PARKBAI: RFID SCANNER WITH SERVO SOURCE CODE

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP8266Webhook.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <FirebaseESP8266.h>  // Install Firebase ESP8266 library
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <RFID.h>
#include <TimeLib.h>
#include <ArduinoJson.h>
#include <Servo.h>
#include "time.h"

#define FIREBASE_HOST "your firebase Realtime database link here/"  //FIREBASE DATABASE HOST
#define FIREBASE_AUTH "your firebase Realtime database key here"      //FIREBASE DATABASE SECRET KEY

#define KEY "your webhooks key here"  // Webhooks Key
#define EVENT "PARKBAI_SMS"          // Webhooks Event Name
Webhook webhook(KEY, EVENT);

RFID rfid(D4, D3);                //D4:pin of tag reader SDA. D3:pin of tag reader RST
unsigned char str[MAX_LEN];         //MAX_LEN is 16: size of the array
LiquidCrystal_I2C lcd(0x27, 16, 2);  // set the LCD address to 0x27 for a 16 chars and 2 line display
```

```
// Replace with your network credentials
const char ssid[] = "ssid name";
const char pass[] = "ssid password";
WiFiUDP ntpUDP;
const long utcOffsetInSeconds = 28800;
NTPClient timeClient(ntpUDP, "time.nist.gov", utcOffsetInSeconds);  // can be change to pool.ntp.org

//Week Days
String weekDays[7] = { "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday" };
//Month names
String months[12] = { "January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December" };

FirebaseConfig config;
FirebaseAuth auth;

FirebaseJson jsonDriver;
FirebaseJson jsonParkOwner;
FirebaseData firebaseData;

//MO CONNECT SA WIFI
void connect() {

  Serial.print("CONNECTING TO WIFI PLEASE WAIT.");

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(1500);
```

```
  }

  Serial.println("\nYOUR RFID SCANNER CONNECTED SUCCESSFULLY");
  // Serial.print("ESP Board MAC Address:");
  // Serial.println(WiFi.macAddress());
  // Serial.print("IP Address: ");
  // Serial.print("http://");
  // Serial.print(WiFi.localIP());
  // Serial.println("/");
  // Serial.println(EVENT);
  // Serial.println(KEY);
}

//MO CONNECT SA FIREBASE
void setup() {

  Serial.begin(9600);
  WiFi.begin(ssid, pass);
  randomSeed(analogRead(A0));

  lcd.init();  // initialize the lcd
  lcd.clear();
  lcd.backlight();
  SPI.begin();
  rfid.init();  // initialize the rfid

  connect();
  timeClient.begin();
  timeClient.setTimeOffset(utcOffsetInSeconds);
```

```
 config.database_url = FIREBASE_HOST;
 config.signer.tokens.legacy_token = FIREBASE_AUTH;
 Firebase.reconnectWiFi(true);
Firebase.begin(&config, &auth);
 //Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}



//MO READ SA CARD
void loop() {
 // Call your logic for checking access here
 // For example, check access every 5 seconds
 static unsigned long previousMillis = 0;
 const unsigned long interval = 3000;  // 5 seconds

 unsigned long currentMillis = millis();



 // Check access here
 if (rfid.findCard(PICC_REQIDL, str) == MI_OK)  //Wait for a tag to be placed near the reader
 {
  Serial.println("RFID CARD DETECTED!");
  String rfidcard = "";           //Temporary variable to store the read RFID number
  if (rfid.anticoll(str) == MI_OK)  //Anti-collision detection, read tag serial number
  {
    Serial.print("The card's ID number is : ");
    for (int i = 0; i < 4; i++)  //Record and display the tag serial number
    {
      rfidcard = rfidcard + (0x0F & (str[i] >> 4));
```

```
      rfidcard = rfidcard + (0x0F & str[i]);
    }
    Serial.println(rfidcard);
    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;

      //FUNCTION NGA MO PASA SA RFIDCARD VALUE SA CHECKACCESS() FUNCTION
      checkAccess(rfidcard);  //Check if the identified tag is an allowed to open tag
    }
   }
   rfid.selectTag(str);  //Lock card to prevent a redundant read, removing the line will make the sketch read cards continually
  }
  rfid.halt();

  lcd.setCursor(1, 0);
  lcd.print("TAP YOUR CARD");
  lcd.setCursor(3, 1);
  lcd.print("GATE CLOSE");
  delay(300);
  lcd.clear();

