EC 6020: EMBEDDED SYSTEM DESIGN

PROJECT REPORT

DESIGN AND IMPLEMENTATION OF AN NFC-BASED AUTOMATIC GATE OPENING SYSTEM WITH ULTRASONIC VEHICLE DETECTION AND OWNER-REQUESTED ACCESS.

Group CG10

Submitted by:
2020/E/100 - MUTHUGALA M.K.M.
2020/E/114 - PRAMUDITHA R.M.H.
2020/E/117 - PREMARANJAN D.

DEPARTMENT OF COMPUTER ENGINEERING

SUBMISSION: 07 AUGUST 2024

Contents

INTRODUCTION

In today's fast-paced world, automation plays a crucial role in enhancing security and convenience in our daily lives. Our project, the Smart Gate Opening System, exemplifies this by integrating modern technology into everyday activities. Automated systems reduce the need for manual intervention, increasing efficiency and reliability. We chose to develop an automated gate opening system to address the common need for secure and hassle-free access to residential parking areas. By utilizing NFC technology, specifically RFID cards, along with an emergency QR code request system, we provide a best solution for managing vehicle access. This project demonstrates how advanced technologies can be seamlessly incorporated into home security systems, making them more user-friendly and effective.

KEY WORDS : Near-Field Communication , Radio-Frequency Identification , Quick Response Code.

PROBLEM STATEMENT

Convenience and security are the top priorities in modern homes. Traditional gate opening systems have operational inefficiencies and vulnerabilities since they frequently depend on manual labor or simple remote controls. Our project applies a cutting-edge, specially designed Smart Gate Opening System for residential use to solve these problems. The primary problem we tackle is the need for a secure, automated method to control access to residential vehicle parking areas, minimizing unauthorized entry and enhancing ease of use for residents.

SOLUTION

Our solution leverages NFC technology, incorporating RFID cards and a supplementary QR code request system for emergency situations or new user access. When a vehicle approaches the parking area, an ultrasonic sensor detects its presence and prompts the user via an LCD screen to present their RFID card. Authorized users can access the parking area seamlessly, while new users or those who have lost their cards can request access through a QR code. This system not only improves security but also adds a layer of convenience, making it a robust solution for modern homes.

OBJECTIVE

Our objective is to significantly reduce manual intervention, enhance user experience, and improve safety and flexibility by integrating automated technology into the gate opening system, ensuring a seamless and secure access process.

SCOPE

Our system initially targets residential gate, aiming to enhance security and convenience through automated gate access and QR code fail-safes. Additionally, further future you can improve system is to adapt the system for corporate office parks and gated communities, incorporating features for effective queue management.

