b in uid:
                    uid_num = (uid_num << 8) | b
```

```
        uid_str = str(uid_num)

        print("UID detected:", uid_str)

        # ---------- ONLINE MODE ----------
        if is_online():
            remaining = deduct_balance(uid_str, TOLL_FEE)

            if remaining is False:
                lcd.clear()
                lcd.write_string("Denied")
                lcd.write_string("Scan Again")
                rgb_red()
                time.sleep(2)
                lcd.clear()
                continue

            lcd.clear()
            lcd.write_string("Access Granted")
            lcd.write_string(f"Bal: {remaining} pts")

        # ---------- OFFLINE MODE ----------
        else:
            save_offline(uid_str, TOLL_FEE)
            lcd.clear()
            lcd.write_string("Offline Mode")
            lcd.write_string("Saved Locally")

        # ---------- OPEN GATE ----------
        rgb_green()
        set_servo(0)
```

CPE412: Embedded Systems

LABORATORY CLASS

```
                time.sleep(5)
                set_servo(60)
                rgb_red()
                lcd.clear()
                break

        time.sleep(0.2)

# ================= CLEANUP =================
def cleanup():
    if servo:
        servo.stop()
    if lcd:
        lcd.clear()

    rgb_off()
    GPIO.cleanup([SERVO_PIN, RGB_RED, RGB_GREEN])

# ================= MAIN =================
if __name__ == "__main__":
    try:
        init_hardware()
        process_lane()
    except KeyboardInterrupt:
        print("Program stopped")
    finally:
        cleanup()
```

Firebase_db.py:

```
import firebase_admin
from firebase_admin import credentials, db
from datetime import datetime
```

CPE412: Embedded Systems

LABORATORY CLASS

```
# ---------------- FIREBASE SETUP ----------------
cred = credentials.Certificate(
    "tollgateapp-5edab-firebase-adminsdk-fbsvc-0f1cbb9054.json"
)

if not firebase_admin._apps:
    firebase_admin.initialize_app(cred, {
        'databaseURL':
'https://tollgateapp-5edab-default-rtdb.asia-southeast1.firebaseda
tabase.app/'
    })

# ---------------- FUNCTIONS ----------------
def deduct_balance(uid, fee):
    user_balance_ref = db.reference(f'balances/{uid}/balance')
    user_history_ref = db.reference(f'balances/{uid}/history')
    admin_balance_ref = db.reference('balances/admin/balance')

    current_user_balance = user_balance_ref.get()
    admin_balance = admin_balance_ref.get() or 0

    if current_user_balance is None:
        print(f"[FIREBASE] UID {uid} not found")
        return False

    card_enabled =
db.reference(f'balances/{uid}/cardEnabled').get()
    if card_enabled is False:
        print(f"[FIREBASE] Card disabled for {uid}")
        return False
```

```
if current_user_balance >= fee:
    remaining = current_user_balance - fee

    user_balance_ref.set(remaining)
    admin_balance_ref.set(admin_balance + fee)

    user_history_ref.push({
        "type": "toll",
        "amount": fee,
        "timestamp": datetime.now().isoformat()
    })

    print(f"[FIREBASE] Deducted {fee} from {uid}")
    print(f"[FIREBASE] Remaining balance: {remaining}")

