Universidad San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Arquitectura de Computadores y Ensambladores 1

Proyecto 1 de Laboratorio

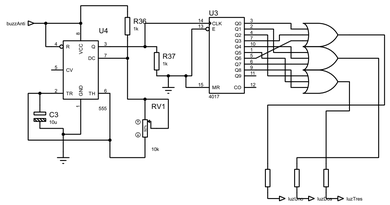
Brayan Alexander Mejia Barrientos – 201900576
Estuardo Gabriel Son Mux – 202003894
Fabian Esteban Reyna Juárez – 202003919
Angel Eduardo Marroquín Canizales – 202003959
Diego Andre Mazariegos Barrientos – 202003975
Javier Alejandro Gutierrez de León – 202004765
Wilber Steven Zúñiga Ruano – 202006629

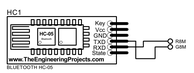
Arduino usado: Arduino Mega 2560

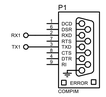
Arduino Mega es una tarjeta de desarrollo open-source construida con un microcontrolador modelo Atmega2560 que posee pines de entradas y salidas (E/S), analógicas y digitales. Esta tarjeta es programada en un entorno de desarrollo que implementa el lenguaje Processing/Wiring. Arduino puede utilizarse en el desarrollo de objetos interactivos autónomos o puede comunicarse a un PC a través del puerto serial (conversión con USB) utilizando lenguajes como Flash, Processing, MaxMSP, etc. Las posibilidades de realizar desarrollos basados en Arduino tienen como límite la imaginación. (Arduino, 2022)

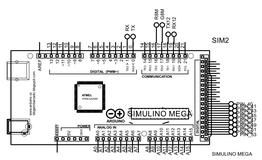
Componentes Usados

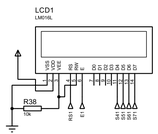
- 2 Arduino Mega 2560 (Debido a la cantidad de pines usados)
- 1 módulo Bluetooth HC – 05
- 1 conector serial
- 1 pantalla LCD
- 17 Resistencia de 10k
- 2 motores stepper
- Varias luces LED
- 2 Buzzer
- 16 Sensores de luz

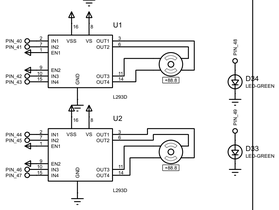


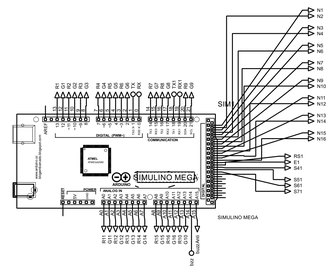




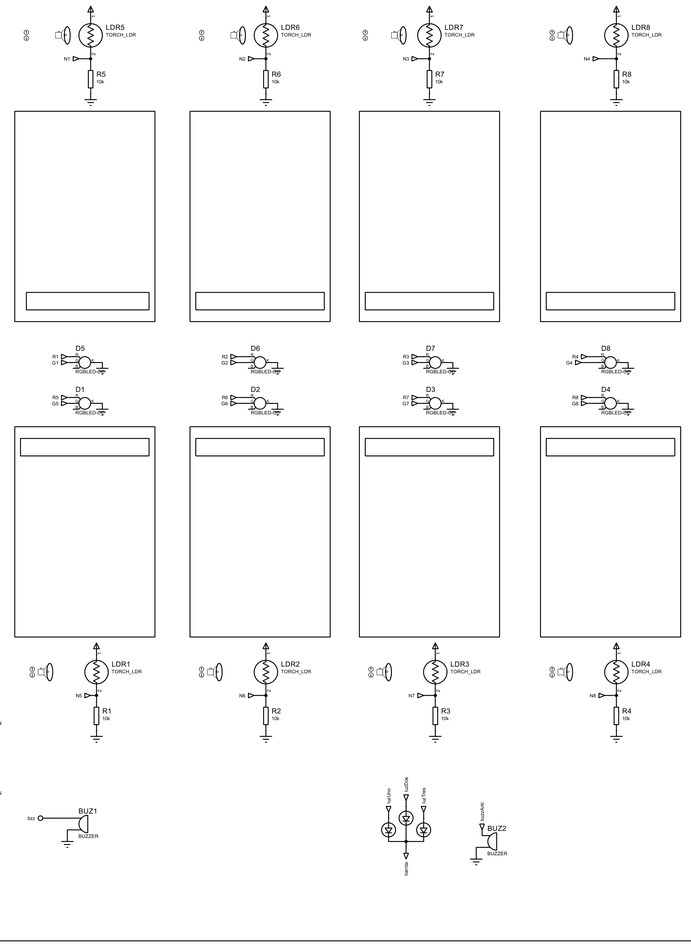




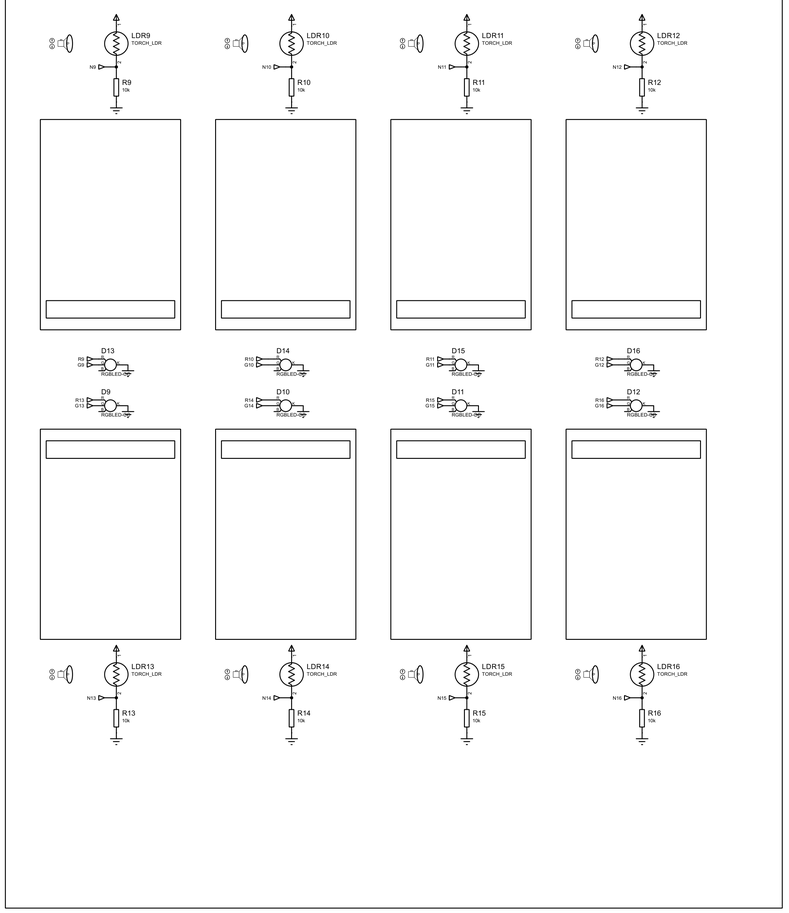




NIVEL 1



NIVEL 2



Backend

Elaborado en node js, escrito en TypeScript y convertirlo automáticamente a JavaScript para su ejecución, esta se consume gracias al servicio de AWS

Librerías y versiones (package.json)

```json
{
    "name": "backend",
    "version": "1.0.0",
    "description": "",
    "main": "index.js",
    ▷ Debug
    "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1",
        "start": "nodemon build/index",
        "build": "tsc -w",
        "server": "node build/index"
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "dependencies": {
        "body-parser": "^1.20.1",
        "cors": "^2.8.5",
        "express": "^4.18.2",
        "morgan": "^1.10.0",
        "nodemon": "^2.0.20",
        "tsc": "^2.0.4"
    },
    "devDependencies": {
        "@types/cors": "^2.8.13",
        "@types/express": "^4.17.14",
        "@types/morgan": "^1.9.3"
    }
}
```

index.ts

```ts
import express, { Application } from 'express';
import morgan from 'morgan';
import cors from 'cors';
import body_parser from 'body-parser';
import indexRouter from './Routes/indexRouter';
```

Imports necesarios

```
class servidor {
    public app: Application;
    constructor() {
        this.app = express();
        this.config();
        this.routes();
    }

    config(): void {
        this.app.set('port', process.env.PORT || 4000);
        this.app.use(morgan('dev'));
        this.app.use(express.urlencoded({ extended: false }));
        this.app.use(express.json());
        this.app.use(express.json({ limit: '50mb' }));
        this.app.use(express.urlencoded({ limit: '50mb' }));
        this.app.use(cors());
        this.app.use(body_parser.urlencoded({ extended: true }))
    }

    routes(): void {
        this.app.use("/", indexRouter)
    }

    start(): void {
        this.app.listen(this.app.get('port'), () => {
            console.log('Server On Port ', this.app.get('port'))
        });
    }
}

export const servidorr = new servidor();
servidorr.start();
```

Clase servidor, creación de su constructor, configuración, rutas, puerto e inicialización del servidor

