# Anexo de códigos

En este anexo se adjuntaron los códigos más relevantes para esta tesina.

# Códigos del lado del servidor

# Código StandarFirmata utilizado en el Arduino MEGA

# Código ConfigurableFirmata utilizado en el Arduino NANO

# Código Servidor Node (server.js)

const express = require('express');

const bodyParser = require('body-parser');

const path = require('path');

const http = require('http');

const app = express();

// API file for interacting with MongoDB

const api = require('./server/routes/api');

// Parsers

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: false}));

// Angular DIST output folder

app.use(express.static(path.join(\_\_dirname, 'dist')));

// CORS

app.use(function(req, res, next) {

    res.header("Access-Control-Allow-Origin", "\*");

    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

    next();

});

// API location

app.use('/api', api.rutas);

app.use('/api', api.placas);

// Enviar todo lo otro a Angular

app.get('\*', (req, res) => {

    if (process.env.AMBIENTE == 'DESARROLLO'){

        //Estoy levantando angular en puerto 4200

     res.status(404).send('Estas en desarrollo');

    }else{

res.sendFile(path.join(\_\_dirname, 'dist/index.html'));

    }

});

//Set Port

const port = process.env.PORT || '3000';

app.set('port', port);

const server = http.createServer(app);

server.listen(port, () => console.log(`Running on localhost:${port}`));

# Código API Express (api.js)

const express = require('express');

const router = express.Router();

const MongoClient = require('mongodb').MongoClient;

const ObjectID = require('mongodb').ObjectID;

const MINIMODISTANCIA = 20;

/\*Esto es para apagar\*/

const control = require('./apagar');

console.log(control.saludar());

if (process.env.AMBIENTE == 'DESARROLLO') {

console.log('Iniciado desarrollo');

var hola = require('./hola');

module.exports = hola;

} else {

console.log('Iniciado Test');

var placas = require('./placas');

module.exports.placas = placas;

}

var cont = 0;

// Connect

const connection = (closure) => {

return MongoClient.connect('mongodb://localhost:27017/sar', (err, db) => {

if (err) return console.log(err);

closure(db);

});

};

// Error handling

const sendError = (err, res) => {

response.status = 501;

response.message = typeof err == 'object' ? err.message : err;

res.status(501).json(response);

};

// Response handling

let response = {

status: 200,

data: [],

message: null

};

router.get('/temperaturas', (req, res) => {

var lista = [];

connection((db) => {

db.collection('temperaturas')

.distinct("fecha").then((fechas) => {

fechas.forEach(f => {

db.collection("temperaturas")

.find({ "fecha": f }, { "valor": 1, "hora": 1, "\_id": 0 }).sort({ "fecha": 1, "hora": 1 }).batchSize(30000)

.toArray(function (err, result) {

if (err) throw err;

lista.push({ series: result, fecha: f });

if (lista.length === fechas.length) {

console.log('Mostrando ----\*\*\*\*');

console.log(lista[5].series.length);

console.log(lista[5].fecha);

res.json(lista);

db.close();

}

});

});

})

.catch((err) => {

sendError(err, res);

})

});

});

router.get('/monoxidos', (req, res) => {

connection((db) => {

db.collection('mq7')

.find()

.toArray()

.then((valores) => {

response = valores;

res.json(response);

db.close();

})

.catch((err) => {

sendError(err, res);

});

});

});

router.get('/monoxidosActual', (req, res) => {

var ahora = new Date();

var despues = new Date();

despues.setSeconds(ahora.getSeconds() + 5);

hora1 = ahora.getHours() + ':' + ahora.getMinutes() + ':' + ahora.getSeconds();

hora2 = despues.getHours() + ':' + despues.getMinutes() + ':' + despues.getSeconds();

// console.log('Buscando con '+hora1+' y '+hora2);

connection((db) => {

db.collection('monoxidos')

.findOne({ "hora": { $gte: hora1, $lte: hora2 } }, { "valor": 1, "hora": 1, "\_id": 0 },

function (err, result) {

if (err) throw err;

// console.log(result);

res.json(result);

db.close();

});

});

});

