Sistema de Peaje por Identificacion de Radio Frecuencia (RFID).

Ocampos R. G.[a], Oviedo R. A.[b], Meaurio B. I.[c], Melgarejo R. D. H.[d],Cho D. L.[e]

Universidad Nacional de Asunción
San Lorenzo, Paraguay.

[a]eocampos@fiuna.edu.py,[b]aaoviedor@fiuna.edu.py,[c]iemeaurio@fiuna.edu.py

[d]hruizdiaz@fiuna.edu.py,[e]lcho@fiuna.edu.py

***Resumen*—En la acutalidad se realizan pagos de peaje por efectivo, sin embargo, nuestra sociedad transiciona a un sistema donde el pago en efectivo se convierte cada vez menos práctico. En este proyecto, se realiza el diseño de una base de datos, la implementación en SQL y el montaje físico de un sistema de peaje por RFID capaz de registrar el ingreso de los empleados al puesto y el cruce de los automóviles con su informacón a una base de datos.**

***Palabras Clave*—SQL, Python, Base de Datos, RFID.**

INTRODUCCION

Los peajes son tarifas que se cobran para utilizar rutas, principalmente utilizados para financiamiento y mantenimiento de ellas. Actualmente los pagos se pueden realizar por tarjetas de credito, débito o efectivo, sin embargo, esto puede tomar mucho tiempo, considerando las altas velocidades transitadas en rutas.

Las tarjetas RFID son tarjetas que utilizan frecuencias de radio para leer y escribir informacion en ellas, responden rápido y son seguras. Actualmente son utilizadas para billetaje elctrónico en los buses.

Un sistema de pago de peaje por RFID podría ser útil para los puestos de peaje, facilitando y agilizando el pago de manera electrónica y segura, igual que el billetaje electronico.

SECCIÓN 1: GENERALIDADES DEL PROYECTO

I. PLANTEAMIENTO DEL PROBLEMA

Se desea implementar un nuevo sistema de gestión para puestos de peaje, que facilite el pago del mismo. El sistema también debe ayudar a gestionar el registro de ingreso y salida de los empleados.

Los empleados deben ingresar a sus debidos turnos y registrar sus entradas y salidas en un biométrico. Para eso es necesario identificar a los empleados, sus cargos y registrar las horas de llegada.

El sistema de peaje utiliza un lector RFID, en este nuevo sistema, cada automóvil debe tener su propia tarjeta RFID, donde se encuentra información del auto como el tipo, la chapa (unica), la marca, y el propietario.

El sistema verificara en la base de datos el dueño del auto, un dueño puede tener varios autos registrados a su nombre. Si el dueño se encuentra registrado, se verifica su saldo y si es suficiente, se realiza el pago automáticamente.

Por seguridad y para verificar los ingresos diarios, el sistema almacena el tiempo y el monto de los pagos. Es por esto que se desea diseñar e implementar una base de datos junto a una GUI (Graphic User Interface) para gestionar la base de datos y realizar el proyecto.

II. OBJETIVOS GENERALES Y ESPECÍFICOS

Objetivos generales:

- Determinar las entidades en cuestión y sus propiedades.
- Identificar las relaciones entre las entidades para determinar el tipo de base de datos que será necesaria.
- Diseñar la base de datos más eficiente que cumpla con las especificaciones.
- Implementar la base de datos en un Raspberry

Objetivos específicos:

- Registrar a los empleados por nombre, apellido, cargo y CIN y registrar sus horas de llegada
- Almacenar la información de los choferes, nombre, apellido, CIN y saldo
- Registrar a cada auto con chapa única, con la información de su tipo, marca, color.
- Registrar las horas de cruce de los autos y su información.
- Desarrollar una GUI para los empleados que facilite la gestión de la base de datos.