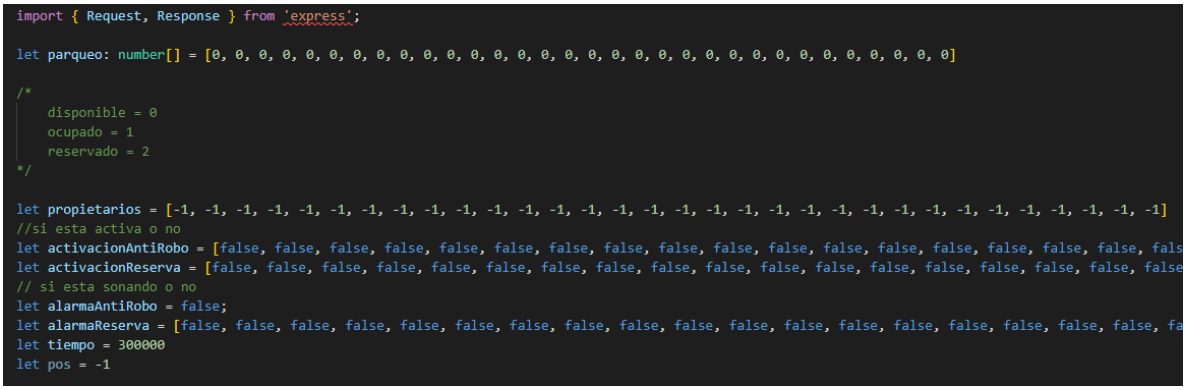
Routes/indexRouter.ts

Creación de las rutas (URL's) a utilizar

```
import { Router } from "express";
import { indexController } from "../Controllers/indexController";


class IndexRouter {
    public router: Router = Router();;
    constructor() {
        this.config();
    }

    config(): void {
        this.router.get("/", indexController.prueba)
        this.router.get("/verParqueo", indexController.verParqueo)
        this.router.post("/setParqueo", indexController.setParqueo)
        this.router.get("/estado", indexController.estadoParqueo)
        this.router.get("/existencia", indexController.hayExistencias)
        this.router.post("/reservar", indexController.reservarParqueo)
        this.router.post("/desahbilitarAlarmaReserva", indexController.DeshabilitarAlarmaReserva)
        this.router.post("/sonidoAlarmaReserva", indexController.SonidoAlarmaReserva)
        this.router.post("/habilitarAlarmaAntiRobo", indexController.HabilitarAlarmaAntiRobo)
        this.router.post("/deshabilitarAlarmaAntiRobo", indexController.DeshabilitarAlarmaAntiRobo)
        this.router.post("/sonidoAlarmaAntirobo", indexController.SonidoalarmaRobo)
        this.router.post("/login", indexController.login)
        this.router.post("/ajusteTiempo", indexController.ajusteTiempo)
        this.router.get("/getAlarmaAntiRobo", indexController.getAlarmaAntiRobo)
        this.router.get("/getactivacionAntiRobo", indexController.getActivacionAntiRobo)
        this.router.get("/getactivacionReserva", indexController.getActivacionReserva)
    }
}

const indexRouter = new IndexRouter();
export default indexRouter.router;
```

Controllers/indexController.ts

Controlador de la API contiene las acciones y los arrays correspondiente a los parqueos.

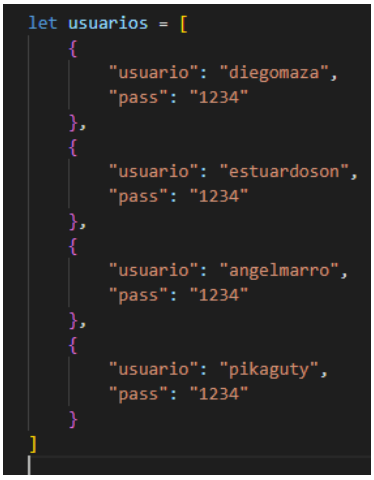


```
import { Request, Response } from 'express';

let parqueo: number[] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

/*
    disponible = 0
    ocupado = 1
    reservado = 2
*/

let propietarios = [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
//si esta activa o no
let activacionAntiRobo = [false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, fals
let activacionReserva = [false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false
// si esta sonando o no
let alarmaAntiRobo = false;
let alarmaReserva = [false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, fa
let tiempo = 300000
let pos = -1
```

Inicialización de los arreglos correspondientes a las posiciones y estados de los parqueos, siendo

- Disponible = 0
- Ocupado = 1
- Reservado = 2

Luego inicialización de estados y propietarios, así como el tiempo de reserva en milisegundos



```
let usuarios = [
    {
        "usuario": "diegomaza",
        "pass": "1234"
    },
    {
        "usuario": "estuardoson",
        "pass": "1234"
    },
    {
        "usuario": "angelmarro",
        "pass": "1234"
    },
    {
        "usuario": "pikaguty",
        "pass": "1234"
    }
]
```

Creación de usuarios

Clase IndexController: contiene lo que se realiza en cada petición y devuelve una respuesta en forma de json

```
public prueba(req: Request, res: Response) {

    res.json({ "funciona": "la api" });
}
```

Dirección de prueba

```
/*
    retorna el arreglo del parqueo, no recibe nada
*/
public verParqueo(req: Request, res: Response) {
    res.json({ "parqueo": parqueo })
}

/*
    setter de parqueos
    recibe
    {"parqueos":[]}
*/
public setParqueo(req: Request, res: Response) {
    parqueo = req.body.parqueos
    res.json({ "mensaje": "OK" })
}
```

```
/*
   retonrna el estado del parqueo en nummeros, no recibe nada
   {
      "disponibles": number,
      "ocupados": number,
      "reservados": number
   }
*/
public estadoParqueo(req: Request, res: Response) {
    let disponible = 0
    let ocupado = 0
    let reservado = 0
    for (let i = 0; i < 32; i++) {
        if (parqueo[i] == 0) {
            disponible += 1
        }
        else if (parqueo[i] == 1) {
            ocupado += 1
        }
        else {
            reservado += 1
        }
    }
    res.json({ "disponibles": disponible, "ocupados": ocupado, "reservados": reservado })
}
```

```
/*
    verifica si existen espacios disponibles en el arreglo, no recibe nada
    {"existencia": boolean}
*/
public hayExistencias(req: Request, res: Response) {
    res.json({ "existencia": parqueo.includes(0) })
}

/*
    reserva el parqueo solicitado por el tiempo establecido por el admin
    {
        "index":number,   -> debe ser el index del parqueo, un numero de 0 a 15
        "propietario":string o id
    }
*/
public reservarParqueo(req: Request, res: Response) {
    let i = req.body.index
    let p = req.body.propietario
    if (parqueo[i] == 0) {
        let bandera = true
        for (let j = 0; j < 32; j++) {
            if (parqueo[j] == 2 && propietarios[j] == p) {
                bandera = false
            }
        }
        if (bandera) {
            pos = i
            parqueo[i] = 2
            propietarios[i] = p
            activacionReserva[i] = true
            res.json({ "res": "OK" })
            let retardo = setTimeout((po=i) => {
                if (parqueo[po] != 1) {
                    console.log("po "+po+" t "+tiempo)
                    parqueo[po] = 0;
                    propietarios[po] = -1;
                    activacionReserva[po] = false
                }
            }, tiempo);
        } else {
            res.json({ "res": "Ya cuenta con una reserva" })
        }

    } else {
        res.json({ "res": "Parqueo no disponible para reserva" })
    }
}
```

```
    desactiva la alarma
    recibe
    {
        "posicion":int,
        "propietario":string o id
    }
*/
public DeshabilitarAlarmaReserva(req: Request, res: Response) {
    let posicion = req.body.posicion
    let propietario = req.body.propietario

    if (propietario == propietarios[posicion] && activacionReserva[posicion]) {
        activacionReserva[posicion] = false
        alarmaReserva[posicion] = false
        propietarios[posicion] = -1
        parqueo[posicion] = 0
        res.json({ "mensaje": "Alarma de reservacion ajustada con exito" })
    } else {
        res.json({ "mensaje": "Solo el propietario puede ajustar su alarma de reservacion" })
    }
```

```
public SonidoAlarmaReserva(req: Request, res: Response) {
    let posicion = req.body.posicion
    let propietario = req.body.propietario

    if (propietario == propietarios[posicion] && alarmaReserva[posicion]) {
        alarmaReserva[posicion] = false
        res.json({ "mensaje": "Alarma de reservacion apagada" })
    } else {
        res.json({ "mensaje": "Solo el propietario puede apagar su alarma de reservacion" })
    }

}
```

```
public HabilitarAlarmaAntiRobo(req: Request, res: Response) {
    let posicion = req.body.posicion
    let propietario = req.body.propietario

    if (-1 == propietarios[posicion] && !activacionAntiRobo[posicion]) {
        activacionAntiRobo[posicion] = true
        propietarios[posicion] = propietario
        res.json({ "mensaje": "Alarma de antirobo ajustada con exito" })
    } else {
        res.json({ "mensaje": "El lugar ya tiene registrado un propietario" })
    }
}
```

```
public DeshabilitarAlarmaAntiRobo(req: Request, res: Response) {
    let posicion = req.body.posicion
    let propietario = req.body.propietario

    if (propietario == propietarios[posicion] && activacionAntiRobo[posicion]) {
        activacionAntiRobo[posicion] = false
        alarmaAntiRobo = false
        propietarios[posicion] = -1
        res.json({ "mensaje": "Alarma de antirobo ajustada con exito" })
    } else {
        res.json({ "mensaje": "Solo el propietario puede ajustar su alarma de robo" })
    }
}
```

```
    recibe
    {
        "usuario":string,
        "pass":string
    }
    retorna
    {
        "correcto":boolean,
        "admin":boolean
    }
    */
public login(req: Request, res: Response) {
    let u = req.body.usuario
    let p = req.body.pass
    let flag = false;
    let admin = false
    for (let i = 0; i < usuarios.length; i++) {
        if (usuarios[i].usuario == u && usuarios[i].pass == p) {
            flag = true
            admin = true
            break;
        }
    }
    res.json({ "correcto": flag, "admin": admin })
}


/*
    ajusta el tiempo de reservacion
    recibe
    {"tiempo":int}
*/
public ajusteTiempo(req: Request, res: Response) {
    tiempo = req.body.tiempo
    res.json({"mensaje":"OK"})
}

/* retorna arreglo actual alarma anti robo */
public getAlarmaAntiRobo(req: Request, res: Response) {
    res.json({ "alarma": alarmaAntiRobo })
}

/*retorna arreglo arreglo antiRobo*/
public getAlarmaReserva(req: Request, res: Response) {
    res.json({ "alarmaReserva": alarmaReserva })
}
```

```
/*retorna arreglo arreglo antiRobo*/
public getAlarmaReserva(req: Request, res: Response) {
    res.json({ "alarmaReserva": alarmaReserva })
}

/*get activacionAntiRobo*/
public getActivacionAntiRobo(req: Request, res: Response) {
    res.json({ "activacionAntiRobo": activacionAntiRobo })
}

// get activacionReserva
public getActivacionReserva(req: Request, res: Response) {
    res.json({ "activacionReserva": activacionReserva })
}
```

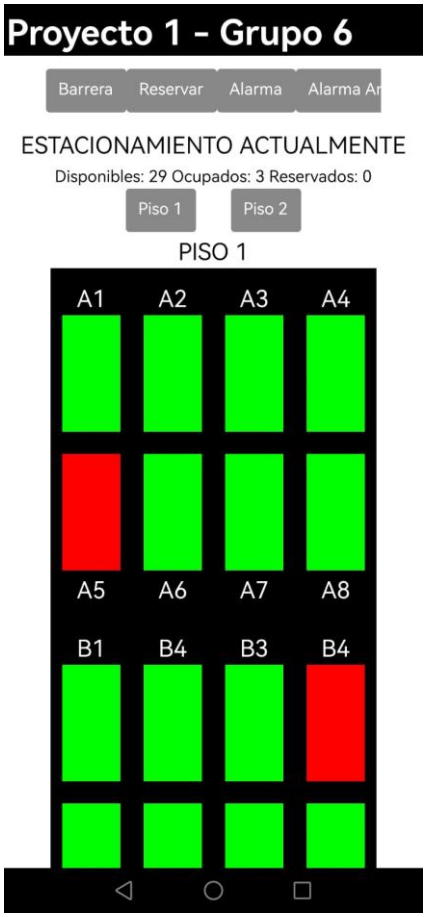
Aplicación Android:

Se creó una aplicación para dispositivos Android la cual nos permite realizar varias acciones como lo son la cantidad y estado de los parqueos, reserva un espacio, además de módulos para poder realizar distintas acciones. Entre estos módulos se encuentran de administración, barrera y reservación, alarma y alarma antirrobo.