/\*

Se exportan las variables para que sean conocidas por Nodejs

\*/

router.get('/apagar', (req, res) => {

console.log('LLego solicitud de apagado...');

control.apagar();

});

router.get('/reiniciar', (req, res) => {

console.log('Llego solicitud de reinicio...');

control.reiniciar();

});

module.exports.rutas = router;

# Código Manejo de Arduino Mega y Arduino Nano (placas.js)

const express = require('express');

const router = express.Router();

const MongoClient = require('mongodb').MongoClient;

const ObjectID = require('mongodb').ObjectID;

const MINIMODISTANCIA = 20;

const connection = (closure) => {

return MongoClient.connect('mongodb://localhost:27017/sar', (err, db) => {

if (err) return console.log(err);

closure(db);

});

};

var ports = [{ id: "mega", port: "/dev/ttyACM0" },

{ id: "nano", port: "/dev/ttyUSB0" }

];

var five = require("johnny-five");

new five.Boards(ports).on("ready", function () {

//Se enciende la placa

var motor1;

var motor2;

var motor3;

var motor4;

// configuro el sensor de monoxido

//Se enciende la placa

var sensor = new five.Sensor({

pin: "A0",

board: this.byId("mega"),

freq: 1000 //Frecuencia en msg

});

//Configuro el sensor de temperatura

var thermometer = new five.Thermometer({

controller: "DS18B20",

pin: 2,

board: this.byId("nano")

});

var gps = new five.GPS({

pins: {

rx: 15,

tx: 14,

board: this.byId("mega")

}

});

// si latitud o longitud cambian

gps.on("ready", function () {

console.log("position");

console.log(" latitude : ", this.latitude);

console.log(" longitude : ", this.longitude);

console.log(" altitude : ", this.altitude);

connection((db) => {

db.collection('gps')

.insert({ "latitude": this.latitude, "longitude": this.longitude, "altitude": this.altitude, "fecha": new Date() })

.then((muestrasGPS) => {

console.log('Insertando muestra GPS');

})

.catch((err) => {

console.log('Error al insertar GPS');

});

});

});

// If speed, course change log it

gps.on("navigation", function () {

console.log("navigation");

console.log(" speed : ", this.speed);

console.log(" course : ", this.course);

connection((db) => {

db.collection('navegacion')

.insert({ "velocidad": this.speed, "curso": this.course, "fecha": new Date() })

.then((muestrasGPS) => {

console.log('Insertando muestra navegacion');

})

.catch((err) => {

console.log('Error al insertar GPS');

});

});

});

//Configuro el sensor de proximidad en el pin 22

var proximityAdelante = new five.Proximity({

controller: "HCSR04",

pin: 22,

board: this.byId("mega")

});

var proximityDerecho = new five.Proximity({

controller: "HCSR04",

pin: 24,

board: this.byId("mega")

});

var proximityIzquierdo = new five.Proximity({

controller: "HCSR04",

pin: 26,

board: this.byId("mega")

});

var distanciaAdelante = 0;

motor1 = new five.Motor({

pins: {

pwm: 10,

dir: 7,

cdir: 6,

board: this.byId("mega")

}

});

motor2 = new five.Motor({

pins: {

pwm: 11,

dir: 8,

cdir: 9,

board: this.byId("mega")

}

});

motor3 = new five.Motor({

pins: {

pwm: 13,

dir: 5,

cdir: 4,

board: this.byId("mega")

}

});

motor4 = new five.Motor({

pins: {

pwm: 12,

dir: 2,

cdir: 3,

board: this.byId("mega")

}

});

function stop() {

motor1.stop();

motor2.stop();

motor3.stop();

motor4.stop();

}

//Muestro la temperatura

thermometer.on("change", function () {

//console.log(this.celsius + "°C");

//Agregado para generar hora

var ahora = new Date();

var hora = ahora.getHours() + ':' + ahora.getMinutes() + ':' + ahora.getSeconds();

var fechaAlmacenar = ahora.getFullYear() + '/' + (ahora.getMonth() + 1) + '/' + ahora.getDate();

connection((db) => {

db.collection('temperaturas')