SECCION 2: MARCO TEÓRICO

Se ha realizado varios trabajos previos relacionados a la problemática, “Diseño de un sistema de control de tráfico y peaje vehicular en la ciudad de Lima utilizando tecnología RFID de ultra alta frecuencia”[1]. Donde se implementó un servidor de bajo costo, que gestiona detección de automóviles por RFID de ultra alta frecuencia para determinar prioridades entre los autos y para realizar el cobro de peajes. Una de las

características más resaltantes de este proyecto, es que es de bajo costo y no se desarrolló infraestructura adicional.

Otro proyecto similar, “Diseño de un sistema de cobro automatizado en el Peaje Chillón, para mejorar el tráfico vehicular en la Panamericana norte, utilizando tecnología RFID”[2], asigna a cada automóvil una etiqueta, que tiene la información del vehículo, y además la información de crédito para realización del pago por RFID.

I. MATERIALES Y MÉTODOS

Para implementar la solucón, se necesitarán los siguentes materiales:

- Raspberry pi 3
- Tarjeta de memoria SD
- 2 Sensores RFID
- Arduino UNO
- Multiples Tarjetas RFID
- Cables unifilares

II. DESARROLLO

Diseño de la base de datos

Se conocen las entidades en cuestión de manera general y sus propiedades según la Tabla 1.

Entidad	Propiedades
Empleados	Id, Nombre, Apellido, Cargo, CIN
Choferes	Id, Nombre, Apellido, CIN, teléfono, saldo
Automóvil	Id, Tipo, Chapa, Marca
Registro	Id, Automóvil, Fecha hora, Monto abonado
Biométrico	Id, Empleado, Fecha hora

Tabla. 1.

Las entidades tienen como propiedades otras entidades, por lo que deben existir relaciones entre las tablas, así se determina que el tipo de base de datos necesaria debe ser relacional. La figura 1 representa las relaciones entre las entidades de manera general.

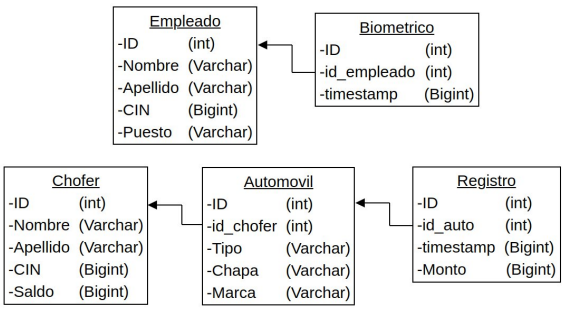

Fig. 1. Esquema preeliminar de la base de datos.

Desarrollo de códigos

Para la creación de la base de datos, tablas y relaciones, se utilizó un script en Python con la librería mysql.connector (Anexo 1 - BaseDatos.py) para evitar la utilización de interfaces gráficas de gestores de base de datos. Esto resulta en una ventaja en la implementación del proyecto en Raspberry, ya que solo se necesitará una GUI para administrar el sistema entero.

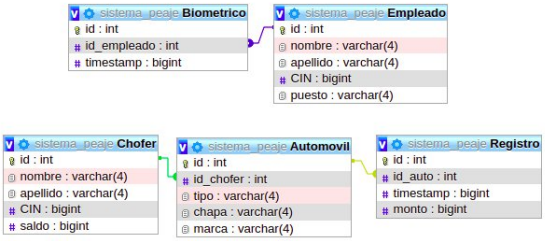

Fig. 2. Esquema de la base de datos implementada en SQL.

Para registrar los ingresos y cruces, se desarrollaron códigos en Python y Arduino. El código en arduino (Anexo 1 – senosres.ino) opera dos sensores RFID y envía una cadena que contiene la ID de la tarjeta por el puerto serial.

Para cargar los datos del ingreso de los empleados y el cruce de autos en la base de datos, se desarrolló un código en Python (Anexo 1 – serial_control.py) que lee los datos del puerto serial con un timestamp, y los carga en la tabla de Biometrico o Registro respectivamente.