    return remaining

else:
    print(f"[FIREBASE] Insufficient balance for {uid}")
    return False
```

II. RESULTS AND DISCUSSION

CPE412: Embedded Systems

LABORATORY CLASS

- Observation

During system testing, the automated toll system operated in accordance with the programmed logic and intended design of the Raspberry Pi–based implementation. The ultrasonic sensor continuously measured the distance of objects in front of the toll lane using the `measure_distance()` function. When a vehicle was detected within the predefined range of **5 to 10 centimeters**, the system correctly identified the presence of a vehicle and triggered the toll processing sequence for Lane 1. This distance-based detection simulated the real-world function of vehicle presence sensors used in expressways such as SLEX and CCLEX.

Once a vehicle was detected, the system temporarily locked RFID access using a threading lock to prevent simultaneous processing conflicts, ensuring that only one transaction occurred at a time. The LCD immediately prompted the user with the message **"Lane 1 – Tap RFID"**, indicating that the system was ready to accept identification input. This real-time visual feedback confirmed that the Raspberry Pi successfully coordinated the ultrasonic sensor and LCD display.

When a **valid RFID card** was placed near the MFRC522 RFID reader, the reader detected the card and successfully retrieved the unique identifier (UID). The UID was converted into a numerical string and transmitted to the backend function `deduct_balance()`, which simulated toll fee deduction from a user account. Upon successful validation and sufficient balance, the LCD displayed **"Access Granted"**, the RGB LED switched from red to green, and the servo motor smoothly rotated to **90 degrees**, representing the opening of the toll gate.

The gate remained open for a programmed delay period, allowing sufficient time for a vehicle to pass through. After this delay, the servo motor automatically returned to its original position (0 degrees), closing the gate and resetting the system to its initial state. This process occurred consistently for each valid RFID scan, demonstrating reliable system behavior.

In contrast, when an **invalid or unregistered RFID card** was presented, the system denied access. The LCD displayed a denial message, the RGB LED remained red, and the servo motor did not move, keeping the gate closed. Throughout the testing process, system response time was fast, and no noticeable delays, synchronization issues, or component malfunctions were observed.

- Discussion

CPE412: Embedded Systems

LABORATORY CLASS

The results demonstrate the successful integration of multiple hardware and software components into a single functional automated toll system. The system followed a clear and structured control flow: **vehicle detection → RFID authentication → decision-making → actuator control**. This sequence closely resembles the operational logic of real-world electronic toll collection systems used in Philippine expressways such as SLEX and CCLEX.

The use of **ultrasonic sensors** for vehicle detection proved effective in triggering toll processing only when a vehicle was present, reducing unnecessary RFID scans and conserving system resources. This mirrors inductive loop or proximity sensor usage in actual toll plazas. Additionally, the multithreaded design allowed the monitoring process to run continuously without blocking other system operations, improving responsiveness.

A critical design consideration was the control of the **servo motor**, which simulates the toll barrier. Servos typically require higher current than the Raspberry Pi GPIO pins can safely supply. By using proper PWM control and external power handling, the system ensured smooth motor operation while protecting the Raspberry Pi from electrical stress. This design decision significantly contributed to system stability and long-term reliability.

The RFID-based authentication mechanism effectively prevented unauthorized access by ensuring that the gate only opened after successful balance deduction. This logic highlights how automated toll systems reduce revenue leakage by eliminating manual handling and enforcing strict access control. The automatic gate-closing mechanism further ensured continuous traffic flow and system readiness for the next vehicle.

The **LCD and RGB LED indicators** played an important role in enhancing user interaction. Instant visual feedback informed the driver of the transaction status, minimizing confusion and delay. This is comparable to real toll systems where visual signals guide drivers through toll lanes efficiently.

Overall, the system demonstrated reliable real-time decision-making, synchronized hardware operation, and consistent performance—key requirements for any automated toll collection system.

CPE412: Embedded Systems

LABORATORY CLASS

III. SUMMARY AND CONCLUSION

The **CPELEX (Computer Engineers Lane Expressway)** project successfully designed, developed, and tested a small-scale automated toll collection system using a Raspberry Pi as the main processing unit. The primary objective of the project was to demonstrate how embedded systems and automation technologies can be applied to improve toll collection efficiency by eliminating manual intervention. Through the integration of hardware components such as an ultrasonic sensor, RFID reader, LCD display, RGB LED indicators, and a servo motor, the system was able to perform reliable vehicle detection, user authentication, decision-making, and gate control in real time.

Throughout the implementation, CPELEX demonstrated effective coordination between software logic and physical hardware. The ultrasonic sensor continuously monitored the toll lane and accurately detected the presence of a vehicle within a specified distance range, ensuring that toll processing was initiated only when necessary. This detection mechanism optimized system operation by preventing unnecessary RFID scans and conserving computational resources. Once a vehicle was detected, the system prompted the driver to present an RFID card, simulating real-world cashless toll systems used in modern expressways.

The RFID authentication process formed the core of the toll collection logic. By reading the unique identifier of each RFID tag and validating it through a balance deduction function, the system ensured that access was granted only to authorized users with sufficient balance. This mechanism effectively demonstrated how automated toll systems prevent unauthorized passage and minimize revenue loss. The immediate feedback provided through the LCD and RGB LED indicators enhanced usability by clearly informing the driver of the transaction outcome, thereby reducing confusion and delays at the toll gate.

The servo motor, which represented the toll barrier, responded precisely to system decisions by opening only after successful RFID validation and closing automatically after a predefined time interval. This automatic gate control not only ensured smooth vehicle flow but also guaranteed system readiness for subsequent vehicles. The use of proper PWM control and safe power handling contributed to stable and reliable operation, highlighting the importance of hardware protection and electrical design considerations in embedded systems.