Para la programación de la aplicación se utilizó MIT app inventor, el cual es un servicio de programación por bloques.

Icono de la app

Vista general



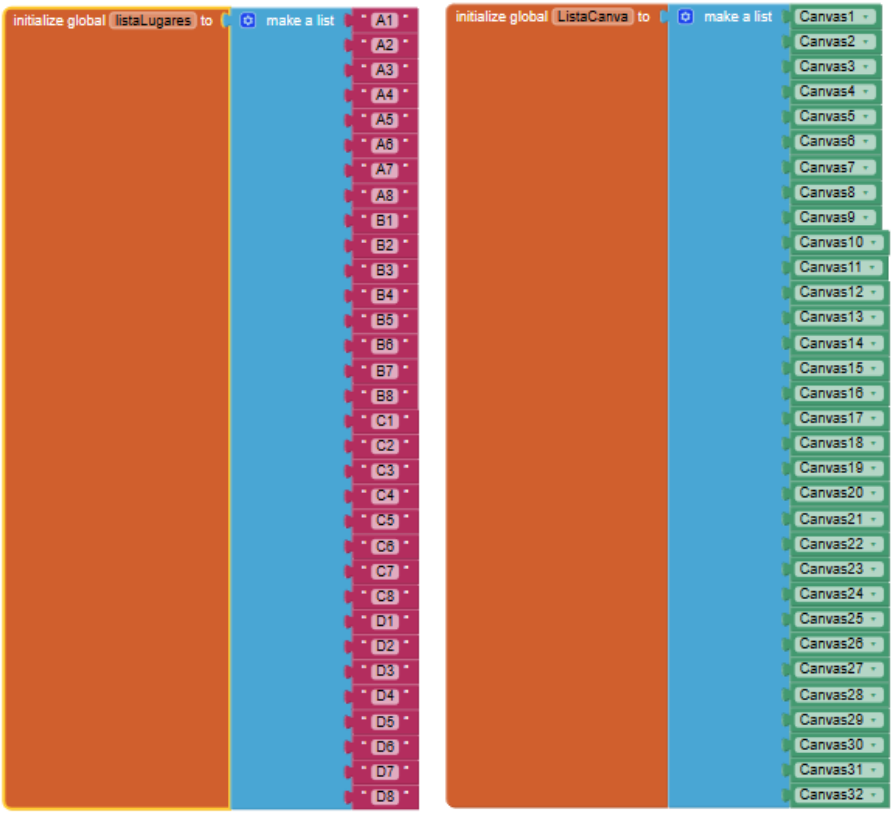
Para la programación de la aplicación se utilizó MIT app inventor, el cual es un servicio de programación por bloques.

Pantalla de Inicio

En la pantalla principal se encuentra un menú

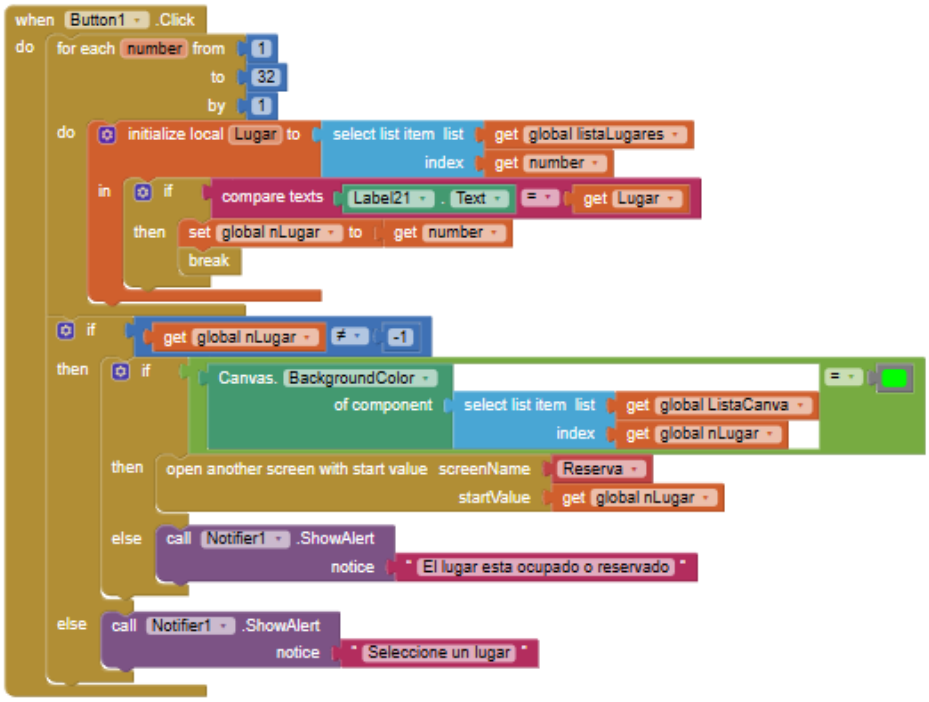
En el caso de los canvas que representan los lugares de los parqueos se realizó un bloque el cual indica que cuando uno es tocado haga visible el label21 en el cual mostrara el lugar escogido.

Listas globales que contienen los canvas y los nombres de las posiciones del estacionamiento.

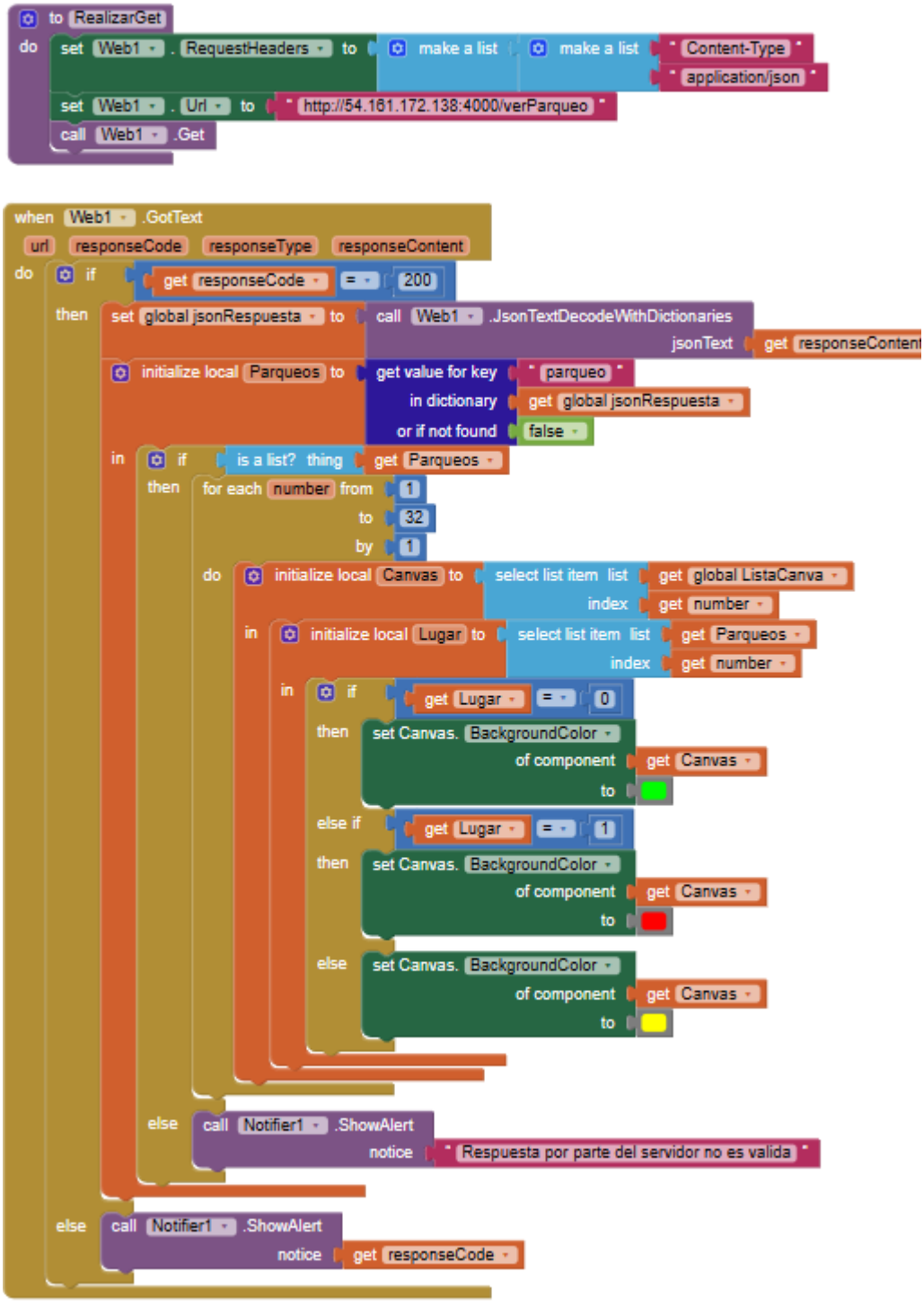


Variables globales que almacenan las respuestas de las peticiones al api rest y el número de la posición seleccionada por el usuario.