  // Any other code that needs to run continuously in the loop
}

void checkAccess(String rfidcard) {
  lcd.setCursor(1, 0);
  lcd.print("TAP YOUR CARD");
```

```
  // Synchronize time
  bool timeUpdated = timeClient.update();
  if (timeUpdated) {
    Serial.println("Time synchronized successfully.");
    // You can access the synchronized time using timeClient.getFormattedTime() or other time-related functions
    String formattedTime = timeClient.getFormattedTime();
    Serial.println("Current time: " + formattedTime);
  } else {
    Serial.println("Time synchronization failed.");
    // Implement error handling or retry logic here
  }
  time_t epochTime = timeClient.getEpochTime();
  struct tm *ptm = gmtime((time_t *)&epochTime);
  int monthDay = ptm->tm_mday;
  int currentMonth = ptm->tm_mon + 1;
  int currentYear = ptm->tm_year + 1900;
  String weekDay = weekDays[timeClient.getDay()];
  String currentMonthName = months[currentMonth - 1];
  String currentDate = String(monthDay) + "-" + String(currentMonth) + "-" + String(currentYear);
  String currentDateWordFormat = String(currentMonthName) + " " + String(monthDay) + ", " + String(currentYear) + " (" + String(weekDay) + ") " + String(timeClient.getFormattedTime());

  // Get RFID values
  Firebase.get(firebaseData, "/ADMIN/RFID-TO-DRIVER/" + rfidcard);
  String rfidvalue = firebaseData.stringData();

  Firebase.get(firebaseData, "/ADMIN/RFID-TO-VEHICLE/" + rfidcard);
  String rfidvehicle = firebaseData.stringData();
```

```
String driver = "DRIVER/";

if (rfidvalue != "") {

  Firebase.get(firebaseData, driver + rfidvalue + "/ACCOUNT/firstname");
  String firstname = firebaseData.stringData();

  Firebase.get(firebaseData, driver + rfidvalue + "/ACCOUNT/phonenumber");
  String phonenumber = firebaseData.stringData();

   Firebase.get(firebaseData, driver + rfidvalue + "/ACCOUNT/license");
   String driverLicense = firebaseData.stringData();

  Firebase.getInt(firebaseData, driver + rfidvalue + "/ACCOUNT/balance");
  int balance = firebaseData.intData();

  Firebase.get(firebaseData, driver + rfidvalue + "/ACCOUNT/status");
  String accountStatus = firebaseData.stringData();

  Firebase.getDouble(firebaseData, "/ADMIN/parkbai_balance");
  double parkbaiBalance = firebaseData.doubleData();

  String ownerID = "bFnYTNgnXMYLrwwjqDoxzreu7Rn2";
  String parkOwner = "PARK_OWNER/";

  Firebase.getString(firebaseData, parkOwner + ownerID + "/PARKING_LOT/Company");
  String ParkName = firebaseData.stringData();
  Firebase.getString(firebaseData, parkOwner + ownerID + "/PARKING_LOT/Address");
```

```
  String Location = firebaseData.stringData();
  Firebase.getInt(firebaseData, parkOwner + ownerID + "/PARKING_FEE/Fee_per3Hr");
  int parkingFee = firebaseData.intData();
  Firebase.getInt(firebaseData, parkOwner + ownerID + "/PARKING_FEE/succeedingHr");
  int succeedingHr = firebaseData.intData();
  Firebase.getInt(firebaseData, parkOwner + ownerID + "/PARKING_FEE/penalty");
  int penalty = firebaseData.intData();
  Firebase.getDouble(firebaseData, parkOwner + ownerID + "/INCOME/Current_Balance");
  double currentBalance = firebaseData.doubleData();
  Firebase.getDouble(firebaseData, parkOwner + ownerID + "/INCOME/Daily_Income" + currentDate +
"/amount");
  double currentAmount = firebaseData.doubleData();
  Firebase.getDouble(firebaseData, parkOwner + ownerID + "/INCOME/Total_Income");
  double totalIncome = firebaseData.doubleData();

  Firebase.getDouble(firebaseData, "/ADMIN/percent");
  double percent = firebaseData.doubleData();

  if (balance < 100) {
    lcd.clear();
    lcd.setCursor(1, 0);
    lcd.print("SORRY, NOT");
    lcd.setCursor(2, 1);
    lcd.print("ENOUGH BALANCE");
    delay(3000);
    lcd.clear();
  } else {
    if (accountStatus == "ofline") {
      Serial.print("rfid card: ");
```

```
      Serial.println(rfidcard);