PROJECT DESIGN AND IMPLEMENTATION

HARDWARE DESIGN INCLUDING BLOCK DIAGRAMS AND SCHEMATICS

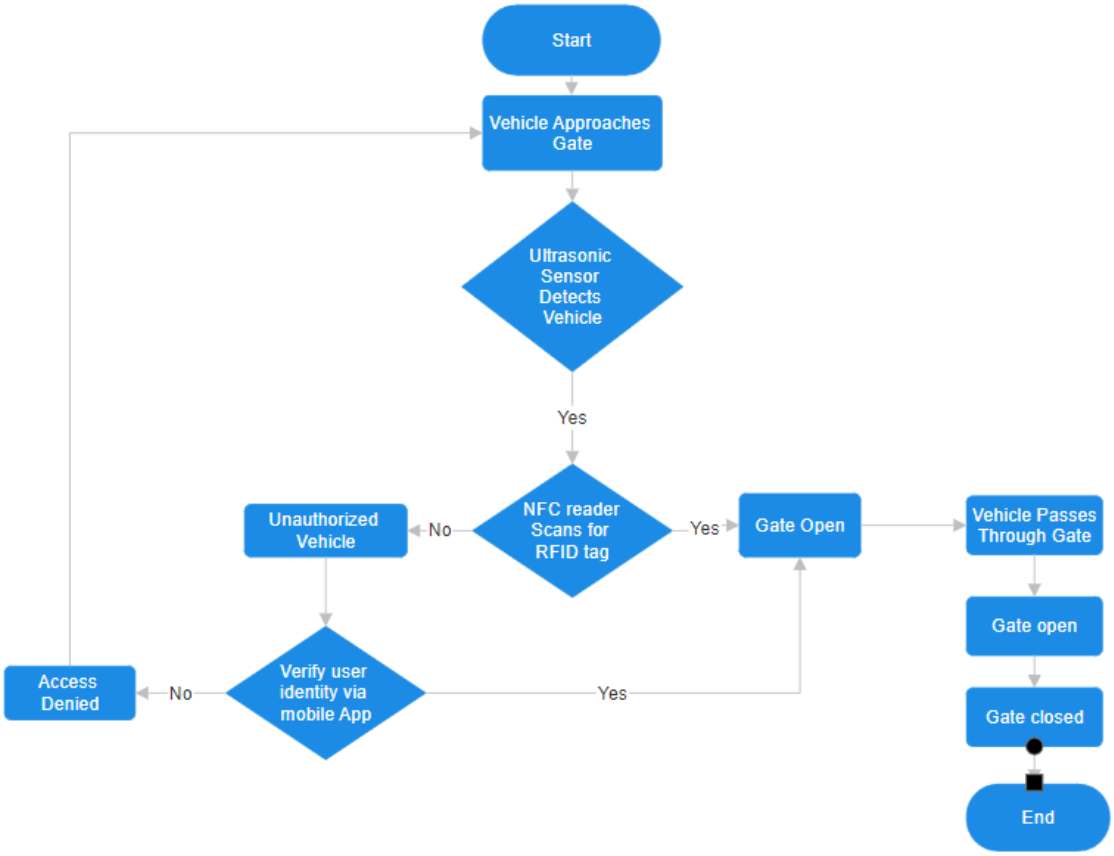


Figure 1 : Flow chart diagram for system work follow

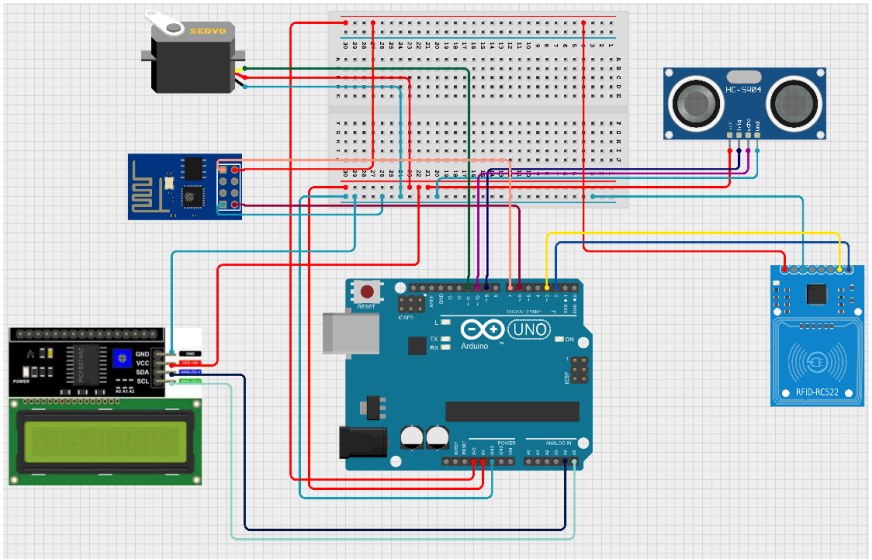


Figure 2: System Architecture

HARDWARE COMPONENT DESCRIPTION

RFID-RC522 Module

- A low-cost device that reads and writes RFID tags at 13.56 MHz.
- RFID can read tags up to 5 cm away.
- RFID CardsTypically have memory sizes of 1KB or 4KB for storing data.
- Used in access control systems for secure entry.
- RFID-RC522 connects to the Arduino Uno using the SPI interface.

Arduino Uno

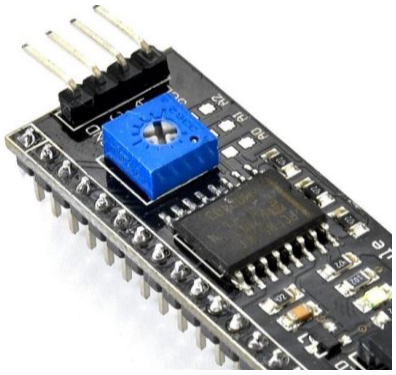
- It is based on the ATmega328P microcontroller, which runs at 16 MHz and provides a wide range of functionalities for embedded applications.
- Can be powered via a USB connection or an external power supply (7-12V recommended).
- Programmed using the Arduino IDE with a simple, beginner-friendly language based on C++.
- The Arduino Uno serves as the central controller, coordinating the various components of our system, including the RFID module, ultrasonic sensor, servo motor, and LCD display.

HC-SR04 Ultrasonic Sensor

- Measures distance by emitting ultrasonic waves and calculating the time it takes for the echo to return after bouncing off an object.
- It provides accurate distance measurements in the range of 2 cm to 400 cm.
- The ultrasonic sensor is used to detect the presence of a vehicle near the gate. When a vehicle approaches, the sensor measures the distance and triggers the gate opening mechanism if the vehicle is within a specified rang

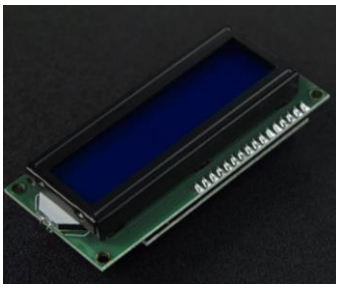
Servo motor

- It is a rotary actuator that allows precise control of angular position.
- Control Signal: Pulse Width Modulation (PWM).
- Operating Voltage: Typically 4.8V to 6V
- The servo motor is used to open and close the gate. When the system detects a vehicle or receives an authorized NFC signal, the servo motor is activated to rotate and move the gate to the open position. After a specified time or another signal, it returns the gate to the closed position.