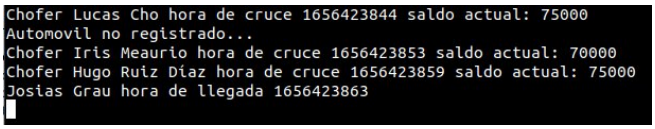

Fig. 3. Control del puerto serial

Para el monitoreo de los datos, se desarrolló una interfáz gráfica con la librería Tkinter de Python (Anexo 1 – gui_manager.py).

La interfáz desarrollada es capaz de solicitar datos diarios, semanales y mensuales a la base de datos, de las tablas Biométrico y Registro.

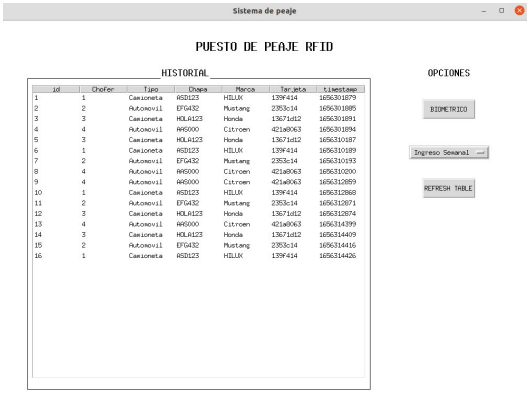

Fig. 4. Interfáz gráfica desarrollada para el proyecto.

Implementación física

Los sensores RFID se comunican con el Arduino UNO mediante el protocolo SPI. Un maestro SPI se comunica con los esclavos SPI según el siguiente esquema.

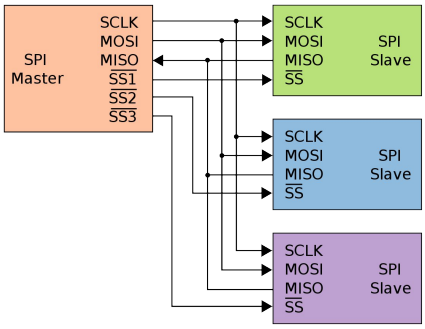

Fig. 5. Maestro SPI con multiples Esclavos SPI.

El circuito implementado sigue el esquema general de maestro y multiples esclavos SPI, pero fué adaptado a los pines de los sensores RFID.

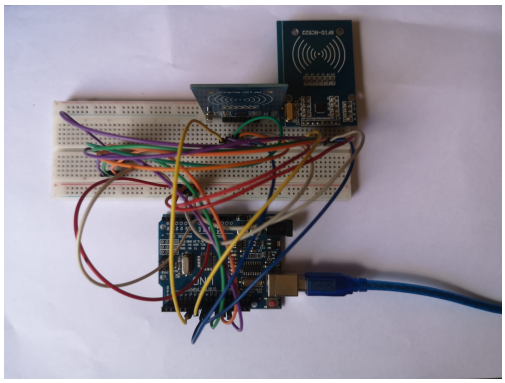

Fig. 6. Montaje del circuito en Arduino UNO.

Finalmente, se conectó el circuito al Raspberry Pi 3, junto a un monitor y teclado. Para el funcionamiento correcto en Raspberry de los códigos desarrollados, se instalaron las librerías **pymysql** y **pandas**.