      Serial.print("rfid card value: ");
      Serial.println(rfidvalue);


      Serial.print("platenumber: ");
      Serial.println(rfidvehicle);


      Serial.print("firstname: ");
      Serial.println(firstname);


      Serial.print("phonenumber: ");
      Serial.println(phonenumber);


      Serial.print("balance: ");
      Serial.println(balance);
      // Parking In logic
      performParkingIn(rfidvalue, rfidvehicle, currentDate);
      openTollGate();
      displayMessageParkIn();


    } else if (accountStatus == "online") {
      // Parking Out logic
      performParkingOut(driver, rfidcard, rfidvalue, rfidvehicle, currentDate, balance, driverLicense,
firstname, phonenumber, currentDateWordFormat,
                        parkbaiBalance, ownerID, parkOwner, ParkName, Location, parkingFee, succeedingHr,
penalty, currentBalance, currentAmount, totalIncome, percent);
      openTollGate();
      displayMessageParkOut();
```

```
    }
   }
  } else {
   Serial.println("CARD NOT FOUND");
   lcd.clear();
   lcd.setCursor(1, 0);
   lcd.print("CARD NOT FOUND");
   delay(3000);
   lcd.clear();
  }
}

void performParkingIn(String rfidvalue, String rfidvehicle, String currentDate) {

  Firebase.setString(firebaseData, "/DRIVER/" + rfidvalue + "/ACCOUNT/status/", "online");
  Firebase.setString(firebaseData, "/DRIVER/" + rfidvalue + "/VEHICLE/" + rfidvehicle + "/status/",
"Parked");
  Firebase.setString(firebaseData, "/DRIVER/" + rfidvalue + "/ACCOUNT/time_in/",
String(timeClient.getFormattedTime()));
  Firebase.setString(firebaseData, "/DRIVER/" + rfidvalue + "/ACCOUNT/time_out/", "--: ---- ");
  Firebase.setString(firebaseData, "/DRIVER/" + rfidvalue + "/ACCOUNT/date_start/", currentDate);
  Firebase.setString(firebaseData, "/DRIVER/" + rfidvalue + "/ACCOUNT/date_end/", "--/--/--");
}

unsigned long generateRandomNumber(unsigned long min, unsigned long max) {
  // Generate a random number within the specified range
  return random(min, max + 1);
}
```

```
void performParkingOut(String driver, String rfidcard, String rfidvalue, String rfidvehicle, String currentDate,
                       int balance, String driverLicense, String firstname, String phonenumber, String currentDateWordFormat,
                       double parkbaiBalance, String ownerID, String parkOwner, String ParkName, String Location, int parkingFee, int succeedingHr, int penalty, double currentBalance, double currentAmount, double totalIncome, double percent) {


  Firebase.setString(firebaseData, driver + rfidvalue + "/ACCOUNT/time_out/", String(timeClient.getFormattedTime()));
  Firebase.setString(firebaseData, driver + rfidvalue + "/ACCOUNT/date_end/", currentDate);


  Firebase.get(firebaseData, "/DRIVER/" + rfidvalue + "/ACCOUNT/date_start");
  String dateStart = firebaseData.stringData();


  Firebase.get(firebaseData, "/DRIVER/" + rfidvalue + "/ACCOUNT/date_end");
  String dateEnd = firebaseData.stringData();


  int getDayStart = dateStart.substring(0, 2).toInt();
  int getDayEnd = dateEnd.substring(0, 2).toInt();


  Firebase.get(firebaseData, driver + rfidvalue + "/ACCOUNT/time_in");
  String timeIn = firebaseData.stringData();


  double timeInHr = timeIn.substring(0, 2).toDouble();
  double timeInMin = timeIn.substring(3, 5).toDouble();
  double timeInSec = timeIn.substring(6, 8).toDouble();
  double timeIntotalSeconds = timeInHr * 3600 + timeInMin * 60 + timeInSec;
  double timeInresult = timeIntotalSeconds / 3600;
```

```
Firebase.get(firebaseData, driver + rfidvalue + "/ACCOUNT/time_out");
String timeOut = firebaseData.stringData();
double timeOutHr = timeOut.substring(0, 2).toDouble();
double timeOutMin = timeOut.substring(3, 5).toDouble();
double timeOutSec = timeOut.substring(6, 8).toDouble();
double timeOuttotalSeconds = timeOutHr * 3600 + timeOutMin * 60 + timeOutSec;
double timeOutresult = timeOuttotalSeconds / 3600;

int numofDaysPark = getDayEnd - getDayStart;

double updateBalance = 0;
double totalSecPark = 0;
double totalHrPark = 0;
double totalHoursPark = 0;
double totalPayment = 0;
double adminPercent = 0;
double PercentFirst = 0;
double parkOwnerIncome = 0;
double updateCurrentAmount = 0;
double updateTotalIncome = 0;
double totalSucceedingHr = 0;