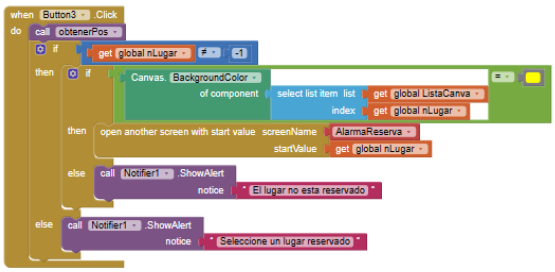
Botón con el que se mueve al apartado de reservar un lugar si el parqueo este desocupado.

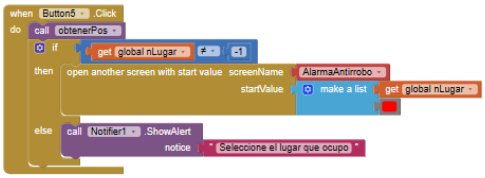


Procedimiento de solicitud a la api para obtener el estado de los parqueos y pintar los canvas del color correspondiente.

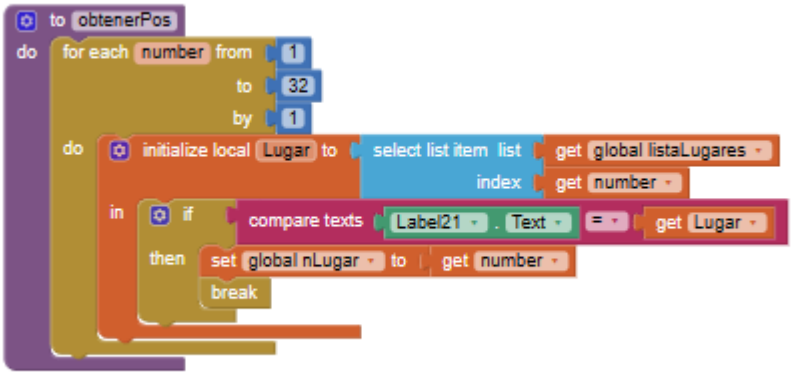


Botones que desplazan a las ventanas de reservar y alarma de reserva respectivamente.

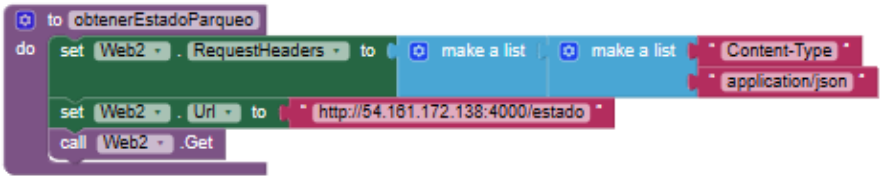


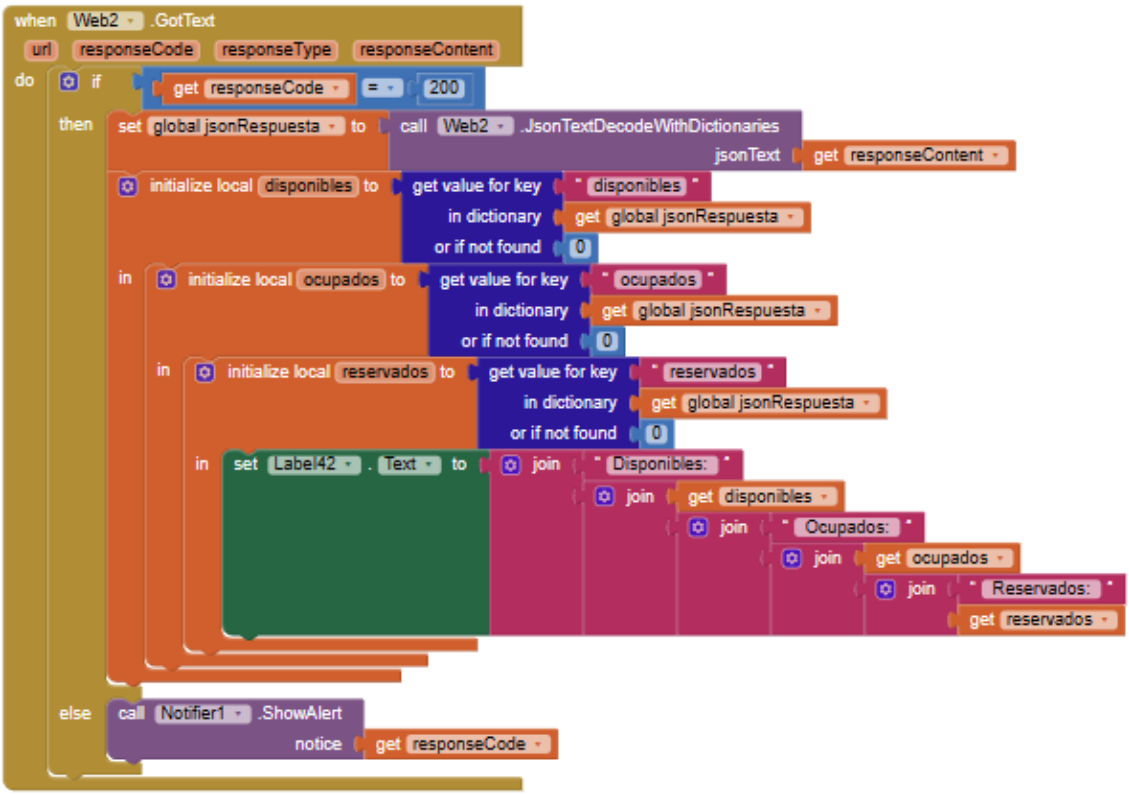


Procedimiento para obtener el index del parqueo seleccionado en la lista de 32 parqueos.

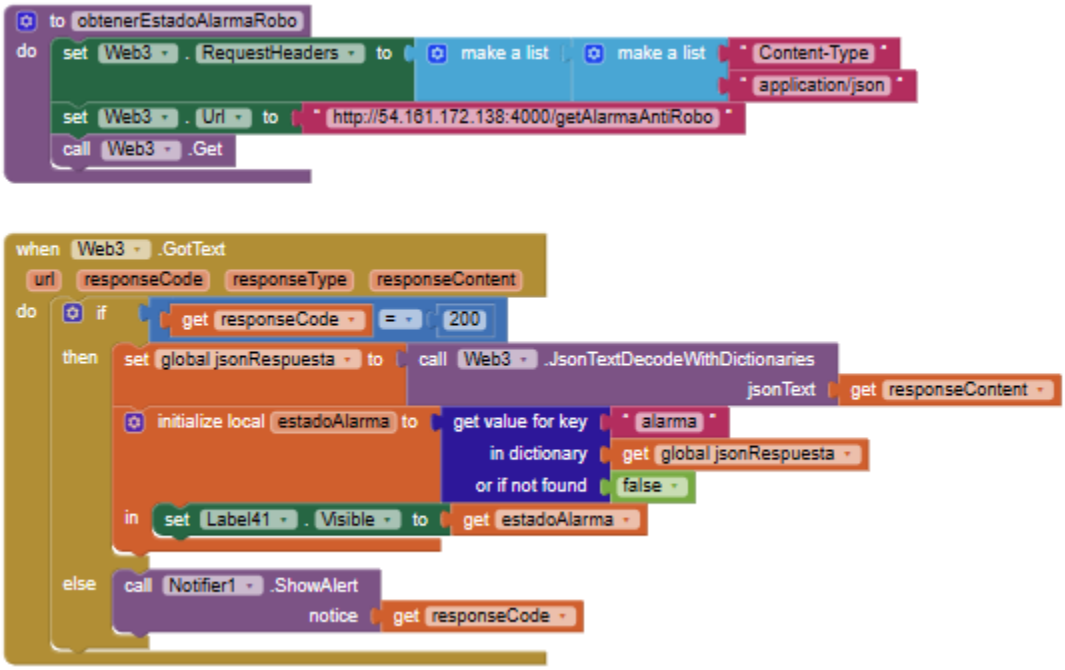


Solicitud al api para obtener el número de parqueos ocupados, disponibles y reservados y mostrarlos en el lable42.

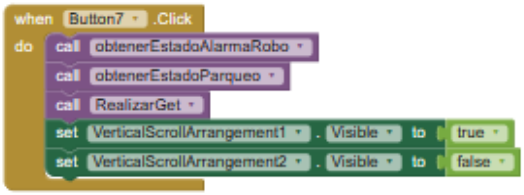

to obtenerEstadoParqueo
do
set Web2 . RequestHeaders to make a list make a list "Content-Type"
"application/json"
set Web2 . Url to "http://54.161.172.138:4000/estado"
call Web2 .Get

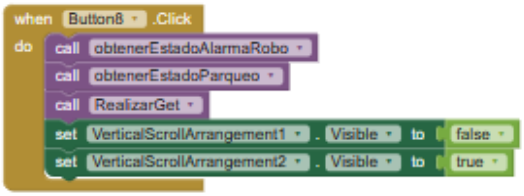

when Web2 .GotText
url responseCode responseType responseContent
do if get responseCode = 200
then set global jsonRespuesta to call Web2 .JsonTextDecodeWithDictionaries
jsonText get responseContent
initialize local disponibles to get value for key "disponibles"
in dictionary get global jsonRespuesta
or if not found 0
in initialize local ocupados to get value for key "ocupados"
in dictionary get global jsonRespuesta
or if not found 0
in initialize local reservados to get value for key "reservados"
in dictionary get global jsonRespuesta
or if not found 0
in set Label42 . Text to join "Disponibles:"
join get disponibles
join "Ocupados:"
join get ocupados
join "Reservados:"
get reservados
else call Notifier1 .ShowAlert
notice get responseCode

Solicitud al api para obtener si el estado de la alarma de robo está encendido o no.



