Julian Andres Ramirez Jimenez – Código: 202010051010

Proyecto final – Teoría de la conmutación – Grupo 004

**Sistema de control de acceso con Internet de las cosas**

**Link Repositorio:** https://github.com/jaramirezj/ProyectoIoT

**Descripción de la idea de proyecto**

El proyecto surge de la idea de controlar el acceso a ciertos lugares, como pueden ser parqueaderos, aulas, sitios de trabajo entre otros. Un elemento muy importante de este proyecto es que busca facilitarle la tarea al administrador del lugar para controlar el acceso de una forma muy sencilla, intuitiva y funcional. Por esto a continuación se describen algunas funcionalidades del sistema:

- Registrar nuevos usuarios.

- Asignar o denegar permisos de acceso a los usuarios que el desee.

- Visualizar cuantas personas/vehículos se encuentran actualmente en el lugar.

- Modificar la cantidad de personas/vehículos que pueden acceder.

-Cerrar completamente el acceso al lugar.

-Liberar completamente el acceso lugar

Dentro de estas funcionalidades destaca que se puedan registrar usuarios con datos relacionados con la tarjeta de acceso y la contraseña, y gracias a una verificación de estas credenciales y las demás variables que intervienen en el problema, se les permita acceder o no.

**Descripción de la solución**

Como solución se desarrolló un sistema de IoT que consiste básicamente en 3 módulos:

* Control de Arduino y electrónica de los sensores y actuadores
* Programación de raspberry.
* Aplicación web del administrador

La idea básica de este sistema es que el usuario antes de ingresar al lugar ponga su tarjeta en el lector e ingrese su código mediante el teclado, todo esto se validará realizando una petición a la raspberry que a su vez hará una petición a la aplicación web, y según lo que se ha establecido previamente el administrador, la aplicación web retornará una respuesta.

**Control de Arduino y electrónica de los sensores y actuadores**

Mediante el Arduino y los demás componentes electrónicos usados se logran obtener los datos del acceso, así como materializar el control que ejerce el administrador sobre el sistema. Para este proyecto se utilizó un Arduino UNO que está programado en el lenguaje propio (C++ para Arduino), además de Arduino se utilizó un sensor lector de tarjetas RFID, en este caso el rc522; también se hizo uso de un teclado matricial 4x4, una pantalla LCD con I2C, un servomotor, un sensor infrarrojo, un pulsador y algunas resistencias.

El programa de Arduino tiene el siguiente flujo:

1. Inicialmente se revisa el estado del parqueadero, esto se hace imprimiendo ‘estado’ por el puerto serial para esperar una respuesta de la raspberry. Al recibir la respuesta se verifica el dato y se le asigna a la variable global “estado’ un número que puede ser 1,2 o 3.

Para mandarle la solicitud a la raspberry: Serial.println(‘estado’);

Para leer el dato por serial: String respuesta = Serial.readStringUntil('\n');

1. En cuanto se recibe la respuesta se procede a ejecutar el programa, pero este se ejecuta según el estado, como se describe a continuación:

2.1.La primera posible respuesta hace que el programa trabaje en el modo normal, es decir se muestra en la pantalla lcd el mensaje ‘Ingrese una tarjeta’(lcd.print("Ingrese una"); lcd.setCursor(0,1); lcd.print("tarjeta");) y se espera que alguien acerque una tarjeta al lector rfid, así:

if (!mfrc522.PICC\_IsNewCardPresent()) {return false;}

if (!mfrc522.PICC\_ReadCardSerial()) {return false; }

Estas dos condiciones revisan si el sensor aún no ha leído ninguna tarjeta, en el caso que alguien ingrese la tarjeta no se cumplirán estas condiciones y continuará la función que realiza la lectura del uid de la tarjeta mediante el siguiente código:

for (byte i = 0; i < mfrc522.uid.size; i++) {

Serial.print(mfrc522.uid.uidByte [i] < 0x10 ? " 0" : " ");

Serial.print(mfrc522.uid.uidByte [i], HEX);

