OMNIPRESENCE

a Remote Reality Platform

DESCRIPCIÓN DEL PROYECTO:

Resumen:

“El coronavirus ha reconfigurado drásticamente, la manera en la que los seres humanos nos relacionamos con el mundo. El distanciamiento social ha modificado nuestras relaciones interpersonales y nuestra relación con el espacio.”

OMNIPRESENCE es un proyecto inmersivo de Realidad Remota(RR), que explora las posibilidades de la robótica y la telepresencia, con el objetivo de desafiar las leyes espacio-temporales, para acceder y recorrer diferentes espacios físicos de manera remota e instantánea.

OMNIPRESENCE es un proyecto de investigación sobre las posibilidades que nos dan las innovaciones propias de la industria 4.0, para el servicio de la creación, el arte y la cultura.
La Industria 4.0 o también conocida como “la cuarta revolución industrial”, está marcada por la aparición de nuevas tecnologías como la robótica, la analítica, la inteligencia artificial, las tecnologías cognitivas, la nanotecnología, el Internet of Things (IoT), VR y AR entre otros.

La introducción de estas tecnologías permiten la hibridación del mundo físico (dispositivos, máquinas e instalaciones) con el mundo digital (sistemas). Esta conexión permite que dispositivos y sistemas colaboren entre sí, y con otros sistemas, para crear nuevos sistemas intercomunicados e inteligentes.

Definimos a la Realidad Remota(RR) como una nueva vertiente de la Realidad Expandida (Realidad Virtual-VR, Realidad Aumentada-AR y Realidad Mixta-MR).
.
La Realidad Remota crea una copia virtual del mundo físico en tiempo real y te permite acceder a ella. Es un ejercicio de exploración de la realidad de manera remota, que propone la creación de objetos robóticos móviles, integrados con cámaras estereoscópicas y sensores, que transmiten información por la red.

OMNIPRESENCE podría definirse también como un proyecto de XR o Cross Reality, un entorno de realidad mixta que proviene de la fusión redes de sensores y actuadores, y mundos virtuales compartidos en línea.

Objetivos:

OMNIPRESENCE es una plataforma de presencia remota o telepresencia, open source, que propone la creación de una flota de avatares robóticos móviles, dotados de cámaras estereoscópicas y con conexión a internet, capaces de moverse en el espacio físico y transmitir, en tiempo real, una visión de 360º del espacio donde se encuentran, a una plataforma donde los usuarios se pueden conectar, con gafas y controles de VR, para

monitorizar, controlar el movimiento de los avatares robóticos y visualizar, las imágenes estereoscópicas en tiempo real enviadas por dichos avatares.

El objetivo principal es crear un prototipo de robot móvil, de control remoto vía internet, de telepresencia de bajo costo basado en hardware y software de código abierto y fácilmente replicable.

Descripción:

Para entender mejor el proyecto, empezaremos por decir que está compuesto por 3 partes:

- Usuario: Desde una gafas de VR se conecta a la plataforma web.
- Plataforma Web: Para monitorización y control de avatares robóticos
- Avatar Robótico: Robot a control remoto dotado de una cámara que hace streaming y que envía la imagen estereoscópica en tiempo real a la plataforma

Usuario:
El usuario puede acceder a la plataforma y cumplir un doble rol.
Puede seguir las instrucciones para construir su avatar y darlo de alta en la plataforma.
Puede monitorizar y controlar los avatares desde la plataforma.
Desde una gafas de VR se conecta a la web, recibe imagen estereoscópica en tiempo real y con los controles, puede mover de posición el avatar en el espacio físico.

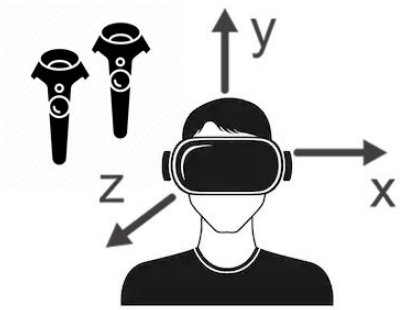
Plataforma Web: Para la creación, monitorización y control de avatares robóticos.