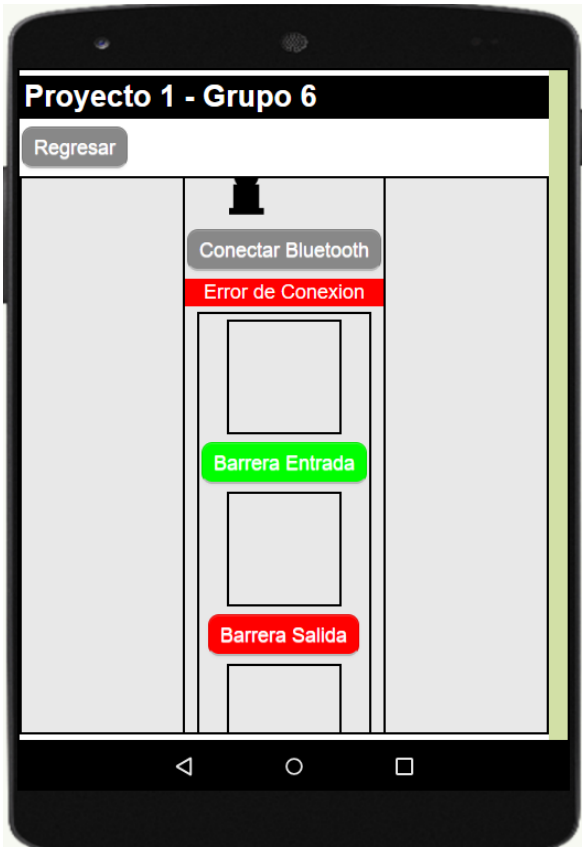
Botones para cambiar entre las vistas del piso 1 y piso 2 del estacionamiento.



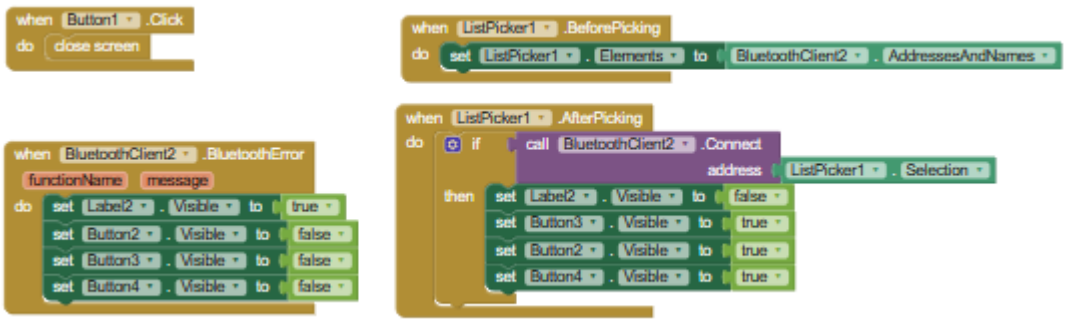


Pantalla de Barreras

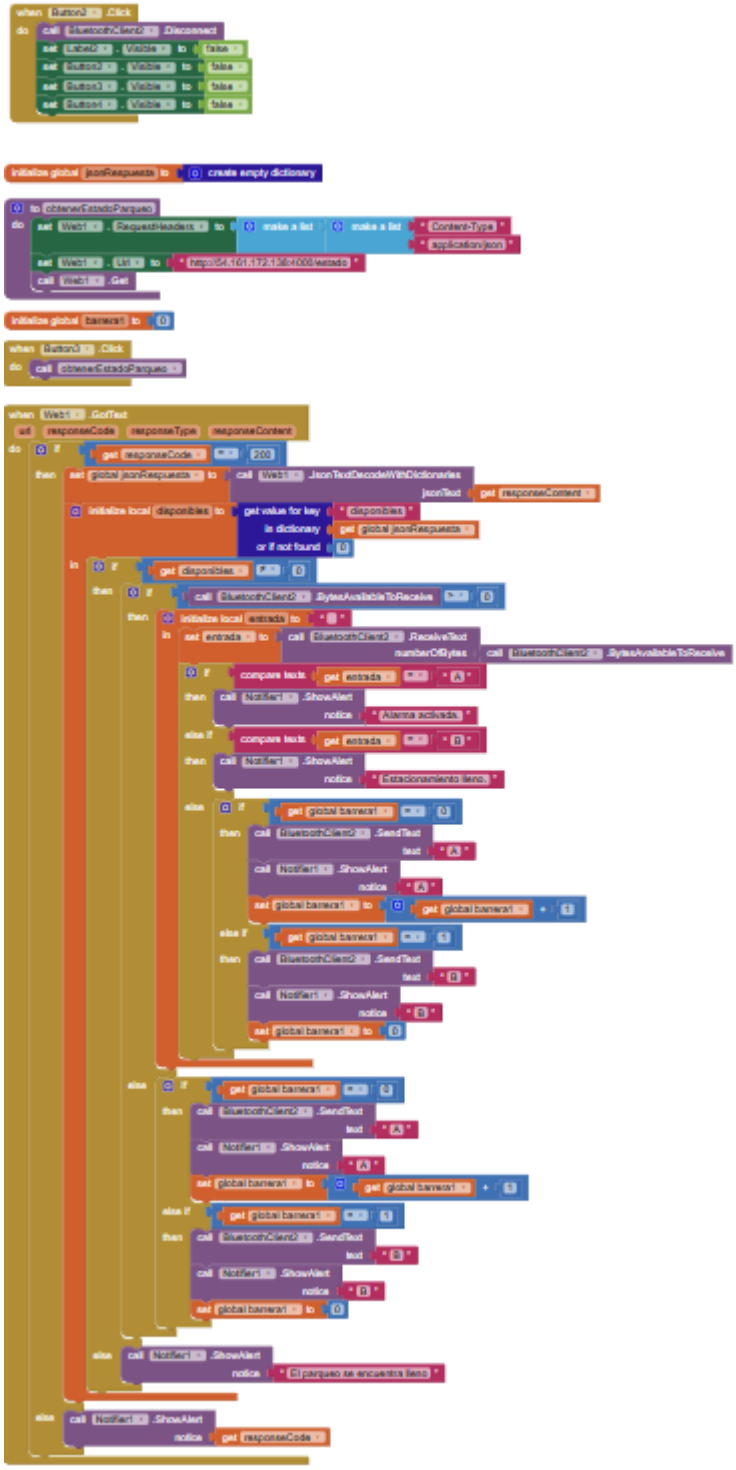
En esta pantalla por medio de una conexión bluetooth el usuario podrá conectarse a las barreras de acceso y salida del estacionamiento.



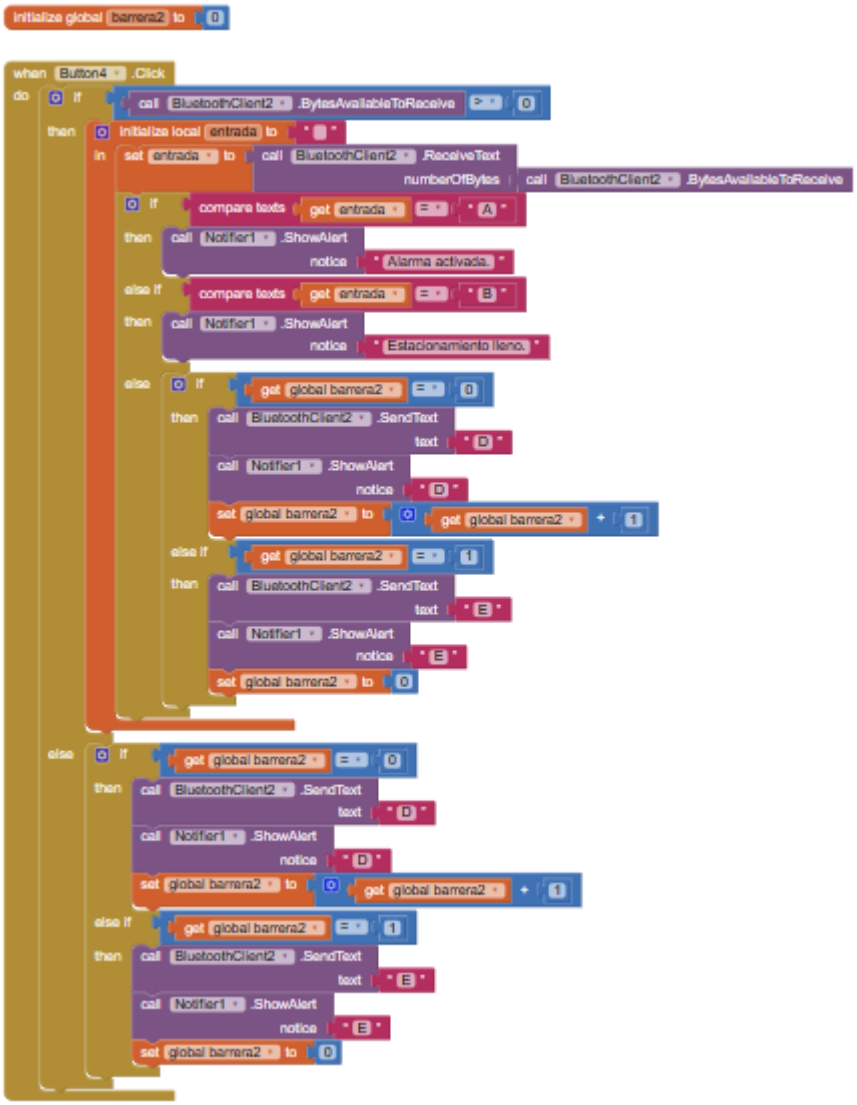
Las funciones para regresar a la pantalla de inicio y conexión bluetooth se encuentran en los siguientes bloques.



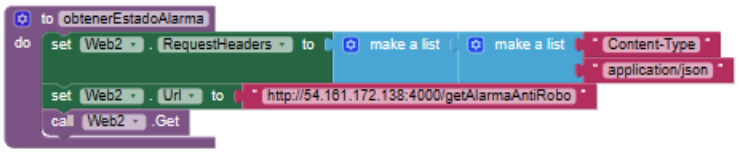
Para abrir la barrera de acceso al parqueo se envía una solicitud GET para saber si el parqueo tiene disponible algún lugar para estacionarse. En caso contrario la barrera no podrá abrirse.