Serial Interface Board Module

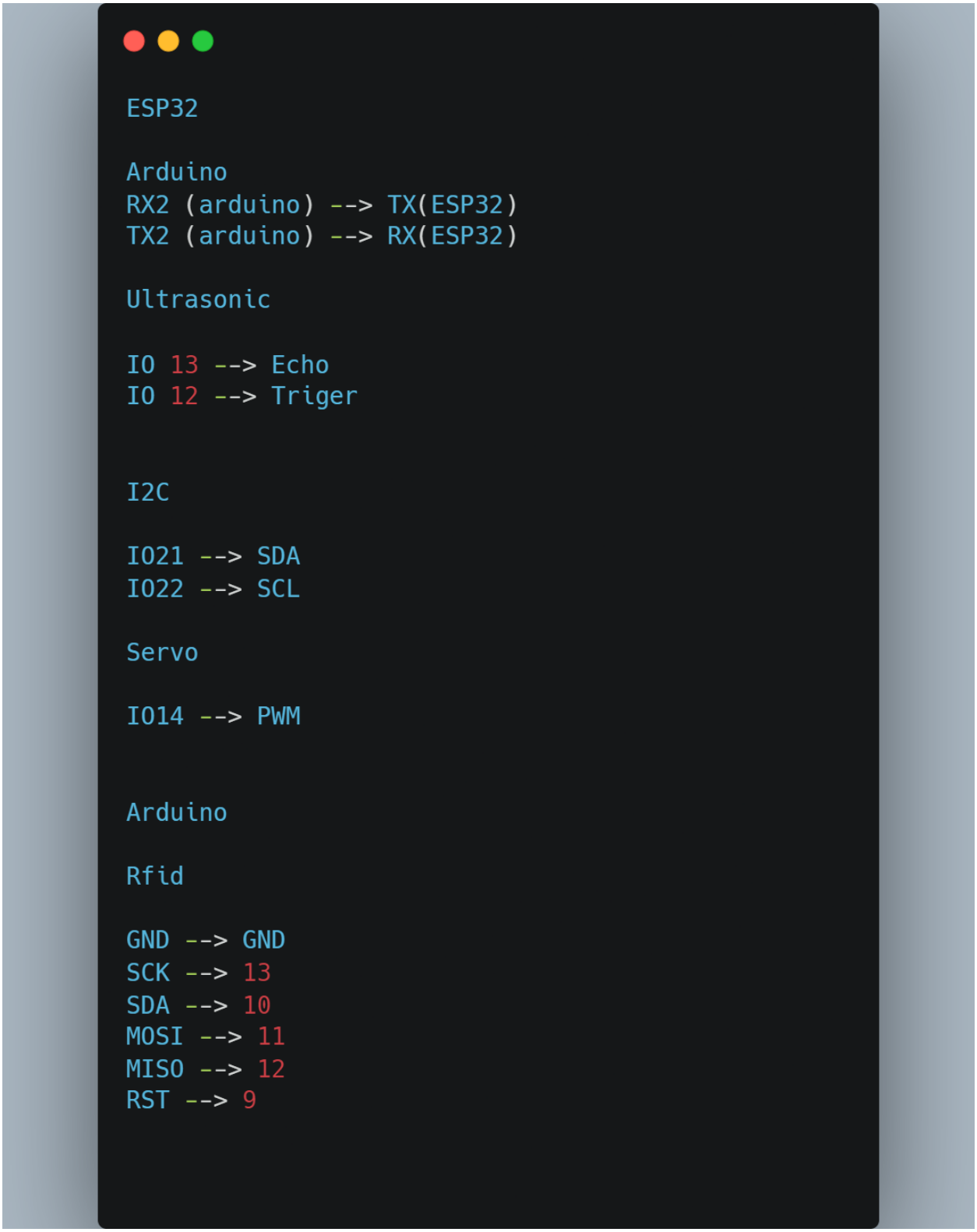
- It is often referred to as a Serial-to-USB converter ,enables communication between a microcontroller and a other devices over serial communication
- Communication Interface: Converts UART (serial communication) to USB.
- Data Transfer Rate: Supports various baud rates (e.g., 9600, 115200 bps).
- Used to program the Arduino and monitor serial data during development.



LCD

- It is a screen that shows visual information using liquid crystals and backlighting.
- Usually operates at 5V or 3.3V. LCD display provides visual feedback to the user, such as showing messages or status updates.
- For example, it can display a welcome message when the gate opens or alert the driver about the gate's status.

Pin Configuration



```
ESP32

Arduino
RX2 (arduino) --> TX(ESP32)
TX2 (arduino) --> RX(ESP32)

Ultrasonic

IO 13 --> Echo
IO 12 --> Triger


I2C

IO21 --> SDA
IO22 --> SCL

Servo

IO14 --> PWM


Arduino

Rfid

GND --> GND
SCK --> 13
SDA --> 10
MOSI --> 11
MISO --> 12
RST --> 9
```

INTERFACING

Step 1 : Vehicle Detection:

Ultrasonic Sensor
Input: Detects the presence of a vehicle near the gate.
Output: Sends a signal to the microcontroller indicating the vehicle's presence.

Microcontroller:
ATMega328P
Inputs: Receives signals from the ultrasonic sensor.
Outputs: Controls the LCD screen and communicates with the RFID reader.

Step 2 : User Interaction

LCD Screen
Input: Displays prompts such as "ADD CARD" when a vehicle is detected.
Output: Receives commands from the microcontroller to show appropriate messages.

RFID Reader
Input: Scans RFID cards presented by users.
Output: Sends RFID card data to the microcontroller for access verification.