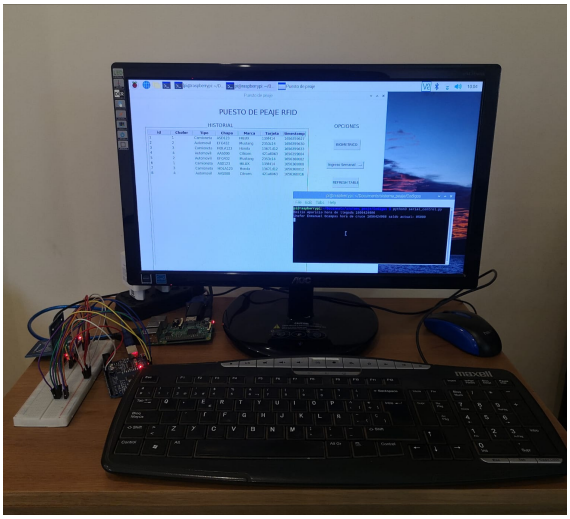

Fig. 7. Montaje del Raspberry

SECCION 3: RESULTADOS

I. CONCLUSION

Se determinaron las entidades y las relaciónes entre ellas para el problema planteado. Se diseñó e implementó exitosamente la base de datos en SQL cumpliendo con los requisitos preliminares. Se desarrolló un script funcional capaz de gestionar la base de datos, con los datos de ingreso de empleados y cruce de automóviles. Finalmente, se implementó el sistema funcional en Raspberry Pi 3, cumpliendo con los objetivos generales y específicos.

II. REFERENCIAS

[1]Espinoza Reyes, C. E. (2017). Diseño de un sistema de cobro automatizado en el Peaje Chillón, para mejorar el tráfico vehicular en la Panamericana Norte, utilizando Tecnología RFID.

[2]Cárdenas Rodríguez, J. C., Melendez Mendoza, L., & Rafaile Solano, W. (2018). Diseño de un sistema de control de tráfico y peaje vehicular en la ciudad de lima utilizando tecnología RFID de ultra alta frecuencia.