.insert({ "valor": this.celsius, "fecha": fechaAlmacenar, "hora": hora, "fechaSys": ahora })

.then((temperaturas) => {

console.log('Insertando temperatura:' + this.celsius);

db.close();

})

.catch((err) => {

console.log('Error al insertar');

});

});

});

sensor.on("change", function (value) {

connection((db) => {

var ahora = new Date();

hora = ahora.getHours() + ':' + ahora.getMinutes() + ':' + ahora.getSeconds();

db.collection('monoxidos')

.insert({ "valor": sensor.scaleTo([20, 2000]), "fecha": new Date(), "hora": hora })

.then((sensorMQ7) => {

console.log('Insertando MQ7');

console.log(sensor.scaleTo([20, 2000]) + 'ppm'); // float

db.close();

})

.catch((err) => {

console.log('Error al insertar valores de mq7');

});

});

});

// Si se generan modificaciones en la distancia de objetos, paran o avanzan los motores

proximityAdelante.on("change", function () {

if (proximityAdelante.cm <= MINIMODISTANCIA) {

stop();

}

});

router.get('/arriba', (req, res) => {

console.log('Accionando arriba');

if (proximityAdelante.cm > MINIMODISTANCIA) {

motor1.forward(255);

motor2.forward(255);

motor3.forward(255);

motor4.forward(255);

res.json("ok");

} else {

res.json("{'error':'objeto adelante'}");

}

});

router.get('/izquierda', (req, res) => {

console.log('Accionando izquierda');

res.json("ok");

motor1.forward(255);

motor2.reverse(255);

motor3.forward(255);

motor4.reverse(255);

});

router.get('/derecha', (req, res) => {

console.log('Accionando derecha');

motor1.reverse(255);

motor2.forward(255);

motor3.reverse(255);

motor4.forward(255);

res.json("ok");

});

router.get('/abajo', (req, res) => {

console.log('Accionando abajo');

motor1.reverse(255);

motor2.reverse(255);

motor3.reverse(255);

motor4.reverse(255);

res.json("ok");

});

router.get('/stop', (req, res) => {

console.log('deteniendo');

motor1.stop();

motor2.stop();

motor3.stop();

motor4.stop();

res.json("ok");

});

router.get('/ultrasonido', (req, res) => {

res.json([{ ultrasonidoAdelante: proximityAdelante.cm },

{ ultrasonidoDerecho: proximityDerecho.cm },

{ ultrasonidoIzquierdo: proximityIzquierdo.cm }

]);

});

router.get('/gps', (req, res) => {

res.json({

latitud: gps.latitude,

longitud: gps.longitude,

altitud: gps.altitude,

velocidad: gps.speed,

sat: gps.sat,

curso: gps.course,

tiempo: gps.time

});

});

router.get('/temperatura', (req, res) => {

res.json({

temperatura: thermometer.celsius, unidad: "celsius"

});

});

router.get('/monoxido', (req, res) => {

res.json({ monoxido: sensor.scaleTo([20, 2000]), unidad: "ppm" });

});

});

module.exports = router;

# Código de Apagar y reiniciar (apagar.js)

// Require child\_process

var exec = require('child\_process').exec;

// Create shutdown function

var os = require('os');

var apagar = function shutdown() {

if (os.type()[0] === 'W' || os.type()[0] === 'w') {

console.log('Es windows');

} else {

console.log('Es linux');

}

console.log(os.type());

exec('shutdown -h now', function (error, stdout, stderr) { console.log(stdout); });

}

var reiniciar = function reiniciar() {

exec('shutdown -r now', function (error, stdout, stderr) { console.log(stdout); });

}

var saludar = function saludar(callback) {

console.log('Hola desde apagar.js');

}

module.exports.reiniciar = reiniciar;

module.exports.apagar = apagar;

module.exports.saludar = saludar;

# Códigos del lado del cliente

# Código de Servicio Angular (servicio.ts)

import { Gps } from './Gps';

import { Injectable } from '@angular/core';

import { Http, Headers, RequestOptions } from '@angular/http';

import {HttpClient} from '@angular/common/http';

import 'rxjs/add/operator/map';

import { Observable } from 'rxjs/Observable';

import { AppComponent } from './app.component';

import { Temperatura } from './Temperatura';

import { Sensores } from './Sensores';

import { GraficaTemperatura } from './graficaTemperatura';

import { Monoxido } from './Monoxido';