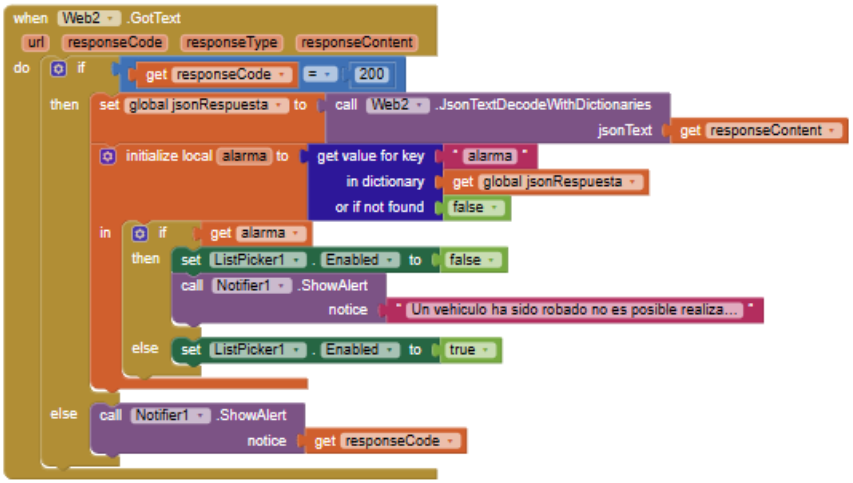


La barrera de salida no cuenta con ninguna restricción para su apertura por lo que podrá accionarse siempre y cuando se pueda realizar la conexión bluetooth.


initialize global barrera2 to 0
when Button4 .Click
do if call BluetoothClient2 .BytesAvailableToReceive > 0
then initialize local entrada to ""
in set entrada to call BluetoothClient2 .ReceiveText
numberOfBytes call BluetoothClient2 .BytesAvailableToReceive
if compare texts get entrada = "A"
then call Notifier1 .ShowAlert
notice "Alarma activada."
else if compare texts get entrada = "B"
then call Notifier1 .ShowAlert
notice "Estacionamiento lleno."
else if get global barrera2 = 0
then call BluetoothClient2 .SendText
text "D"
call Notifier1 .ShowAlert
notice "D"
set global barrera2 to get global barrera2 + 1
else if get global barrera2 = 1
then call BluetoothClient2 .SendText
text "E"
call Notifier1 .ShowAlert
notice "E"
set global barrera2 to 0
else if get global barrera2 = 0
then call BluetoothClient2 .SendText
text "D"
call Notifier1 .ShowAlert
notice "D"
set global barrera2 to get global barrera2 + 1
else if get global barrera2 = 1
then call BluetoothClient2 .SendText
text "E"
call Notifier1 .ShowAlert
notice "E"
set global barrera2 to 0

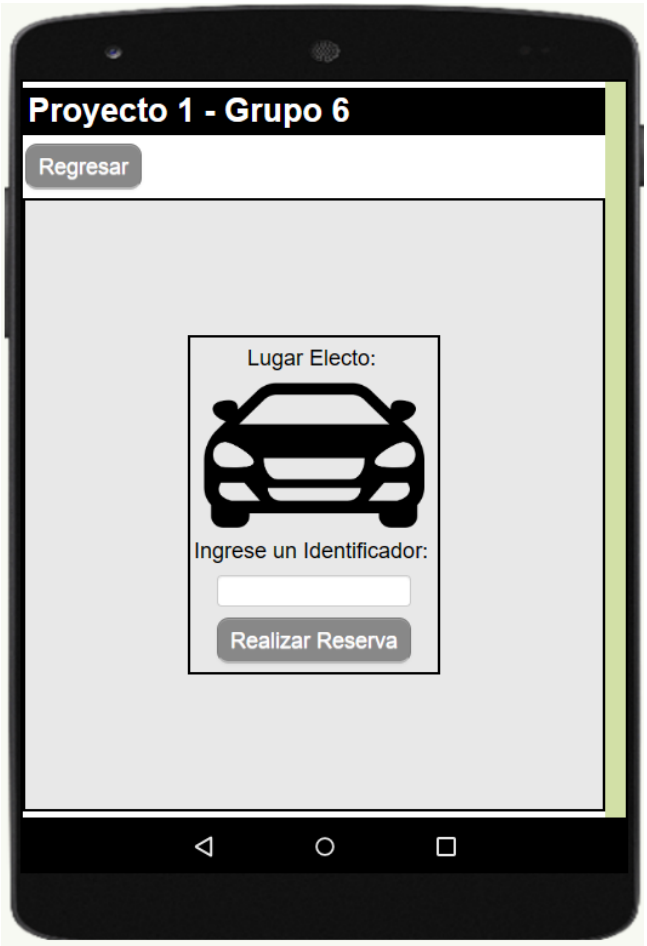
Sin embargo, la conexión bluetooth cuenta con la restricción dictaminada por una solicitud GET realizada al api rest la cual devuelve el valor de la alarma antirrobos, en caso este activada entonces no se puede entablar una conexión bluetooth. El bloque de código de esta solicitud se encuentra a continuación.



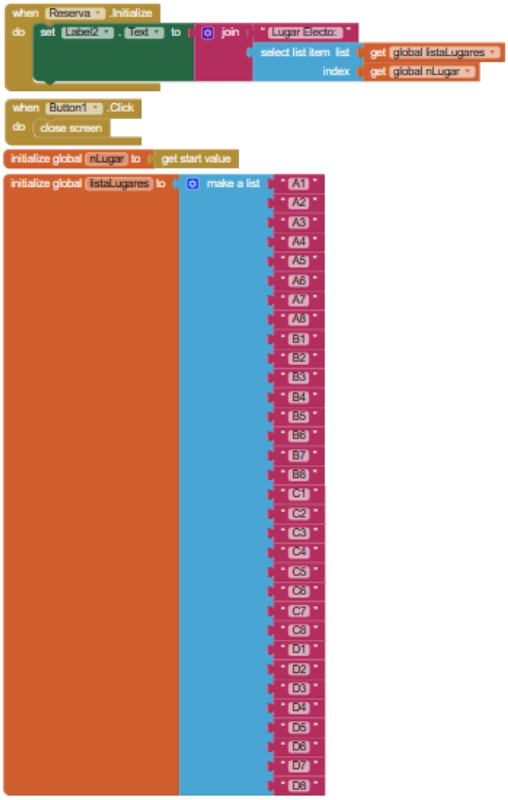


Pantalla de Reservaciones

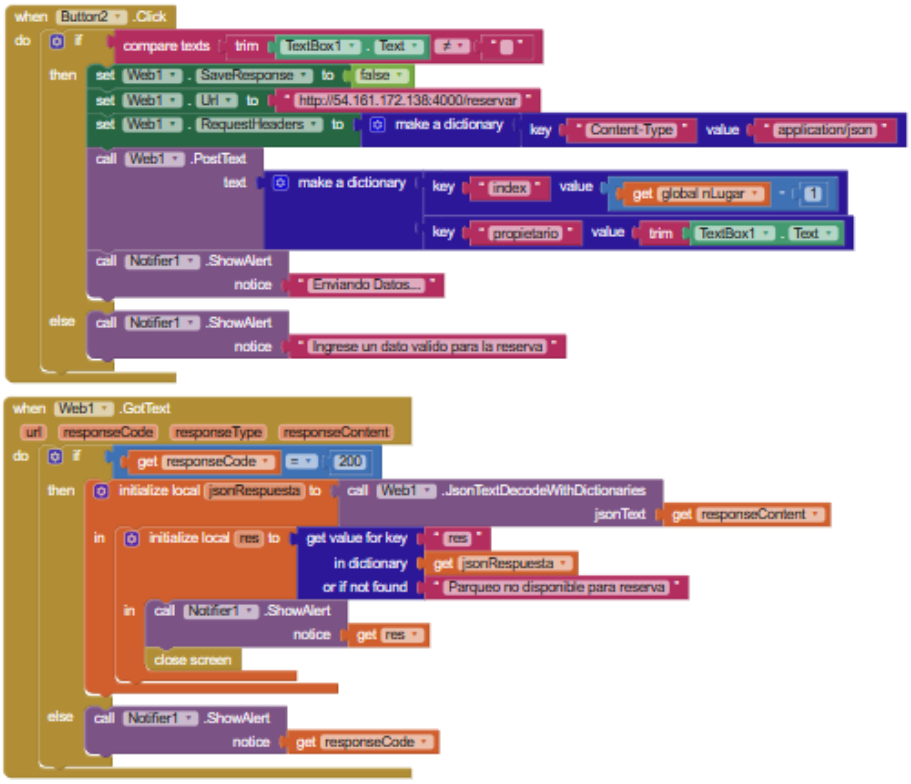
En esta pantalla el usuario podrá reservar espacios de estacionamientos siempre y cuando el espacio que reserve se encuentre disponible.



Las variables globales y acciones iniciales para abrir dicha pantalla se encuentran en los bloques que se presentan a continuación.

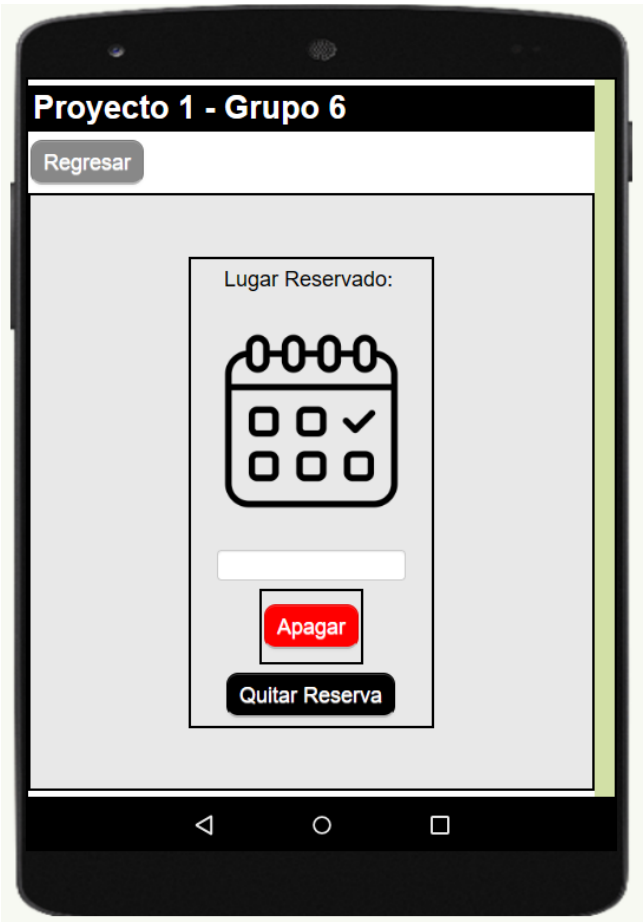


Para enviar los datos de la reserva al presionar el botón de envío se hará una solicitud al API con la que se guardaran.