III. ANEXOS

1. CÓDIGOS UTILIZADOS

a. BaseDatos.py

```
        import mysql.connector
        import pandas as pd
        def printProgressBar (iteration, total, prefix = '', suffix = '', decimals = 1, length = 100, fill = '█', printEnd = "\r"):
            percent = ("{0:." + str(decimals) + "f}").format(100 * (iteration / float(total)))
            filledLength = int(length * iteration // total)
            bar = fill * filledLength + '-' * (length - filledLength)
            print(f'\r{prefix} |{bar}| {percent}% {suffix}', end = printEnd)
        def create_database(**kw):
            db_name=kw.get('db_name',None)
            conexion1=mysql.connector.connect(host="localhost", user="root", passwd="password", database= "" )
            cursor1=conexion1.cursor()
            line="CREATE DATABASE "+db_name
            try:
                print(line)
                cursor1.execute(line)
                conexion1.commit()
                print("The database was created successfully...")
            except Exception as e: None
            conexion1.close()
        def create_table(**kw):
            table,var,db_name,line=kw.get('table',None),kw.get('var',[]),kw.get('db_name',None),""
            for a in var: line+=", "+a
            line="CREATE TABLE "+table+" (id INT AUTO_INCREMENT PRIMARY KEY"+line+");"
            conexion1=mysql.connector.connect(host="localhost",user="root",passwd="password",database=db_name)
            cursor1=conexion1.cursor()
            try:
                print(line)
                cursor1.execute(line)
                conexion1.commit()
                print("The SQL table was created successfully...")
            except Exception as e: None
            conexion1.close()
        def create_relation(**kw):

table1,table2,key1,key2,name,db_name=kw.get('table1',None),kw.get('table2',None),kw.get('key1',None),kw.get('key2',None),kw.get('name',None),kw.get
('db_name',None)
            line='ALTER TABLE '+table1+' ADD CONSTRAINT '+name+' FOREIGN KEY ('+key1+') REFERENCES '+table2+'('+key2+') ON DELETE CASCADE ON
UPDATE CASCADE;'
            try:
                print(line)
                conexion1=mysql.connector.connect(host="localhost", user="root", passwd="password", database=db_name )
                cursor1=conexion1.cursor()
                cursor1.execute(line)
                conexion1.commit()
                print('The SQL relation was added successfully...')
            except Exception as e:None
            conexion1.close()
        def csv_to_sql(**kw):
            path=kw.get('path',None)
            table=kw.get('table',None)
            df=pd.read_csv(path);print(df.head());
            var=df.columns.to_list()
            string="INSERT INTO "+table+" "+str(tuple(var)).replace("'","")+" VALUES "+"("+"{},"*(len(var)-1)+"{})"
            conexion1=mysql.connector.connect(host="localhost", user="root", passwd="password", database= "sistema_peaje" )
            cursor1=conexion1.cursor()
            l=len(df.index)
            dat,q='',0
            for v in var:
                exec('c'+str(q)+'=df["'+v+'"].to_list()')
                dat+=',c'+str(q)+'[k]';q+=1
            dat=dat[1:]
            print(df);print('Loading data...\n')
            printProgressBar(0,l,prefix='Progress:',suffix='Complete',length=50)
            for k in range(0,l):
                try:
                    cursor1.execute(eval('string.format('+dat+')'))
                    conexion1.commit()
                except Exception as e: print(e)
                printProgressBar(k+1,l,prefix='Progress:',suffix='Complete',length = 50)
            conexion1.close()
        def drop_database(**kw):
            db_name=kw.get('db_name',None)
            conexion1=mysql.connector.connect(host="localhost", user="root", passwd="password", database= db_name )
            cursor1=conexion1.cursor()
            line='DROP DATABASE '+db_name
            try:
                print(line)
                cursor1.execute(line)
                conexion1.commit()
                print('The database was dropped successfully...')
            except Exception as e:print(e)
            conexion1.close()
        drop_database(db_name='sistema_peaje')
        create_database(db_name='sistema_peaje')
        create_table(table='Biometrico',var=['id_empleado INT','timestamp BIGINT'],db_name='sistema_peaje')
        create_table(table='Empleado',var=['nombre VARCHAR(10)','apellido VARCHAR(10)','CIN BIGINT','puesto VARCHAR(10)','tarjeta
VARCHAR(8)'],db_name='sistema_peaje')
        create_table(table='Automovil',var=['id_chofer INT','tipo VARCHAR(10)','chapa VARCHAR(10)','marca VARCHAR(10)','tarjeta
VARCHAR(8)'],db_name='sistema_peaje')
        create_table(table='Registro',var=['id_auto INT','timestamp BIGINT','monto BIGINT'],db_name='sistema_peaje')
        create_table(table='Chofer',var=['nombre VARCHAR(10)','apellido VARCHAR(10)','CIN BIGINT','saldo BIGINT'],db_name='sistema_peaje')
        create_relation(table1='Registro',table2='Automovil',key1='id_auto',key2='id',db_name='sistema_peaje',name='registros')
        create_relation(table1='Automovil',table2='Chofer',key1='id_chofer',key2='id',db_name='sistema_peaje',name='dueño')
        create_relation(table1='Biometrico',table2='Empleado',key1='id_empleado',key2='id',db_name='sistema_peaje',name='firma')
        csv_to_sql(path='/home/lucas/FIUNA/9no_semestre/BD/Proyecto/Codigos/datasets/choferes.csv',table='Chofer')
        csv_to_sql(path='/home/lucas/FIUNA/9no_semestre/BD/Proyecto/Codigos/datasets/autos.csv',table='Automovil')
        csv_to_sql(path='/home/lucas/FIUNA/9no_semestre/BD/Proyecto/Codigos/datasets/empleados.csv',table='Empleado')
```