@Injectable()

export class ServicioAplicacion {

public sensores: Sensores;

constructor(public http: Http, public http2: HttpClient, public app: AppComponent) {

this.sensores = new Sensores();

}

/\*\*

\* Evento que sirve para enviar desde el componente

\* @param url Url a enviar el evento [arriba, abajo, izquierda, derecha]

\*/

enviarEvento(accion: String): Observable <any> {

return this.http.get(this.app.rutaBasica + accion);

}

solicitarTemperaturaActual(): Observable <Temperatura> {

return this.http.get(this.app.rutaBasica + 'temperatura')

.map(res => res.json());

}

/\*\*

\* Devuelve ([{ultrasonidoAdelante:proximityAdelante.cm},

    {ultrasonidoDerecho: proximityDerecho.cm},

    {ultrasonidoIzquierdo: proximityIzquierdo.cm}

    ])

\*/

solicitarUltrasonidosActual(): Observable <any> {

return this.http.get(this.app.rutaBasica + 'ultrasonido')

.map(res => res.json());

}

/\*\*devuelve

\* {monoxido: sensor.scaleTo([20,2000]), unidad: "ppm"}

\*/

solicitarMonoxidoActual(): Observable <any> {

return this.http.get(this.app.rutaBasica + 'monoxido')

.map(res => res.json());

}

solicitarGpsActual(): Observable <Gps> {

return this.http.get(this.app.rutaBasica + 'gps')

.map(res => res.json());

}

pedirImagen(): Observable<boolean> {

return this.http.get(this.app.rutaWeb)

.map(res => res.json());

}

actualizarValores(muestraSensores: Sensores) {

this.sensores = muestraSensores;

}

apagar() {

return this.http.get(this.app.rutaBasica + 'apagar');

}

reiniciar() {

return this.http.get(this.app.rutaBasica + 'reiniciar');

}

solicitarTodasTemperaturas(): Observable <GraficaTemperatura[]> {

return this.http2.get<GraficaTemperatura[]>(this.app.rutaBasica + 'temperaturas');

}

solicitarMonoxidoActualBD(): Observable <Monoxido> {

return this.http2.get<Monoxido>(this.app.rutaBasica + 'monoxidosActual');

}

}

# Código de appComponent.ts Angular

import { Sensores } from './Sensores';

import { Component } from '@angular/core';

import { TemperaturaService } from './temperatura.service';

import { Temperatura } from './Temperatura';

import { DatePipe } from '@angular/common';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css'],

providers: [TemperaturaService],

})

export class AppComponent {

titulo = 'Sistema Autónomo Robótico';

public temperaturas: Temperatura[];

public sensoresActuales: Sensores;

public tiempoDelay;

public mensaje = '';

public rutaBasica = 'http://192.168.2.1:3000/api/';

// public rutaBasica = 'http://localhost:3000/api/';

// public rutaBasica = 'http://192.168.1.39:3000/api/';

public rutaWeb = 'http://192.168.2.1:8081';

constructor(public service: TemperaturaService) {

this.sensoresActuales = new Sensores();

this.tiempoDelay = 1000;

}

}

# Código de appModule (rutas) [Extracto]

const routes: Routes = [

{ path: '', component: PizarraComponent },

{ path: 'estadisticas', component: EstadisticasComponent },

{ path: 'control', component: PanelInferiorComponent }

];

@NgModule({

declarations: [

AppComponent,

CapturaVideoComponent,

TablaSensoresComponent,

TablaInfoComponent,

PanelInferiorComponent,

PanelControlComponent,

PizarraComponent,

EstadisticasComponent,

EncabezadoComponent,

],

imports: [

BrowserModule,

HttpModule,

RouterModule.forRoot(routes),

NgxChartsModule,

BrowserAnimationsModule,

FormsModule,

HttpClientModule

],

providers: [TemperaturaService],

bootstrap: [AppComponent]

})

export class AppModule { }