//CALCULATE PAYMENT
// IF TIME OUT IS LESSER THAN TIME IN
// EXCEED TO THE NEXT DAY
if (timeOuttotalSeconds < timeIntotalSeconds) {

  double timeOutTotalSecAdded24hr = (timeOutHr + 24) * 3600 + timeOutMin * 60 + timeOutSec;
  totalSecPark = timeOutTotalSecAdded24hr - timeIntotalSeconds;
```

```
totalHrPark = totalSecPark / 3600;

// NOT EXCEED TO THE NEXT DAY  NO CHARGING OVERNIGHT FEE
if (numofDaysPark < 1) {

  // NO ADDITIONAL SUCCEDING HOURS
  if (totalHrPark <= 3) {

    totalPayment = parkingFee;
    totalSucceedingHr = 0.00;
    penalty = 0.00;

  }
  // ADDITIONAL SUCCEDING HOURS
  else {
    totalSucceedingHr = totalHrPark - 3;
    totalSucceedingHr = totalSucceedingHr * succeedingHr;
    totalPayment = parkingFee + totalSucceedingHr;
    penalty = 0.00;
  }
}

// EXCEED TO THE NEXT DAY CHARGING OVERNIGHT FEE
else {


  // NO ADDITIONAL SUCCEDING HOURS
  if (totalHrPark <= 3) {
```

```
      totalPayment = parkingFee + penalty;
      totalSucceedingHr = 0.00;

    }
    // ADDITIONAL SUCCEDING HOURS
    else {
      totalSucceedingHr = totalHrPark - 3;
      totalSucceedingHr = totalSucceedingHr * succeedingHr;
      totalPayment = parkingFee + totalSucceedingHr + penalty;
    }
  }

  updateBalance = balance - totalPayment;
  PercentFirst = totalPayment * percent;
  adminPercent = PercentFirst + parkbaiBalance;
  parkOwnerIncome = currentBalance + (totalPayment - PercentFirst);
  updateTotalIncome = totalIncome + (totalPayment - PercentFirst);
  updateCurrentAmount = parkOwnerIncome + currentAmount;
}

// IF TIME OUT IS GREATER THAN TIME IN
// WITH IN THE DAY
else {
  totalSecPark = timeOuttotalSeconds - timeIntotalSeconds;
  totalHrPark = totalSecPark / 3600;

  // NOT EXCEED TO THE NEXT DAY  NO CHARGING OVERNIGHT FEE
  if (numofDaysPark < 1) {
```

```
  // NO ADDITIONAL SUCCEDING HOURS
  if (totalHrPark <= 3) {

    totalPayment = parkingFee;
    totalSucceedingHr = 0.00;
    penalty = 0.00;

  }
  // NO ADDITIONAL SUCCEDING HOURS
  else {
    totalSucceedingHr = totalHrPark - 3;
    totalSucceedingHr = totalSucceedingHr * succeedingHr;
    totalPayment = parkingFee + totalSucceedingHr;
    penalty = 0.00;
  }
}
// EXCEED TO THE NEXT DAY CHARGING OVERNIGHT FEE
else {
  // NO ADDITIONAL SUCCEDING HOURS
  if (totalHrPark <= 3) {

    totalPayment = parkingFee + penalty;
    totalSucceedingHr = 0.00;

  }
  // NO ADDITIONAL SUCCEDING HOURS
  else {
    totalSucceedingHr = totalHrPark - 3;
    totalSucceedingHr = totalSucceedingHr * succeedingHr;
```

```
    totalPayment = parkingFee + totalSucceedingHr + penalty;
   }
  }

  updateBalance = balance - totalPayment;
  PercentFirst = totalPayment * percent;
  adminPercent = PercentFirst + parkbaiBalance;
  parkOwnerIncome = currentBalance + (totalPayment - PercentFirst);
  updateTotalIncome = totalIncome + (totalPayment - PercentFirst);
  updateCurrentAmount = parkOwnerIncome + currentAmount;
}

Serial.print("time in to seconds: ");
Serial.println(timeIntotalSeconds);