b. gui_manager.py

```
import mysql.connector
import pandas as pd
import time
import datetime as datetime
import tkinter as tk
from tkinter import ttk
global dsd,hst,header,result
dsd=datetime.datetime.today().replace(minute=0, hour=0, second=0, microsecond=0)
hst=dsd+datetime.timedelta(hours=24)
dsd,hst=str(dsd),str(hst)
header=('id','Nombre','Apellido','CIN','Puesto','Tarjeta','timestamp')
#FUNCTIONS
def data_get(**kw):
    table=kw.get('table',None)
    var=kw.get('var',None)
    desde=kw.get('desde',None)
    hasta=kw.get('hasta',None)
    desde=int(datetime.datetime.strptime(desde,"%Y-%m-%d %H:%M:%S").timestamp())
    hasta=int(datetime.datetime.strptime(hasta,"%Y-%m-%d %H:%M:%S").timestamp())
    conexion1=mysql.connector.connect(host="localhost", user="root", passwd="password", database= "sistema_peaje" )
    cursor1=conexion1.cursor()
    df=pd.DataFrame(columns=var)
    for item in var:
        try:
            cursor1.execute("SELECT "+item+" FROM "+table+" WHERE timestamp>= "+str(desde)+" AND timestamp< "+str(hasta)+" ORDER BY timestamp ASC")
            df[item]=[i[0] for i in cursor1.fetchall()]
        except Exception as e:print(e)
    conexion1.close()
    return df
def toggle():
    global dsd,hst,header,result
    for i in result.get_children():result.delete(i)

table,related,var,var2,t,st,header='Biometrico','Empleado',['id','id_empleado','timestamp'],'nombre,apellido,CIN,puesto,tarjeta','HISTORIAL','BIOMETRICO',('id','Nombre','Apellido','CIN','Puesto','Tarjeta','timestamp')
    if(reg_bio.config('text')[-
1]=='HISTORIAL'):table,related,var,var2,t,st,header='Registro','Automovil',['id','id_auto','timestamp'],'id_chofer,tipo,chapa,marca,tarjeta','BIOMETRICO','HISTORIAL',('id','Chofer','Tipo','Chapa','Marca','Tarjeta','timestamp')
    else:None
    label2.configure(text=st)
    reg_bio.configure(text=t)
    refresh(table,related,var,var2,header)
def refresh(table,related,var,var2,header):
    df=data_get(table=table,desde=dsd,hasta=hst,var=var)
    id_df=df[var[0]].to_list()
    id_list=df[var[1]].to_list()
    time_list=df[var[2]].to_list()
    conexion1=mysql.connector.connect(host="localhost", user="root", passwd="password", database= "sistema_peaje" )
    cursor1=conexion1.cursor()
    data=[]
    for k in range(len(header)):result.heading(str(k),text=header[k])
    for i in range(0,len(id_list)):
        cursor1.execute("SELECT "+var2+" FROM "+related+" WHERE id="+str(id_list[i]))
        data.append((id_df[i],)+cursor1.fetchall()[0]+(time_list[i],))
    for d in data:result.insert('',tk.END,values=d)
    conexion1.close()
def update():
    global dsd,hst,header,result
    for i in result.get_children():result.delete(i)

table,related,var,var2,header='Biometrico','Empleado',['id','id_empleado','timestamp'],'nombre,apellido,CIN,puesto,tarjeta',('id','Nombre','Apellido','CIN','Puesto','Tarjeta','timestamp')
    if(reg_bio.config('text')[-
1]=='BIOMETRICO'):table,related,var,var2,header='Registro','Automovil',['id','id_auto','timestamp'],'id_chofer,tipo,chapa,marca,tarjeta',('id','Chofer','Tipo','Chapa','Marca','Tarjeta','timestamp')
    else:None
    refresh(table,related,var,var2,header)
def chart_change(choice):
    global dsd,hst
    today=datetime.datetime.today().replace(minute=0, hour=0, second=0, microsecond=0)
    if choice=='Ingreso Diario':
        dsd=today
        hst=today+datetime.timedelta(hours=24)
    elif choice=='Ingreso Semanal':
        dsd=today-datetime.timedelta(days=7)
        hst=today+datetime.timedelta(hours=24)
    elif choice=='Ingreso Mensual':
        dsd=today-datetime.timedelta(days=30)
        hst=today+datetime.timedelta(hours=24)
    dsd,hst=str(dsd),str(hst)
#GRAPHIC VARIABLES
w,h=1000,750
bkc='white'
dx,dy=50,50
#ROOT SCREEN CANVAS
root=tk.Tk()
root.title('Muon Matrix control panel')
canvas=tk.Canvas(root,width=w,height=h,relief='raised')
canvas.configure(background=bkc)
canvas.pack()
#GEOMETRIC FIGURES
canvas.create_rectangle(dx,2*dy+10,200+w/2,h-dy)
#LABELS
label1=tk.Label(root,text='PUESTO DE PEAJE RFID',font=('Ubuntu',20),background=bkc)
label2=tk.Label(root,text='HISTORIAL',font=('Ubuntu',14),background=bkc)
label3=tk.Label(root,text='OPCIONES',font=('Ubuntu',14),background=bkc)
canvas.create_window(w/2,dy,window=label1)
canvas.create_window(w/4+100,2*dy,window=label2)
canvas.create_window(3*dx+w/2+200,2*dy,window=label3)
#DROPDOWN BARS
options=["Ingreso Diario","Ingreso Semanal","Ingreso Mensual"]
var=tk.StringVar(root);var.set("Ingreso Diario")
bar_selec=tk.OptionMenu(root,var,*options,command=chart_change)
canvas.create_window(7*dx+w/2,5*dy,window=bar_selec)
#BUTTONS
reg_bio=tk.Button(text='BIOMETRICO',width=10,command=toggle,pady=10)
ref_but=tk.Button(text='REFRESH TABLE',width=10,command=update,pady=10)
reg_bio.place(x=6*dx+w/2,y=3*dy)
ref_but.place(x=6*dx+w/2,y=6*dy)
#TREEVIEW
result=ttk.Treeview(root,columns=('0','1','2','3','4','5','6'),show='headings',height=27)
for k in range(0,len(header)):
    result.heading(str(k),text=header[k])
    result.column(str(k),minwidth=0,width=90,stretch='no')
canvas.create_window(7.5*dx,8*dy,window=result)
#MAIN LOOP
root.mainloop()
```