Step 2 : Access Control

RFID Card Data Processing
Input: RFID card data received from the RFID reader.
Output: Verification result sent to the microcontroller to grant or deny access.

Step 3 : Emergency and Request Handling

QR Code System
Input: Scans QR codes for emergency access or new user requests.
Output: Sends request data to owner.

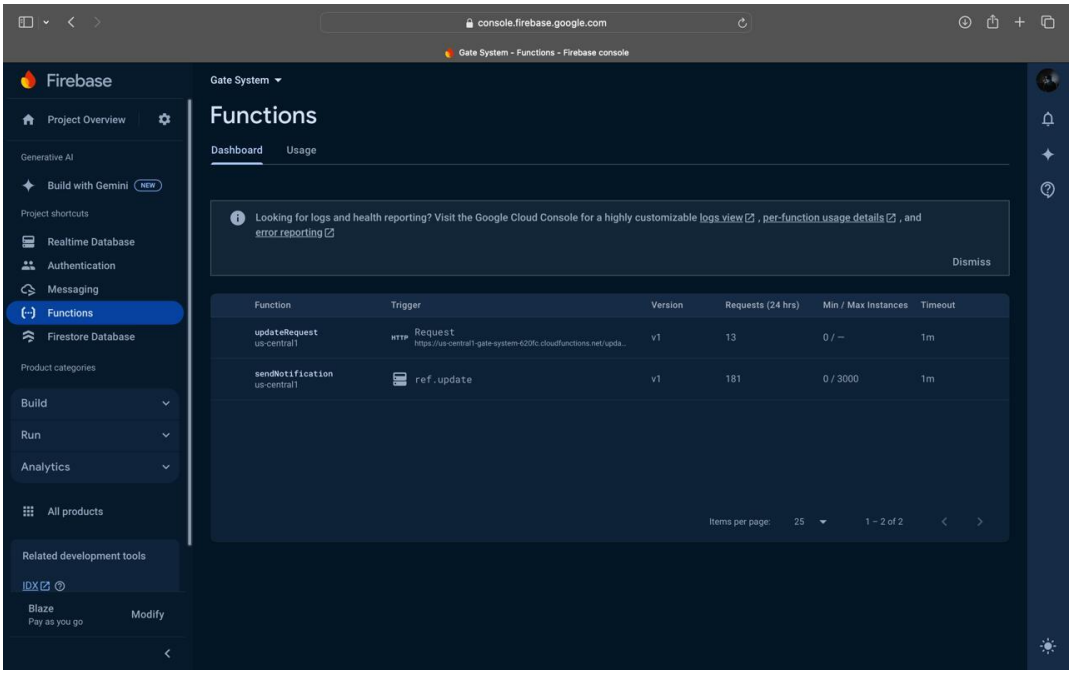
Access Authorization
Input: Data from RFID reader and QR code system.
Output: Commands to open the gate or deny access based on verification and request approval.

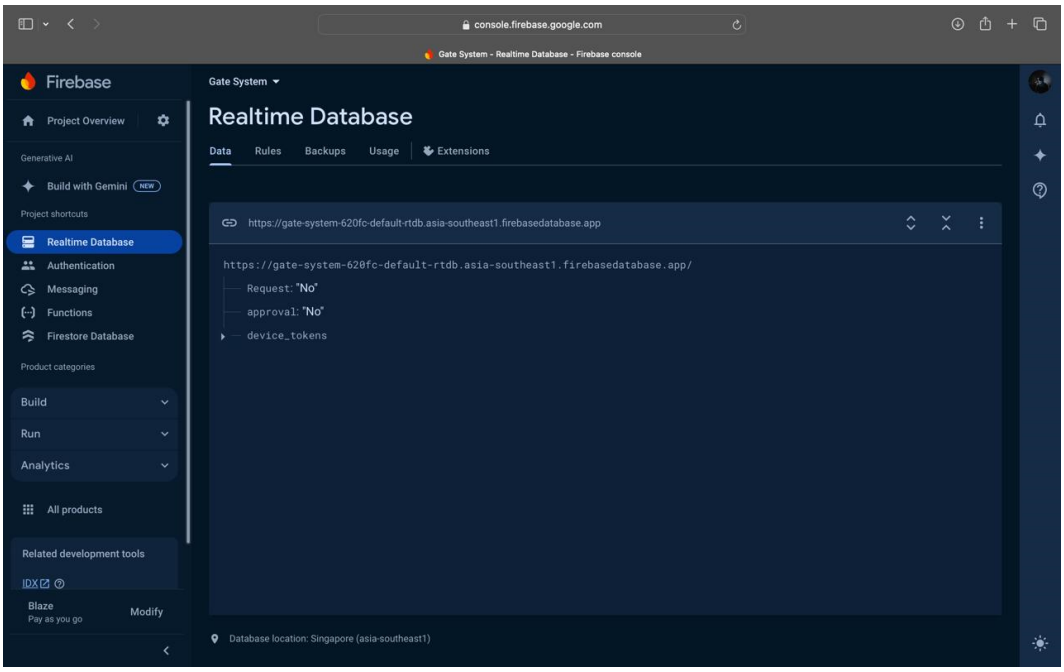
Step 4 : Gate Mechanism

Servo Motor
Input: Receives command from the microcontroller to open or close the gate.
Output: Physically opens or closes the gate.