Serial.print("time out to seconds: ");
Serial.println(timeOuttotalSeconds);



Serial.print("Parking fee: ");
Serial.println(parkingFee);

Serial.print("total hours park: ");
Serial.println(totalHrPark);

Serial.print("total payment: ");
Serial.println(totalPayment);

Serial.print("Updated Balance: ");
```

```
  Serial.println(updateBalance);


  Serial.print("Admin (mangawartahay) porsyensto: ");
  Serial.println(adminPercent);


  Serial.print("Owner (chill ra ez money) income: ");
  Serial.println(parkOwnerIncome);


  Serial.print("Owner (chill ra ez money) total income: ");
  Serial.println(updateTotalIncome);


  Serial.print("Owner (chill ra ez money) income everyday ni sya: ");
  Serial.println(updateCurrentAmount);


  Firebase.setString(firebaseData, driver + rfidvalue + "/ACCOUNT/status/", "ofline");
  Firebase.setString(firebaseData, driver + rfidvalue + "/VEHICLE/" + rfidvehicle + "/status/", "---");
  Firebase.setDouble(firebaseData, driver + rfidvalue + "/ACCOUNT/balance/", updateBalance);


  Firebase.setDouble(firebaseData, parkOwner + ownerID + "/INCOME/Daily_Income/" + currentDate +
"/amount/", updateCurrentAmount);
  Firebase.setDouble(firebaseData, parkOwner + ownerID + "/INCOME/Current_Balance/",
parkOwnerIncome);
  Firebase.setDouble(firebaseData, parkOwner + ownerID + "/INCOME/Total_Income/",
updateTotalIncome);


  Firebase.setDouble(firebaseData, "ADMIN/parkbai_balance/", adminPercent);


  unsigned long randomNumber = generateRandomNumber(100000000, 999999999);


  jsonDriver.add("ref_number", randomNumber);
```

```
  jsonDriver.add("rfid", rfidcard);
  jsonDriver.add("date", currentDateWordFormat);
  jsonDriver.add("time_in", timeIn);
  jsonDriver.add("time_out", timeOut);
  jsonDriver.add("add_Fee", totalSucceedingHr);
  jsonDriver.add("overnight_fee", penalty);
  jsonDriver.add("hours_park", totalHrPark);
  jsonDriver.add("total_payment", totalPayment);
  jsonDriver.add("park_address", Location);
  jsonDriver.add("company_name", ParkName);
  jsonDriver.add("ownerID", ownerID);

  Firebase.setJSON(firebaseData, driver + rfidvalue + "/PARKING_HISTORY/" + currentDateWordFormat,
jsonDriver);

  jsonParkOwner.add("ref_number", randomNumber);
  jsonParkOwner.add("rfid", rfidcard);
  jsonParkOwner.add("date", currentDate);
  jsonParkOwner.add("time_in", timeIn);
  jsonParkOwner.add("time_out", timeOut);
  jsonParkOwner.add("hours_park", totalHrPark);
  jsonParkOwner.add("total_payment", totalPayment);
  jsonParkOwner.add("driverLicense", driverLicense);
  jsonParkOwner.add("user_uid", rfidvalue);

  Firebase.pushJSON(firebaseData, parkOwner + ownerID + "/PARKING_HISTORY/", jsonParkOwner);
  Firebase.pushJSON(firebaseData, "ADMIN/DRIVER/PARKING_TRANSACTION/", jsonParkOwner);
```

```
  String sendMessage = "Hello, " + firstname + " you time in at: " + timeIn + " and time out at: " + timeOut + " your total hours parked: " + totalHrPark + ", Total payment:P" + totalPayment + " php and your balance now is:P" + updateBalance + " php on date: " + currentDateWordFormat + ", reference number:" + randomNumber + " Thank you for parking with us!";
  webhook.trigger(phonenumber, sendMessage);
}



Servo tollGate;
int servoPin = D8;
void openTollGate() {
  tollGate.attach(servoPin);
  tollGate.write(90 * 2);
  delay(3000);
  tollGate.write(0);
}

void displayMessageParkIn() {
  // Code to display a message on an LCD screen
  // Example code to display a message on an LCD
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("CARD DETECTED");
  lcd.setCursor(3, 1);
  lcd.print("PARKING IN");
  delay(3000);
  lcd.clear();
}
```

```
void displayMessageParkOut() {
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("PARKING OUT");
  lcd.setCursor(2, 1);
  lcd.print("THANK YOU !!!");
  delay(3000);
  lcd.clear();
}
```