c. serial_control.py

```
        import serial
        import time
        import mysql.connector
        try:
            string_auto="INSERT INTO Registro (id_auto,timestamp,monto) VALUES ({},{},{})"
            string_biom="INSERT INTO Biometrico (id_empleado,timestamp) VALUES ({},{})"
            conexion1=mysql.connector.connect(host="localhost", user="root", passwd="password", database= "sistema_peaje" )
            cursor1=conexion1.cursor()

ser=serial.Serial(port='/dev/ttyUSB0',baudrate=9600,parity=serial.PARITY_NONE,stopbits=serial.STOPBITS_ONE,bytesize=serial.EI
GHTBITS,timeout=0)
            while True:
                if ser.inWaiting():
                    ct=str(int(time.time()))
                    res=ser.readline(ser.in_waiting).decode('utf-8').replace('\n','').replace('\r','').split(',')
                    ser.reset_input_buffer()
                    if res[0]=="empleado":
                        try:
                            cursor1.execute("SELECT id,nombre,apellido FROM Empleado WHERE tarjeta='"+res[1]+"'")
                            ret=[i for i in cursor1.fetchall()[0]]
                            cursor1.execute(string_biom.format(ret[0],ct))
                            conexion1.commit()
                            print(ret[1]+' '+ret[2]+' hora de llegada '+ct)
                        except:print('Usuario no registrado...')
                    elif res[0]=="automovil":
                        try:
                            cursor1.execute("SELECT id,id_chofer,chapa FROM Automovil WHERE tarjeta='"+res[1]+"'")
                            ret=[i for i in cursor1.fetchall()[0]]
                            cursor1.execute("SELECT saldo FROM Chofer WHERE id="+str(ret[1]))
                            saldo=cursor1.fetchall()[0][0]
                            if saldo>0:
                                cursor1.execute("UPDATE Chofer SET saldo=saldo-5000 WHERE id="+str(ret[1]))
                                conexion1.commit()
                                cursor1.execute(string_auto.format(ret[0],ct,'5000'))
                                conexion1.commit()
                                cursor1.execute("SELECT nombre,apellido,saldo FROM Chofer WHERE id='"+str(ret[1])+"'")
                                ret=[i for i in cursor1.fetchall()[0]]
                                print('Chofer '+ret[0]+' '+ret[1]+' hora de cruce '+ct+' saldo actual: '+str(ret[2]))
                            else:print('Saldo insuficiente...')
                        except:
                            print('Automovil no registrado...')
                    time.sleep(2)
                time.sleep(0.1)
        except Exception as e:print(e)
        try:
            ser.close()
            conexion1.close()
        except:None
```