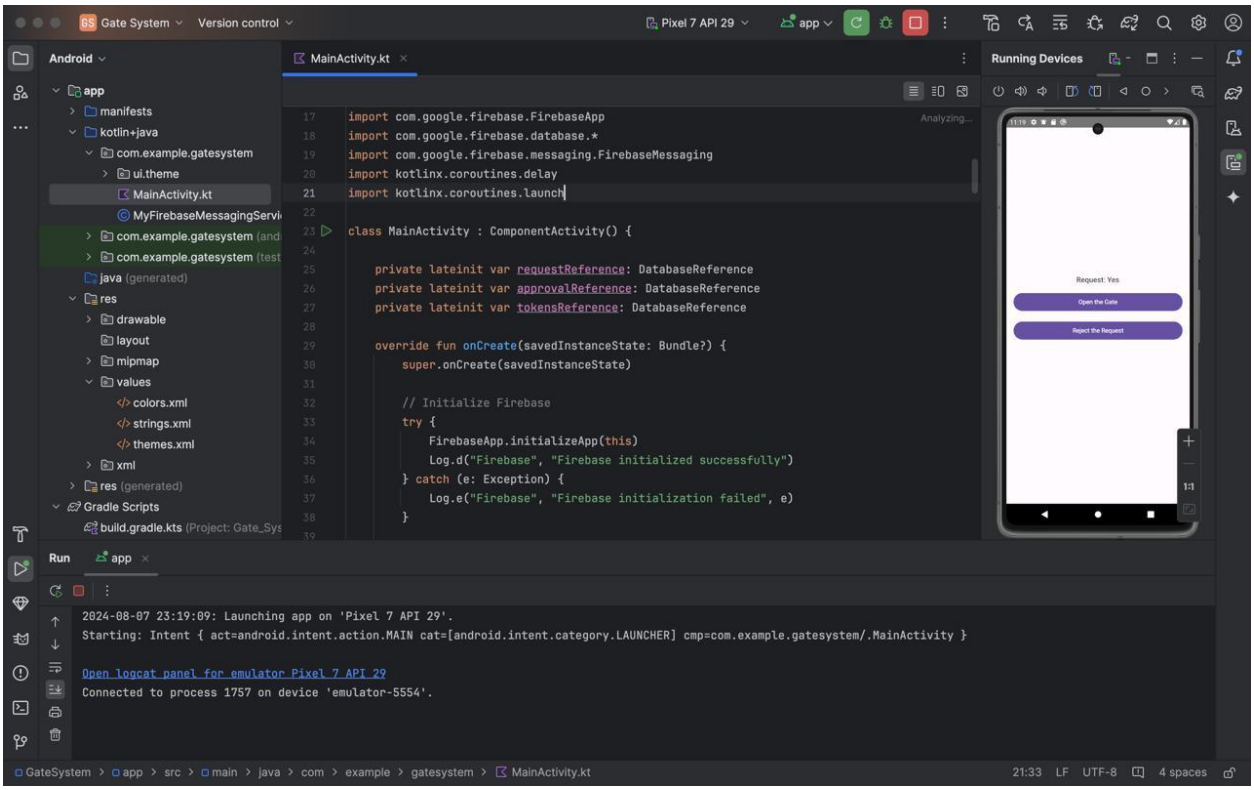
SOFTWARE DESIGN

Real time database



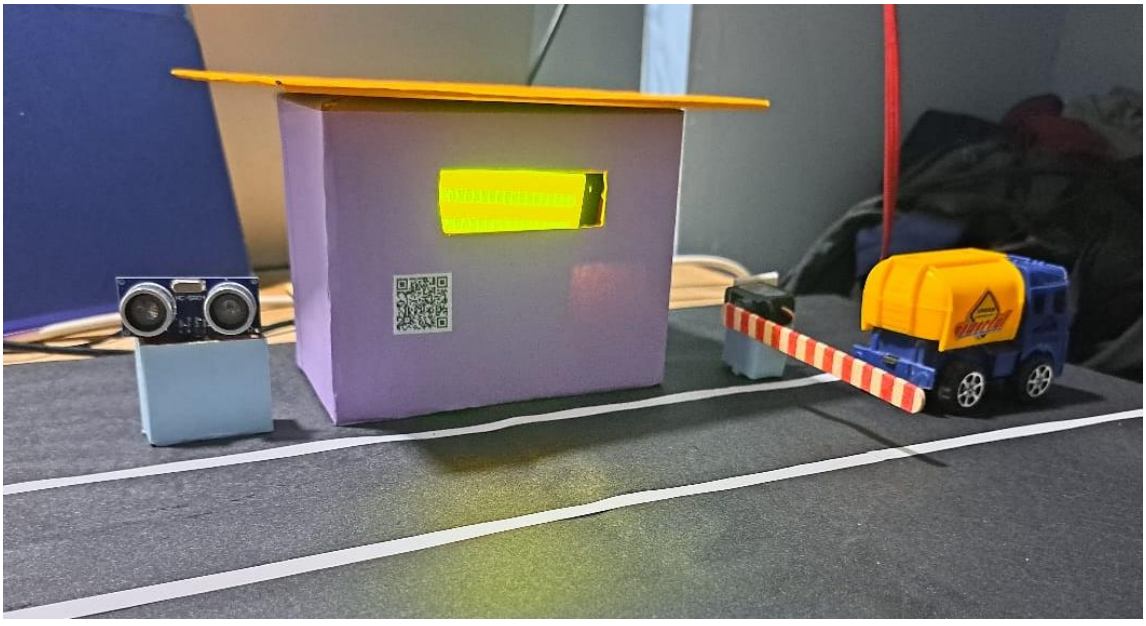


App configuration Using Android Studio



Technology : Firebase implementation such as Realtime database, Firebase Functions, Cloud messaging

IMPLEMENTATION



CHALLENGES FACED AND SOLUTIONS

Challenge: Ensuring the ultrasonic sensor reliably detects the presence of a vehicle without false positives or negatives.