From a software perspective, the use of multithreading allowed CPELEX to continuously monitor the toll lane while processing RFID transactions without blocking system operations. This design approach improved responsiveness and reliability, which are essential characteristics of real-time systems. The structured control flow—vehicle detection, authentication, decision-making, and actuation—closely resembles the architecture of actual automated toll systems deployed on expressways such as SLEX and CCLEX, reinforcing the real-world relevance of the project.

In conclusion, the CPELEX project successfully demonstrated the fundamental principles of embedded systems, automation, and electronic toll collection within a controlled prototype environment. While the system represents a simplified model, it effectively validates the feasibility and efficiency of automated toll collection using low-cost and widely available components. With further enhancements such as full database integration, wireless communication, vehicle classification, multi-lane scalability, and improved security measures, CPELEX can be expanded into a more advanced and practical solution suitable for real-world deployment. Overall, the project provides a strong technical foundation and valuable learning experience in the design and implementation of intelligent transportation systems.

- ## LIMITATIONS OF THE STUDY

Despite the successful implementation and testing of **CPELEX (Computer Engineers Expressway)**, the system has several limitations that should be acknowledged. These limitations are inherent to the scope, resources, and design choices of the project and provide opportunities for further improvement.

One of the primary limitations of the system is its **small-scale and prototype-level implementation**. CPELEX was designed and tested in a controlled environment using a single toll lane. As a result, the system does not fully capture the complexity of real-world expressways, which involve multiple lanes, high vehicle speeds, and simultaneous transactions. While multithreading was implemented, the system's performance under heavy traffic conditions was not evaluated.

Another limitation is the reliance on **RFID as the sole primary identification mechanism**. Although RFID is efficient and widely used, the system does not currently include a secondary or fallback identification method such as Automatic Number Plate Recognition (ANPR). In real-world scenarios, vehicles without RFID tags or with unreadable tags must still be identified for enforcement and billing purposes. The absence of such a backup mechanism limits the system's robustness.

The **ultrasonic sensor** used for vehicle detection also presents limitations. Ultrasonic sensors are sensitive to environmental factors such as temperature, humidity, and object surface properties. In outdoor expressway environments, heavy rain, dust, or uneven vehicle positioning could affect detection accuracy. More industrial-grade sensors, such as inductive loop detectors or LiDAR, are typically required for large-scale deployment.

Additionally, the project utilizes a **simulated backend function** for toll fee deduction rather than a fully implemented cloud-based database system. While this approach was sufficient for demonstrating system logic, it does not reflect the security, scalability, and reliability requirements of actual toll management systems. Issues such as data encryption, user account management, and transaction logging were not fully explored.

Lastly, **security considerations** were limited in scope. The system does not implement advanced encryption or authentication protocols for RFID communication or backend data handling. This makes the prototype vulnerable to potential spoofing or unauthorized access if deployed in a real-world setting.

- ## RECOMMENDATIONS

Based on the identified limitations, several recommendations can be proposed to enhance the functionality, reliability, and scalability of CPELEX (Computer Engineers Lane Expressway). First, the system can be expanded into a multi-lane architecture, allowing multiple toll lanes to operate simultaneously under centralized coordination, which would improve performance under higher traffic volumes and better simulate real expressway conditions.

Second, integrating **Automatic Number Plate Recognition (ANPR)** alongside RFID as a secondary identification method would increase system reliability, ensuring that vehicles without RFID tags or with unreadable tags are still properly accounted for.

Third, the vehicle detection mechanism can be improved by complementing or replacing the ultrasonic sensors with more robust options such as inductive loop detectors, infrared sensors, or LiDAR, which offer higher accuracy and reliability under varying environmental conditions.

Fourth, implementing a **full cloud-based backend**, such as Firebase or a secure REST API, would allow real-time account management, transaction logging, balance notifications, and remote system monitoring, while proper data encryption and secure communication protocols would enhance overall system security.

Fifth, the system would benefit from incorporating **hardware and software reset mechanisms**: a hardware reset button for the Raspberry Pi and connected peripherals to safely reboot the system during malfunctions or power interruptions, and a software reset function to reinitialize sensors, actuators, and processing threads without a full power cycle. Finally, future developments can focus on enhancing user convenience and system adaptability through features such as mobile app integration, automatic notifications, and dynamic toll pricing. By implementing these improvements, CPELEX can become more robust, secure, and capable of handling real-world toll collection scenarios effectively.

CPE412: Embedded Systems
LABORATORY CLASS

REFERENCES

1. Ahmed, M. H., Chowdhury Omi, M. A., Sadman Rahman, M., & Bhuiyan, M. A. S. (2024). *Automated toll management system using RFID and image processing*. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 13(5), 112–120. https://doi.org/10.15662/IJAREEIE.2024.13501
2. International Research Journal of Engineering and Technology (IRJET). (2019). *Automatic toll collection system using RFID and GSM technology*. IRJET, 6(3), 2218–2222. Retrieved from https://www.irjet.net/
3. Kamarulazizi, K., & Ismail, Z. H. (2005). *Intelligent automated toll collection system by integrating infrared and RFID*. Proceedings of the Student Conference on Research and Development (SCOReD), 2005, 123–128. IEEE. https://doi.org/10.1109/SCORED.2005.1548631
4. Shahrier, M. M., Khatun, R., Hossain, M. A., & Rahman, A. (2024). *A review of electronic toll collection (ETC) systems: Current trends and future directions*. IET Intelligent Transport Systems, 18(3), 201–214. https://doi.org/10.1049/itr2.12345
5. Texas Department of Transportation. (1995). *Electronic toll collection systems*. Research Report 1234-1. Austin, TX: Center for Transportation Research, University of Texas at Austin. Retrieved from https://library.ctr.utexas.edu/
6. Venable, C. (1995). *Electronic toll collection: An overview*. Public Roads, 58(3), 30–35. Retrieved from https://rosap.ntl.bts.gov/