d. senosr.ino

```
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN_e  6
#define SS_PIN_a  7
#define RST_PIN_e 8
#define RST_PIN_a 9
MFRC522 mfrc522_e(SS_PIN_e, RST_PIN_e);
MFRC522 mfrc522_a(SS_PIN_a, RST_PIN_a);
String tarjeta;
void setup() {
         Serial.begin(9600);
         SPI.begin();
  pinMode(RST_PIN_e,OUTPUT);
  pinMode(RST_PIN_a,OUTPUT);
         mfrc522_e.PCD_Init();
  mfrc522_a.PCD_Init();
}
void loop(){
         digitalWrite(RST_PIN_e,HIGH);
  digitalWrite(RST_PIN_a,LOW);
         if ( mfrc522_e.PICC_IsNewCardPresent()) {
    tarjeta="";
    if ( mfrc522_e.PICC_ReadCardSerial()){
      for (byte i=0;i<mfrc522_e.uid.size;i++){
        tarjeta+=String(mfrc522_e.uid.uidByte[i],HEX);
      }
      Serial.print("empleado,"+tarjeta);
      mfrc522_e.PICC_HaltA();
    }
         }
  digitalWrite(RST_PIN_e,LOW);
  digitalWrite(RST_PIN_a,HIGH);
  if( mfrc522_a.PICC_IsNewCardPresent()) {
    tarjeta="";
    if ( mfrc522_a.PICC_ReadCardSerial()){
      for (byte i=0;i<mfrc522_a.uid.size;i++){
        tarjeta+=String(mfrc522_a.uid.uidByte[i],HEX);
      }
      Serial.print("automovil,"+tarjeta);
      mfrc522_a.PICC_HaltA();
    }
  }
}
```

2. DOCUMENTACION

a. Plan de trabajo

PLAN DE TRABAJO - PROYECTO FINAL DE BASE DE DATOS - SISTEMA DE PEAJE POR RADIO FRECUENCIA (RFID)											
Etapa	Fechas de avances										
	Martes 29/03	Martes 05/04	Martes 12/04	Martes 19/04	Martes 26/04	Martes 03/05	Martes 10/05	Martes 17/05	Martes 31/05	Martes 07/06	Martes 14/06	Martes 21/06
Conceptualizacion	Justificacion Objetivos Estado del arte											
Diseño		Entidades principales Propiedades Relaciones Modelo lógico inicial	Posibles expansiones Entidades posibles Modelo lógico expandido Flujograma del GUI									
Implementacion				Creacion de tablas en SQL Programacion del GUI en python								
Montaje								Montaje en RASPERRY				
Finalizacion												Entrega

b. Tabla de presupuestos

Componente	Proveedor	Cantidad	Precio unitario	Precio final
Raspberry Pi 3	Electrónica Plett	1	365.000	365.000
Arduino UNO	Electrónica Plett	1	80.000	80.000
Sensor RFID RC522	Electrónica Plett	2	55.000	110.000
Kit de tarjetas RFID	Electrónica Plett	1	40.000	40.000
			TOTAL	595.000 Gs.

INDICE

INTRODUCCION 1
SECCIÓN 1: GENERALIDADES DEL PROYECTO 1
I. PLANTEAMIENTO DEL PROBLEMA 1
II. OBJETIVOS GENERALES Y ESPECÍFICOS 1
SECCION 2: MARCO TEÓRICO 1
I. MATERIALES Y MÉTODOS 2
II. DESARROLLO 2
Diseño de la base de datos 2
Desarrollo de códigos 2
Implementación física 3
SECCION 3: RESULTADOS 3
I. CONCLUSION 3
II. REFERENCIAS 3
III. ANEXOS 4
1. CÓDIGOS UTILIZADOS 4
2. DOCUMENTACION 8