Solution: Calibrate the ultrasonic sensor to the specific distance range expected for vehicle detection.

Challenge: Creating an intuitive user interface that clearly instructs users on how to use the system.

Solution: Design clear and prompts on the LCD screen, such as "ADD CARD" when the vehicle is detected.

TIMELINE

	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 14
Proposal Submission							
Collecting components							
Development NFC-Based Automatic Gate Opening System							
Development of Ultrasonic Vehicle Detection System							
Development of Owner-Requested Access							
Integration and Testing							
Documentation							
Final Demonstration & Submission							

COMPONENTS AND COST

PART NAME	QUANTITY	UNIT PRICE (LKR)	TOTAL PRICE (LKR)
RFID-RC522	1	380.00	380.00
Arduino UNO	1	2150.00	2150.00
HC-SR04 Ultrasonic Sensor	1	250.00	250.00
Node MCU ESP32S	1	420.00	1290.00
Servo	1	550.00	550.00
Bread Board	1	200.00	200.00
Jumper Wires	3	150.00	450.00
RFID Cards	2	40.00	80.00
Serial Interface Board Module	1	210.00	210.00
LCD Display	1	480.00	480.00
			6040.00

REFLECTION

Leveraging the ATMega328P microprocessor, we utilized its versatility and capability to interface with multiple peripherals. This choice was influenced by the course's emphasis on selecting appropriate microcontrollers for specific applications.

Understanding the importance of real-time operation in embedded systems, we designed our gate opening system to respond immediately to vehicle detection and user actions, ensuring seamless and efficient operation.

We applied the principles of low power consumption and cost-effective design, This consideration was crucial for creating a practical and affordable solution for residential use.

Using the knowledge gained from the course about interfacing embedded systems with different peripherals and protocols, we successfully integrated the ultrasonic sensor for vehicle detection, the RFID reader for access control, and the NFC technology for secure communication.

CONCLUSION

Our Smart Gate Opening System successfully integrates advanced NFC technology to enhance security and convenience for residential vehicle parking. We address common issues with traditional gate systems by automating gate access with RFID cards and offering a QR code-based fail-safe for emergencies or new user access. In order to provide a smooth and safe access procedure, the implementation makes use of an ultrasonic sensor to identify the presence of vehicles and an intuitive LCD panel for user interaction. The system exhibits potential for future growth in addition to satisfying the current needs of residential environments. It can be implemented with queue management capabilities by future researchers, which makes it appropriate for gated housing communities and corporate office parks. This project exemplifies how modern embedded systems can be effectively applied to solve real-world problems, enhancing both security and user experience.

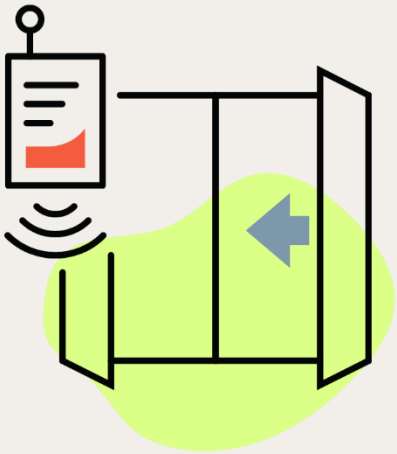
REFERENCES

- Steffen, R. et al. (2010) 'Near Field Communication (NFC) in an automotive environment', 2010 Second International Workshop on Near Field Communication [Preprint]. doi:10.1109/nfc.2010.11.
- Ang, J.T. et al. (2013) 'ISCAPS - innovative smart car park system integrated with NFC Technology and e- valet function', 2013 World Congress on Computer and Information Technology (WCCIT) [Preprint]. doi:10.1109/wccit.2013.6618762.
- USE OF ULTRASONIC SENSOR BY DETECTING VEHICLE SAFE DISTANCE BASED ON ARDUINO UNOGregorius
- Advent Trisman GaurifaIndustrial Engineering, Bhayangkara University, Greater Jakarta, Indonesia Prinsloo, J. and Malekian, R. (2016) 'Accurate vehicle location system using RFID, an internet of things approach', Sensors, 16(6), p. 825. doi:10.3390/s16060825.

APPENDIX

MINIMIZED VERSION OF THE POSTER

USER MANUAL



Smart Gate Opening System

User Manual

① Instructions for Use

Drive your vehicle close to the gate.Ensure the vehicle is within the detection range of the ultrasonic sensor

② User Interaction

Look at the LCD screen for instructions.When the screen displays "PLACE YOUR CARD," prepare your RFID card.