}

Luego de que se lee la tarjeta se imprime ‘Ingrese codigo’ por la pantalla lcd , se pide que se ingrese una contraseña mediante el teclado matricial( char key = keypad.getKey();) y se espera hasta que se pulse el botón de envío(while(digitalRead(pinEnter)!=1){}), con esto se enviaran los datos por serial a la raspberry, imprimiendo primero ‘uid’,luego el id leído de la tarjeta, después ‘pass’ y finalmente el código ingresado. Después de esto se esperará una respuesta por serial de la raspberry, y dependiendo de esta respuesta el Arduino sabrá si permite el acceso o no. Las posibles respuestas que se pueden recibir por terminal son:

-‘correct’ ->Se permite el acceso, se imprime por el lcd ‘Puede ingresar’ y el servomotor se mueve(servoMotor.write(90);) permitiendo que se acceda, luego cuando se ingrese al lugar un sensor infrarrojo lo detecta (while(digitalRead(pinIR)!=0){}) y cuando se detecta el ingreso el servo vuelve a su posición normal.

-‘wrong\_card’->Se deniega el acceso y se muestra por la pantalla lcd ‘Tarjeta no reconocida’.

-‘wrong\_pass’->Se deniega el acceso y se muestra por la pantalla lcd ‘Codigo incorrecto’.

-‘full’-> Se deniega el acceso y se muestra por la pantalla lcd ‘Parqueadero lleno’.

-‘denied’-> Se deniega el acceso y se muestra por la pantalla lcd ‘Acceso denegado’.

2.2.La segunda respuesta posible hace que el programa bloquee completamente el acceso, simplemente imprimiendo por la pantalla lcd ‘Acceso bloqueado’.

2.3.La tercera respuesta posible libera completamente el acceso, haciendo que no se exijan credenciales para acceder al lugar y se imprime por la pantalla lcd ‘Acceso libre’.

De forma paralela a estos procesos se está revisando cuando alguien sale del lugar, esto se hace mediante el sensor infrarrojo(if(digitalRead(pinIR)==0))y en caso de que se detecte que alguien sale se imprime por serial un mensaje indicando ‘salida’ que recibirá la raspberry.

Finalmente, cada cierto tiempo se va actualizando el estado del acceso como se describió al principio.

**Programación de raspberry**

Se hizo uso de una raspberry pi 3 que se programó en Python, básicamente esta sirve como un puente entre el Arduino y el servidor donde está alojada la aplicación web del administrador. La idea es básicamente la siguiente: El Arduino envía mensajes por serial a la raspberry, esta hace una petición al servidor, recibe la respuesta y se la retorna al Arduino por serial. Para conectar la raspberry al serial se utiliza la librería serial y el codigo( with serial.Serial("/dev/ttyUSB0",9600,timeout=1) as ser:). Primero la raspberry recibe los mensajes deñ arduino de la siguiente forma: (read = ser.readline().decode() o también rfid = ser.readline().decode('utf-8').strip()) y dependiendo del mensaje que llegue se realiza una petición GET al servidor así:

URL = "http://ip.../ProyectoIoT/controller/IOTcontroller.php"

PARAMS = {'rfid':rfid, 'pass':password}

r = requests.get(url = URL, params = PARAMS)

para que posteriormente el servidor de una respuesta a la raspberry(respuesta = r.text), esta respuesta será enviada al Arduino mediante el serial(ser.write(respuesta.encode())).

En este caso hay 3 posibles mensajes que recibe la raspberry desde el Arduino. El primero corresponde a los datos del identificador de la tarjeta y la contraseña, en donde se envían estos datos y se le pasa la respuesta del servidor del Arduino. El segundo posible mensaje es el que se da indicando que alguien salió del lugar es decir la cadena ‘salida’, en este caso se le envía el servidor a este mensaje para que actualice la cantidad de personas en el lugar. El tercer posible mensaje es la consulta del estado con la cadena ‘estado’, en este caso se le envía al servidor y se recibe la respuesta, para luego enviarla por serial al Arduino.

**Aplicación web del administrador**

Los lenguajes utilizados para desarrollar la aplicación web fueron PHP en cuanto a la lógica, además de HTML, CSS, JavaScript, el framework Bootstrap y la librería DataTables JS, y en cuanto a la base de datos se utilizó MySQL. Un dato importante es que esta aplicación se desplegó en un servidor local llamado XAMPP.