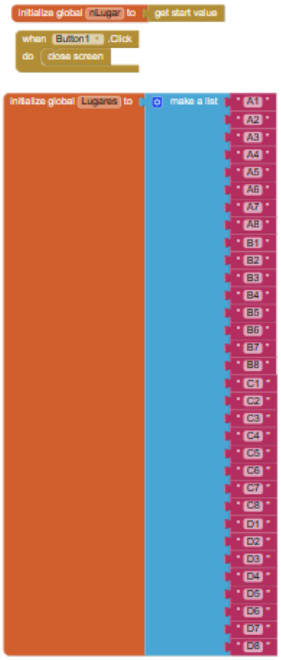


Pantalla de Alarma de Reserva

En esta pantalla el usuario será capaz de apagar y deshabilitar la alarma de una reservación así mismo podrá cancelar su reservación.



Las variables globales para el funcionamiento de esta pantalla se muestran en los siguientes bloques.

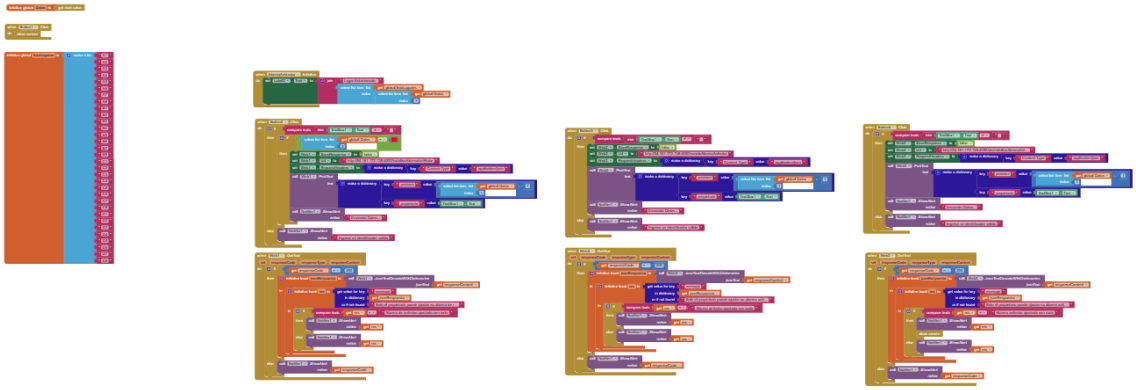


Para el funcionamiento de cada botón se realizan diferentes solicitudes POST para cambiar el estado de la alarma que cuyo propietario sea el indicado.

Pantalla de Alarma Antirrobo

En esta pantalla el usuario puede registrar su lugar de estacionamiento para activar la alarma antirrobo, así como silenciarla o desactivarla.

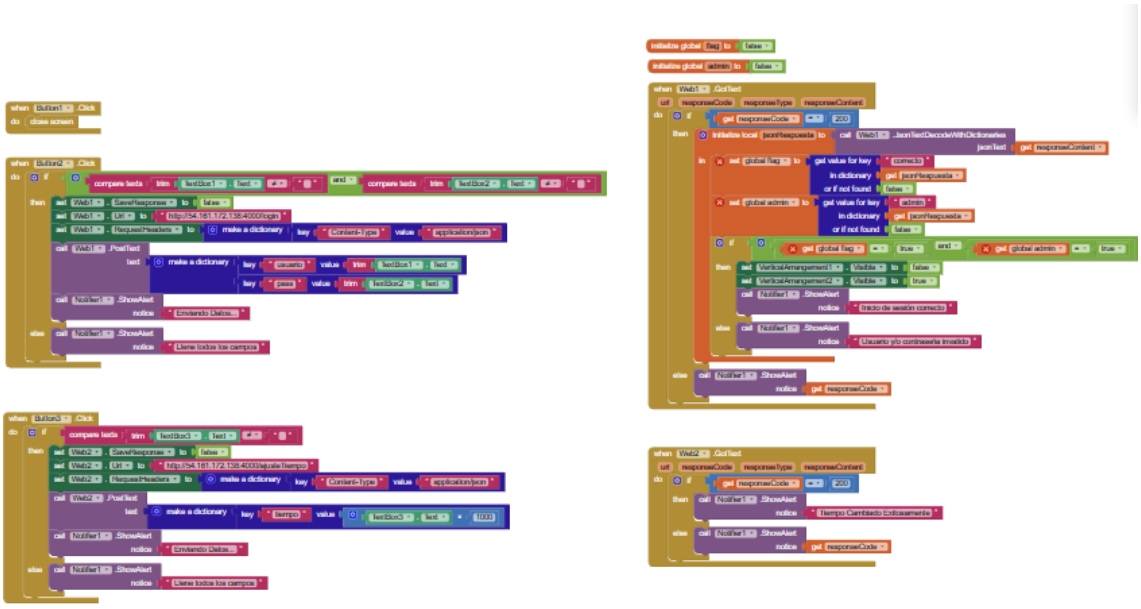
Los bloques para el funcionamiento de esta pantalla son similares a los de la pantalla anterior únicamente cambiando la ruta a la que se realiza la solicitud.



Pantalla Administrador

En esta pantalla el administrador puede ingresar para cambiar el tiempo de duración de las próximas reservas que se realicen.

Los bloques para el funcionamiento de esta pantalla se muestran a continuación los cuales realizan solicitudes POST para confirmar la existencia del usuario y posteriormente cambiar el tiempo de las reservaciones.



Código Arduino:

Creación de variables globales y arreglos para el manejo de parqueos y los posibles estados

```
int bloque1[] = {22,23, 24,25, 26,27, 28,29, 30,31, 32,33, 34,35, 36,37};

int luzb1[] = {13,11,9,7,5,3,14,16,20,A12,A0,A2,A4,A6,A8,A10};
int luzb2[] = {12,10,8,6,4,2,15,17,21,A13,A1,A3,A5,A7,A9,A11};

int disponibilidad[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

int alarmaReserval[] = {false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false};

int buzzer = A14;
int alarmaAnti = A15;
bool shutel[] = {false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false};

String inputString = "";
bool valido = false;

//Disponible - 0
//Ocupado - 1
//Reservado - 2
```

Variables locales usadas para la pantalla LCD, así como la creación de los caracteres personalizados que se utilizaron.

```
// ---------------------------------PARA LCD-------------------------
// libreria para el manejo de LCD
#include <LiquidCrystal.h>

// asignamos los pines de salida del arduino
const int rs1 = 38, en1 = 39, d41 = 40, d51 = 41, d61 = 42, d71 = 43;

// inicializamos el objeto LCD
LiquidCrystal lcd(rs1, en1, d41, d51, d61, d71);

// Creamos los array que utiliza la funicion createChar()
// Creamos los caracteres especiales para el carro
byte carchar4[8]= {B00000,B00000,B01110,B11111,B11011,B11111,B01110,B00000};
byte carchar5[8]={B11111,B11111,B00011,B00011,B00011,B11111,B11111,B00000};
byte carchar7[8]={B11111,B11111,B00000,B00000,B11111,B11111,B11111,B00000};
byte carchar8[8]={B11111,B11111,B00000,B00000,B11111,B11111,B11111,B00000};
byte carchar9[8]={B11000,B11110,B00110,B00011,B01111,B01111,B01111,B01111};

// Creamos las flechas
byte arrowchar[8]={B00100, B01110, B11111, B01110, B01110,B01110,B00000,B00000};
byte arrowchar2[8]={B00100, B00110, B11111, B11111, B00110,B00100,B00000,B00000};
// Creamos la barra invertida
byte negchar[8]={ B00000,B10000,B11000,B01100,B00110,B00011,B00001,B00000};



unsigned long  TiempoAnterior;

// para los delay en intervalos usando millis
#define INTERVALO_MENSAJE1 5000
#define INTERVALO_MENSAJE2 7000
#define INTERVALO_MENSAJE3 11000
#define INTERVALO_MENSAJE4 15000
unsigned long tiempo_1 = 0;
unsigned long tiempo_2 = 0;
unsigned long tiempo_3 = 0;
unsigned long tiempo_4 = 0;


// -------------------------------------FIN PARA LCD ------------------------------
```

Setup: inicialización de 2 puertos serial con una velocidad de 9600 baudios por segundo, de los pines para los estacionamientos, los pines para la pantalla lcd, los char personalizados y la variable de tiempo con la función milis();

```
void setup() {
  Serial.begin(9600);//iniciailzamos la comunicación
  Serial1.begin(9600);//iniciailzamos la comunicación

  for (int i = 0; i<16; i++){
    pinMode(bloquel[i],INPUT);
  }

  for (int i = 0; i<16; i++){
    pinMode(luzbl[i],OUTPUT);
    pinMode(luzb2[i],OUTPUT);
  }

  pinMode(buzzer,OUTPUT);
  pinMode(alarmaAnti,OUTPUT);


   //-----------------------------------------SETUP LCD-----------------
  // indicamos el tamaño del display
    // inicializamos la pantalla indicando filas y columnas
  lcd.begin(16,2);
  // asignamos un numero a la funcion createChar seguido de el nombre de el array creado

  // Este es un nuevo caracter
  lcd.createChar(6,negchar);

  // Creamos las flechas
  lcd.createChar(2,arrowchar);
  lcd.createChar(3,arrowchar2);

 //Creamos las partes del carro
  lcd.createChar(4,carchar4);
  lcd.createChar(5,carchar5);
  lcd.createChar(7,carchar7);
  lcd.createChar(8,carchar8);
  lcd.createChar(9,carchar9);


  TiempoAnterior = millis();

  //--------------------------------FIN SETUP LCD---------------------------------

}
```

Loop():

```
// ----- PARA EL LCD -----
int Parqueos_Disponibles = 0;
int Parqueos_Reservados = 0;
int Parqueos_Ocupados = 0;
```

Variables para el LCD

```
  //********** Nivel 1*************
  for (int i = 0; i < 16; i++){
    if (disponibilidad[i] != 2 && shutel[i] == false){
      //|| shute == false
      if (digitalRead(bloquel[i]) == HIGH){
        //Rojo
        digitalWrite(luzbl[i], HIGH);
        digitalWrite(luzb2[i], LOW);
        disponibilidad[i] = 1;
        Parqueos_Ocupados ++;

      }else{
        //Verde
        digitalWrite(luzbl[i], LOW);
        digitalWrite(luzb2[i], HIGH);
        disponibilidad[i] = 0;
        Parqueos_Disponibles++;
      }
    }else{
      //Amarillo
      if (digitalRead(bloquel[i]) == HIGH){
        disponibilidad[i] = 1;
        digitalWrite(luzbl[i], HIGH);
        digitalWrite(luzb2[i], LOW);
        shutel[i] = true;
        digitalWrite(buzzer, HIGH);
      }else{
        disponibilidad[i] = 2;
        digitalWrite(luzbl[i], HIGH);
        digitalWrite(luzb2[i], HIGH);
        shutel[i] = false;
        digitalWrite(buzzer, LOW);
      }
      Parqueos_Reservados++;
```