③ RFID Reader

Present your RFID card to the RFID reader.Hold the card steady until the reader has scanned it.

④ Access Control

The system to verify your RFID card.If authorized, the gate will automatically open

⑤ QR Code System

In case of lost RFID card or if a new user needs access, scan the provided QR code using your smartphone.
Follow the instructions on your smartphone to send an access request.

⑥ Access Authorization

Wait for the system owner's approval if you have made an access request via QR code.If access is granted, the gate will open.

⑦ Gate Mechanism

If your access is authorized, the gate will open automatically.
Drive through the gate once it has opened.
The gate will close automatically after you have passed.

CODE

MyFirebaseMessagingService

```java
package com.example.gatesystem;

import android.annotation.SuppressLint;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Build;
import android.util.Log;
import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;
import com.google.firebase.messaging.FirebaseMessagingService;
import com.google.firebase.messaging.RemoteMessage;

public class MyFirebaseMessagingService extends FirebaseMessagingService {

    private static final String CHANNEL_ID = "gatesystem_channel";
    private static final String TAG = "FCMService";

    @Override
    public void onCreate() {
        super.onCreate();
        createNotificationChannel();
    }

    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        Log.d(TAG, "From: " + remoteMessage.getFrom());

        // Handle FCM messages here.
        if (remoteMessage.getData().size() > 0) {
            String message = remoteMessage.getData().get("message");
            sendNotification(message);
        }

        if (remoteMessage.getNotification() != null) {
            String message = remoteMessage.getNotification().getBody();
            sendNotification(message);
        }
    }

    @SuppressLint("MissingPermission")
    private void sendNotification(String messageBody) {
        Intent intent = new Intent(this, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent,
PendingIntent.FLAG_ONE_SHOT | PendingIntent.FLAG_IMMUTABLE);

        NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(this,
CHANNEL_ID)
                .setSmallIcon(R.drawable.baseline_notifications_active_24) // Replace with your
notification icon
                .setContentTitle("Gate System Notification")
                .setContentText(messageBody)
                .setAutoCancel(true)
                .setContentIntent(pendingIntent);

        NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);
        notificationManager.notify(0, notificationBuilder.build());
    }

    private void createNotificationChannel() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            CharSequence name = "Gate System Channel";
            String description = "Channel for Gate System notifications";
            int importance = NotificationManager.IMPORTANCE_HIGH;
            NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
            channel.setDescription(description);

            NotificationManager notificationManager = getSystemService(NotificationManager.class);
            notificationManager.createNotificationChannel(channel);
        }
    }
}
```