Esta aplicación web consiste básicamente en dos pantallas, en la primera se da cierta información sobre el proyecto, se puede gestionar el estado del lugar, entre las opciones Activo, Bloqueado y Acceso libre, además de la cantidad de personas que pueden ingresar y se muestran cuantas personas se encuentran actualmente en el lugar. Desde esta pantalla se puede acceder a la segunda, en la cual se gestionan los usuarios, allí se muestran los usuarios que se encuentran registrados en el sistema, se pueden eliminar usuarios, bloquear o activar el acceso a diferentes usuarios y registrar nuevos usuarios con su respectiva identificación, nombre y apellidos, id de tarjeta y contraseña.

Además de la parte que puede visualizar el administrador, hay un programa en especifico que es fundamental para el correcto funcionamiento del proyecto, este es el controlador de IoT. En este programa el servidor está atento a las peticiones HTTP GET que hace la raspberry, en el caso que llegue alguna se da respuesta al hacer la validación en la base de datos mediante alguno de los métodos de la clase Conexion. Primero se revisa si llega alguna de las siguientes peticiones GET

if(isset($\_GET['salida']))

if(isset($\_GET['rfid'])&&isset($\_GET['pass']))

if(isset($\_GET['estado']))

En el caso que llegue alguna petición se realiza la conexión con la base de datos, se recibe el mensaje que mande la validación y se envía a la raspberry:

$conexion = new Conexion();

$resultado = $conexion->verificarCredenciales($tarjeta,$pass);

 header('Content-type: text/html');

  echo $resultado;

Otra ejemplo puede ser:

header('Content-type: text/html');

        echo $conexion->verificarEstado()

Tanto para este controlador de IOT, como para las pantallas a las que accede el administrador se realizan operaciones y consultas sobre la base de datos, para esto se tiene la clase Conexión, en la que se realiza tanto la conexión como las operaciones. El código de conexión a la base de datos es el siguiente:

public function \_\_construct(){

            $this->conexion = new mysqli("localhost","root","","sca");

}

Dentro de esta clase se realizan diferentes SQL QUERYS, tanto de consulta, inserción, actualización y eliminación. Un ejemplo de la estructura general que se utiliza en los métodos de esta clase es la siguiente:

$consulta = "CONSULTA SQL”

$resultado = $this->conexion->query($consulta);

            if($resultado){

               retornar un Mensaje o

retornar que la consulta fue exitosa

Básicamente se tiene una consulta, se le dice a la conexión que la ejecute y finalmente según el resultado se retorna true , false o un mensaje especifico.

**Arquitectura del proyecto**

A continuación, se muestra una representación de la arquitectura del proyecto mediante un diagrama de despliegue

Diagrama

Descripción generada automáticamente

Link del diagrama: https://drive.google.com/file/d/1P7u2UDs2HM6ZEONgGCRpV0ijifRqYf8m/view?usp=sharing

**Modelo de datos**

Diagrama

Descripción generada automáticamente

Esta base de datos consta básicamente de dos tablas, que son parqueadero y usuario, cada parqueadero tiene un id único, además de un estado representado numéricamente donde 1=”Estado normal”, 2=”Bloqueado”, 3=”Acceso libre”. El campo ‘maximo’ representa la cantidad máxima de personas/vehículos que pueden acceder al lugar y ‘actual’ corresponde a la cantidad actual de personas/vehículos dentro del parqueadero. En cuanto al usuario se tiene ‘idusuario’ que corresponde a la cedula del respectivo usuario, además de ‘idrfid’, que es el id de la tarjeta para este usuario, una contraseña o código, los nombres y apellidos, un ‘estado\_actual’ indicando si se tiene permitido el acceso o no, ‘registrado’ que corresponde a la fecha de registro de este usuario y finalmente el id del parqueadero donde está registrado este usuario.(En la base de datos se manejó como parqueadero, pero como se mencionó anteriormente puede ser un aula, sitio de trabajo, entre otros).

**Capturas de pantalla de la aplicación web:**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Algunas fotos del montaje electrónico**

Una computadora en una mesa

Descripción generada automáticamente con confianza baja

Imagen que contiene electrónica, tabla, computadora

Descripción generada automáticamente