Revisión del estado de los parqueos para luego prender el color led correspondiente

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//Alarma antirrobos
//Si está desocupado (disponible = 0) y la alarma sigue activa (alarma=true) se eniende la alarma
for (int i = 0; i < 16; i++){
  if (disponibilidad[i] == 0  && alarmaReserval[i] == true){
    digitalWrite(alarmaAnti, HIGH);
    break;
  }else{
    digitalWrite(alarmaAnti, LOW);
  }
}

String reporte = "";
for (int i = 0; i<16; i++){
  reporte.concat(disponibilidad[i]);
  reporte.concat(",");
}
Serial1.print(reporte);
```

Revisión de parqueos para determinar si hubo un robo y activar la alarma, además del reporte de parqueos para imprimir el reporte

```
  //Aqui se lee la respuesta de Python

  String ret = Serial1.readString();

  String strs[20];
  int StringCount = 0;

  while (ret.length() > 0){
    int index = ret.indexOf(',');
    if (index == -1){
      strs[StringCount++] = ret;
      break;
    }else{
      strs[StringCount++] = ret.substring(0, index);
      ret = ret.substring(index+1);
    }
  }

  for (int i = 0; i < 16; i++){
     disponibilidad[i] = strs[i].toInt();
  }

  delay(500);
```

Obtención de los datos obtenidos mediante la respuesta de Python

```
//--------------------------------LCD LOOP----

if(Parqueos_Disponibles==0){
    if(millis()-TiempoAnterior >= 1000){
      TiempoAnterior = millis();
  // limpiamos la pantalla LCD
  lcd.clear();
lcd.setCursor(6,1);
lcd.write("[");
lcd.setCursor(7,1);
lcd.write("/");
lcd.setCursor(8,1);
lcd.write("_");
lcd.setCursor(9,1);
lcd.write(6);
lcd.setCursor(10,1);
lcd.write("]");
lcd.setCursor(7,0);
lcd.write(6);
lcd.setCursor(8,0);
lcd.write(8);
lcd.setCursor(9,0);
lcd.write("/");
lcd.setCursor(13,1);
lcd.write("X");
lcd.setCursor(13,0);
lcd.write("X");
lcd.setCursor(3,1);
lcd.write("X");
lcd.setCursor(3,0);
lcd.write("X");
}
```

```
if(millis()-TiempoAnterior >= 1000){
  TiempoAnterior = millis();
  lcd.clear();
lcd.setCursor(5,0);
lcd.write("PARQUEO");
lcd.setCursor(6,1);
lcd.write("LLENO");
 lcd.setCursor(13,1);
lcd.write("X");
lcd.setCursor(13,0);
lcd.write("X");
lcd.setCursor(3,1);
lcd.write("X");
lcd.setCursor(3,0);
lcd.write("X");
```

Mensaje a mostrar con su respectiva animación en la pantalla lcd si el parqueo se encuentra lleno.

```
}else{

    // son if para no utilizar delay ya que el delay pausa todo el codigo
    // por lo tanto se utilizaron los millies
    if(millis() > tiempo_1 + INTERVALO_MENSAJE1){
      tiempo_1 = millis();
    lcd.setCursor(0,0);
    lcd.print("PARQUEOS ");
    lcd.setCursor(0,1);
    lcd.print("DISPONIBLES: ");
    lcd.print(Parqueos_Disponibles);


  }

  if(millis() > tiempo_2 + INTERVALO_MENSAJE2){
      tiempo_2 = millis();
      lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("PARQUEOS ");
    lcd.setCursor(0,1);
    lcd.print("0CUPADOS: ");
    lcd.print(Parqueos_Ocupados);


  }

  if(millis() > tiempo_3 + INTERVALO_MENSAJE3){
      tiempo_3 = millis();
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("PARQUEOS ");
      lcd.setCursor(0,1);
      lcd.print("RESERVADOS: ");
      lcd.print(Parqueos_Reservados);
```

```
}
if(millis() > tiempo_4 + INTERVALO_MENSAJE4){
    tiempo_4 = millis();
    // Este ciclo for es para el desplazamiento a la derecha
  for(int i= 0; i<24;i++){
  // idicamos la posicion de el caracter en el display + i
  // Como i va aumentando los caracteres se empiezan a desplazar
  lcd.setCursor(1+i,1);
  // Escribimos sobre la LCD
  lcd.write("[");
  lcd.setCursor(2+i,1);
  lcd.write(4);
  lcd.setCursor(3+i,1);
  lcd.write("_");
  lcd.setCursor(4+i,1);
  lcd.write(4);
  lcd.setCursor(5+i,1);
  lcd.write("]");
  lcd.setCursor(2+i,0);
  lcd.write("/");
  lcd.setCursor(3+i,0);
  lcd.write(8);
  lcd.setCursor(4+i,0);
  lcd.write(9);
  lcd.setCursor(7+i,1);
  lcd.write(3);
  lcd.setCursor(7+i,0);
  lcd.write(3);
  // Estas son las flechas
  lcd.setCursor(15,0);
  lcd.write(2);
  lcd.setCursor(15,1);
  lcd.write(2);
```

```
      lcd.setCursor(-9+i,0);
      lcd.write("AVANCE");

      lcd.setCursor(i,1);
      lcd.write(3);
      lcd.setCursor(i,0);
      lcd.write(3);
      delay(100);

      // leugo se borra

      lcd.clear();


  }// fin del for
    }// fin del delay

  }// fin de else
```

Mensaje a mostrar con el estado de los parqueos si estos aun disponen de algún espacio

Arduino 2 (Auxiliar)

Se necesito utilizar un segundo Arduino debido a la cantidad de pines a usar, además este tiene el código referente al motor stepper usado para la simulación de la barrera con recepción de bluetooth

Inclusión de librería Stepper así como declaración de variables necesarias para el control del servo

```
#include <Stepper.h>

// Declaramos la variable para controlar el servo
const int spr = 100;
Stepper barreraEntrada(spr, 40, 41, 42, 43);
Stepper barreraSalida(spr, 44,45,46,47);
//Variables para bluetooth
char entradaBluetooth;
int ledBarreraEntrada = 48;
int ledBarreraSalida = 49;
int pinesBarreras[] = {40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53};

String inputString = "";
bool valido = false;
```

Setup()

inicialización de los puertos serial con una velocidad de 9600 baudios por segundo, así como la inicialización de pines

```
void setup() {
  Serial.begin(9600);//iniciailzamos la comunicación
  Serial1.begin(9600);
  Serial2.begin(9600);
  inputString.reserve(200);

  for(int i=0; i<13; i++){
      pinMode(pinesBarreras[i], OUTPUT);
  }

  barreraEntrada.setSpeed(10);
  barreraSalida.setSpeed(10);
}

void serialEvent(){
  while (Serial.available()){
    char inChar = (char)Serial.read();
    if (inChar != '_'){
      inputString += inChar;
    }else{
      valido = true;
    }

  }
}
```

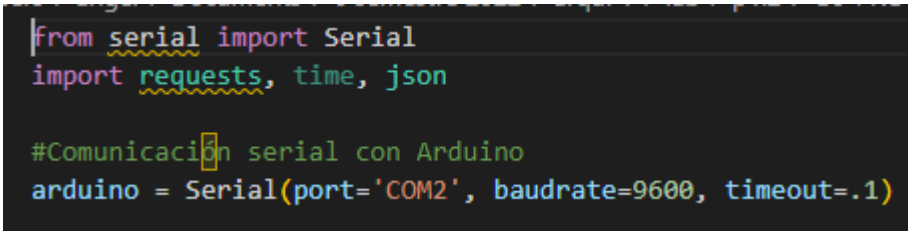
Loop():

Control del motor para controlar la barrera por medio de la señal recibida por bluetooth, dependiendo de la solicitud se realiza un movimiento diferente.

```
void loop(){
  if(Serial2.available() > 0){
    entradaBluetooth = Serial2.read();

    if(entradaBluetooth == 'A'){
      digitalWrite(ledBarreraEntrada, HIGH);
      // Desplazamos a la posición 90°
      barreraEntrada.step(-66);
    }

    if(entradaBluetooth == 'B'){
      digitalWrite(ledBarreraEntrada, LOW);
      // Desplazamos a la posición 0°
      barreraEntrada.step(66);
    }

    if(entradaBluetooth == 'D'){
      digitalWrite(ledBarreraSalida, HIGH);
      // Desplazamos a la posición 90°
      barreraSalida.step(-66);
    }

    if(entradaBluetooth == 'E'){
      digitalWrite(ledBarreraSalida, LOW);
      // Desplazamos a la posición 0°
      barreraSalida.step(66);
    }
  }

}
```

Python: Usado para consumir la Api en el lado del Arduino y poder enviar la información necesaria para actualizar los datos.

Conexión por medio de un puerto serial



```
try:
    # Informacion que manda arduino
    data = arduino.readline().decode("utf-8")

    if data != "": # Para que no imprima basura
        li = list(data.split(","))
        c = 0
        for i in li:
            li[c] = int(i)
            c = c + 1
```

```
#print(li)
# Request a la API
# Se obtiene el contenido del parqueo para verificar si se apartó algo
x = requests.get("http://54.161.172.138:4000/verParqueo")
x = json.loads(x.text).get("parqueo")

#print("Recibo de la app")
#print(x)
c = 0
for i in x:
    if i == 2: #Verificamos si hay apartados del lado de la app
        li[c] = 2

    #Se da en el caso que el tiempo de la reserva se termine
    if li[c] == 2 and i == 0:
        li[c] = 0

    c = c + 1
```

```
            # Se manda el estado de los parqueos detectado por el circuito
            estado = {'parqueos': li}
            x = requests.post("http://54.161.172.138:4000/setParqueo", json = estado)
            if (json.loads(x.text).get("mensaje")) == "OK":
                print("Se envió")
                print(li)

            # Informacion que se manda al arduino
            cadena = ""
            c = 0
            for i in li:
                if c < 31:
                    cadena = cadena + str(li[c]) + ","
                else:
                    cadena = cadena + str(li[c])
                c += 1
            arduino.write(bytes(cadena, 'utf-8'))
    except:
        continue
```