FireBaseAndESP32

```
#include <Arduino.h>
#if defined(ESP32)
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif
#include <Firebase_ESP_Client.h>
#include <ESP32Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <NewPing.h> // Include the NewPing library

// Provide the token generation process info.
#include "addons/TokenHelper.h"
// Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"

// Insert your network credentials
#define WIFI_SSID "Mobitel 4G"
#define WIFI_PASSWORD "Pramuditha.1215"

// Insert Firebase project API Key
#define API_KEY "AIzaSyDIXYc05Lkl8ZITW76Y907FolQYMUbw8tc"

// Insert RTDB URL
#define DATABASE_URL "https://gate-system-620fc-default-rtdb.asia-southeast1.firebasedatabase.app

// Define Firebase Data object
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

// Servo pin
#define SERVO_PIN 14

// Servo object
Servo gateServo;

// LCD I2C address (adjust as needed, commonly 0x27 or 0x3F)
#define LCD_ADDR 0x27
#define LCD_COLUMNS 16
#define LCD_ROWS 2

// LCD object
LiquidCrystal_I2C lcd(LCD_ADDR, LCD_COLUMNS, LCD_ROWS);

// Define the pins for the ultrasonic sensor
#define TRIGGER_PIN  12
#define ECHO_PIN     13
#define MAX_DISTANCE 200 // Maximum distance to measure (in centimeters)

// Create a NewPing object
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

bool signupOK = false;
bool gateOpening = false;
bool gateClosing = false;
String lastApprovalStatus = "No";
String lastRequestStatus = "No";
```

```
void setup() {
  Serial.begin(115200);

  // Initialize LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Initializing....");

  // Connect to WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());

  // Initialize Firebase
  config.api_key = API_KEY;
  config.database_url = DATABASE_URL;
  config.token_status_callback = tokenStatusCallback; // see
addons/TokenHelper.h
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  // Sign up
  if (Firebase.signUp(&config, &auth, "", "")) {
    Serial.println("Firebase signup ok");
    signupOK = true;
  } else {
    Serial.printf("%s\n", config.signer.signupError.message.c_str());
    return;
  }
  delay(1000);

  // Attach servo to pin
  gateServo.attach(SERVO_PIN);

  // Initialize gate position (closed)
  gateServo.write(0);
  Serial.println("Gate initialized to closed position");
}
```

```
void loop() {
  // Check distance from the ultrasonic sensor
  unsigned int uS = sonar.ping(); // Measure the distance
  float distance = uS / US_ROUNDTRIP_CM; // Convert the distance to
centimeters
  // Display "Scan NFC or QR" based on the distance
  displayInitialMessage(distance);

  if (Firebase.ready() && signupOK) {
    // Read the approval and request statuses from Firebase
    if (Firebase.RTDB.getString(&fbdo, "approval")) {
      String approvalStatus = fbdo.stringData();
      if (approvalStatus != lastApprovalStatus) {
        lastApprovalStatus = approvalStatus;
        if (approvalStatus == "Yes") {
          openGate();
        }
      }
    } else {
      Serial.println("Failed to read approval status");
      Serial.println("REASON: " + fbdo.errorReason());
    }

    if (Firebase.RTDB.getString(&fbdo, "Request")) {
      String requestStatus = fbdo.stringData();
      if (requestStatus != lastRequestStatus) {
        lastRequestStatus = requestStatus;
        if (requestStatus == "Yes") {
          handleRequest();
          if (Firebase.RTDB.setString(&fbdo, "Request", "No")) {
            Serial.println("Firebase updated to No");
          } else {
            Serial.println("Failed to update Firebase to No");
            Serial.println("REASON: " + fbdo.errorReason());
          }
        } else if (requestStatus == "No") {
          displayInitialMessage(distance); // Go back to the initial message
        }
      }
    } else {
      Serial.println("Failed to read request status");
      Serial.println("REASON: " + fbdo.errorReason());
    }
  }

  delay(1000); // Check every second
}

void displayInitialMessage(float distance) {
  if (distance > 2 && distance < 5) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Scan NFC or");
    lcd.setCursor(0, 1);
    lcd.print("Scan QR");
    lcd.backlight(); // Ensure the backlight is on
  } else {
    lcd.clear();
    lcd.noBacklight(); // Turn off the backlight
  }
}
```

```
void openGate() {
  Serial.println("Opening gate");
  lcd.clear();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Gate Open");
  lcd.setCursor(0, 1);
  lcd.print("Close in: 10");
  gateServo.write(90); // Adjust angle as needed
  gateOpening = true;
  gateClosing = false;

  // Countdown before closing the gate
  for (int i = 10; i >= 0; i--) {
    lcd.setCursor(0, 1);
    lcd.print("Close in: ");
    lcd.print(i);
    lcd.print(" sec");
    delay(1000); // Wait for one second
    if (Firebase.RTDB.getString(&fbdo, "approval") && fbdo.stringData() == "No") {
      break; // Stop countdown if approval changes to "No"
    }
  }

  closeGate();
}

void closeGate() {
  Serial.println("Closing gate");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Gate Closing");
  gateServo.write(0); // Adjust angle as needed
  gateClosing = true;
  gateOpening = false;
  delay(1000); // Wait for the servo to move

  // Update Firebase
  if (Firebase.RTDB.setString(&fbdo, "approval", "No")) {
    Serial.println("Firebase updated to No");
  } else {
    Serial.println("Failed to update Firebase to No");
    Serial.println("REASON: " + fbdo.errorReason());
  }

  // Turn off the LCD
  lcd.clear();
  lcd.noBacklight(); // Turn off the LCD backlight
  delay(1000); // Wait a moment before checking again
}

void handleRequest() {
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Request send");
  lcd.setCursor(0, 1);
  lcd.print("Waiting: 10");

  // Countdown before checking approval status
  for (int i = 10; i >= 0; i--) {
    lcd.setCursor(0, 1);
    lcd.print("Waiting: ");
    lcd.print(i);
    lcd.print(" sec");
    delay(1000); // Wait for one second
    if (Firebase.RTDB.getString(&fbdo, "approval") && fbdo.stringData() == "Yes") {
      openGate();
      return;
    }
  }

  displayInitialMessage(sonar.ping() / US_ROUNDTRIP_CM); // Go back to the initial
}essage
```

arduino code

```
#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>

#define RST_PIN         9          // Configurable, see typical pin layout above
#define SS_PIN          10         // Configurable, see typical pin layout above
#define SERVO_PIN       7          // Pin connected to the servo signal line

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance
Servo myServo;                      // Create a servo object

void setup() {
    Serial.begin(9600);         // Initialize serial communications with the PC
    SPI.begin();                // Init SPI bus
    mfrc522.PCD_Init();         // Init MFRC522
}

void loop() {
    // Look for new cards
    if (!mfrc522.PICC_IsNewCardPresent()) {
        return;
    }

    // Select one of the cards
    if (!mfrc522.PICC_ReadCardSerial()) {
        return;
    }

    // Authenticate using key A
    MFRC522::MIFARE_Key key;
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;  // default key is 0xFF

    byte block = 4; // Block to read from
    byte buffer[18];
    byte size = sizeof(buffer);

    // Authenticate the block
    MFRC522::StatusCode status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key, &
(mfrc522.uid));
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("No"));
    }

    // Read data from the block
    status = mfrc522.MIFARE_Read(block, buffer, &size);
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("No"));
    }

    // Check the read data
    buffer[16] = '\0'; // Null-terminate the string
    String employeeName = (char*)buffer;

    if (employeeName.equals("Hasitha        ")) {
        Serial.println(F("Yes"));
        delay(1000);
        loop();
    } else {
        Serial.println(F("No"));
        delay(1000);
        loop();
    }

    // Halt PICC
    mfrc522.PICC_HaltA();
    // Stop encryption on PCD
    mfrc522.PCD_StopCrypto1();
}
```

Other appendix code file are in codefiles.zip file