- Creación de avatares: La plataforma online, cuenta con un tutorial paso a paso, para la creación de un avatar robótico con motores DC para su desplazamiento, dotado de GPS y Cámara Estereoscópica móvil.
- Monitorización de avatares: La plataforma online, cuenta con una sección con un mapa del mundo y te permite visualizar los diversos puntos donde existe un avatar, si están activos o no, cuánta batería les queda.
- Control de avatares: Una vez seleccionado el avatar a controlar, la plataforma te lleva a una página de control, desde donde podrás visualizar la imagen estereoscópica obtenida en tiempo real y controlar tu avatar robótico.

Avatar Robótico: Dotado de una cámara que hace streaming capaz de enviar una imagen estereoscópica en tiempo real. La cámara se encuentra en un soporte giratorio que es controlado con el giroscopio de las gafas de VR. Todo esto va sobre un chasis metálico, con motores DC y cuyos movimientos son controlados por el usuario de manera remota a través de la plataforma.

OMNIPRESENCE

a Remote Reality Platform



Usuario

WebVR

Plataforma online

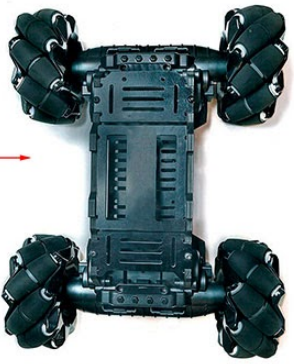
Wifi

Cámara estereoscopica

2iC

Raspberry Pi 4

GPIO



Chasis 4x Motor DC

GPIO

Servo Motores Rotación e inclinación de Camara

GPIO

Modulo GPS

Josecarlos FLórez

http://josecarlosflorez.com/

OMNIPRESENCE

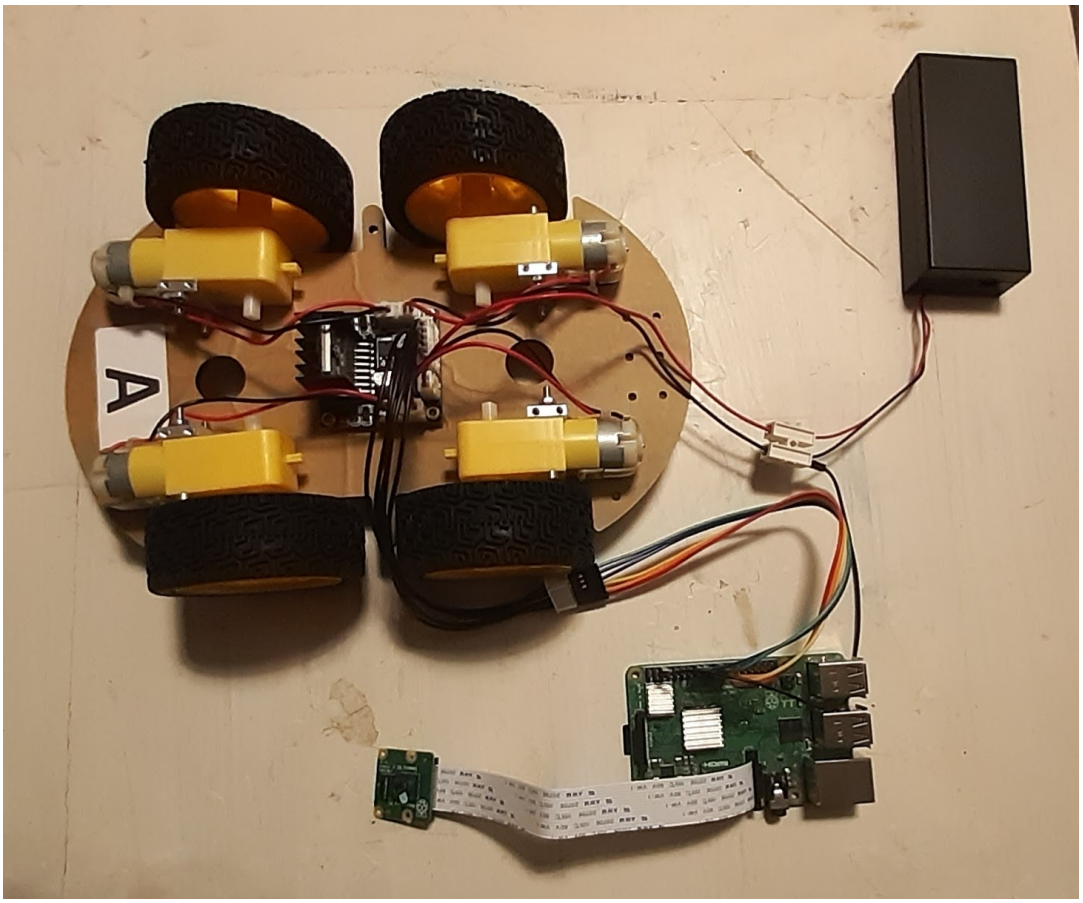
a Remote Reality Platform

Proceso de Construcción

Para este primer prototipo, crearemos un robot móvil, conectado a internet, que dotado con una cámara, pueda ser controlado desde una página web en cualquier parte del mundo.

El dispositivo móvil usaremos ha sido creado usando un chasis y motores DC controlados por una raspberry pi a través de los pines GPIO, usando el driver Driver L298N.

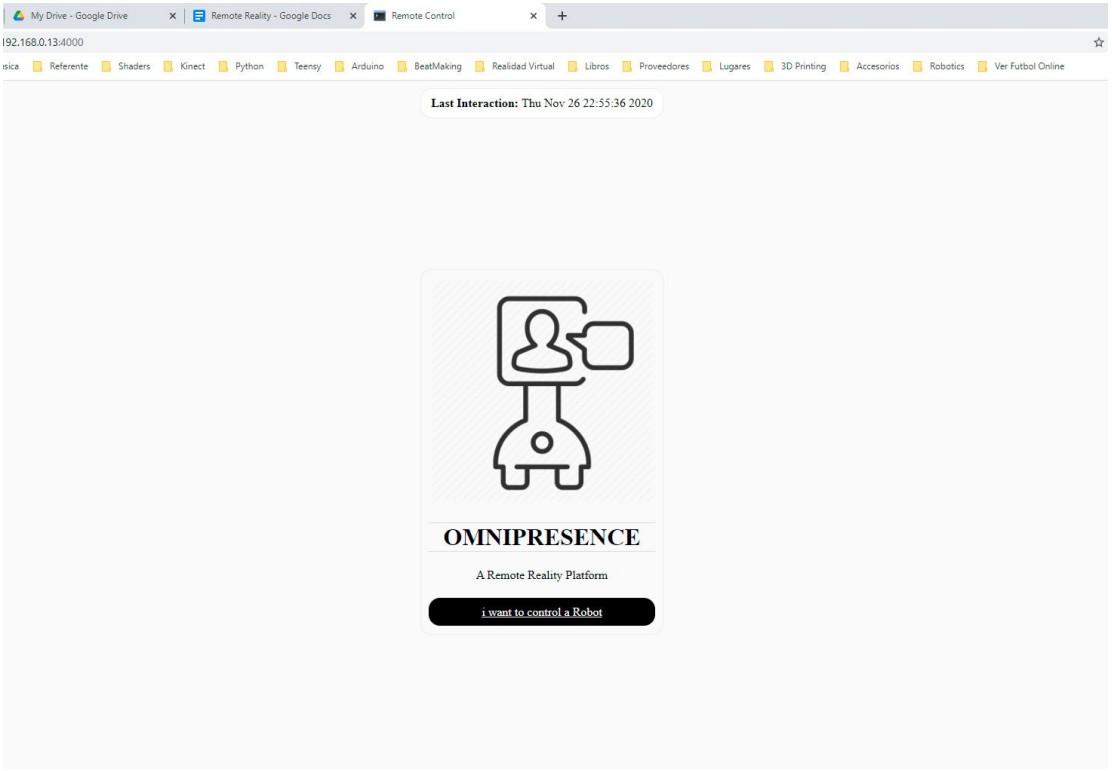
Para realizar el streaming de video en tiempo real, estamos usando la Raspberry Pi Camera Module V.2, junto con la librería PiCamera, que nos dará menor latencia que una cámara web USB ya que la cámara va por i2c.

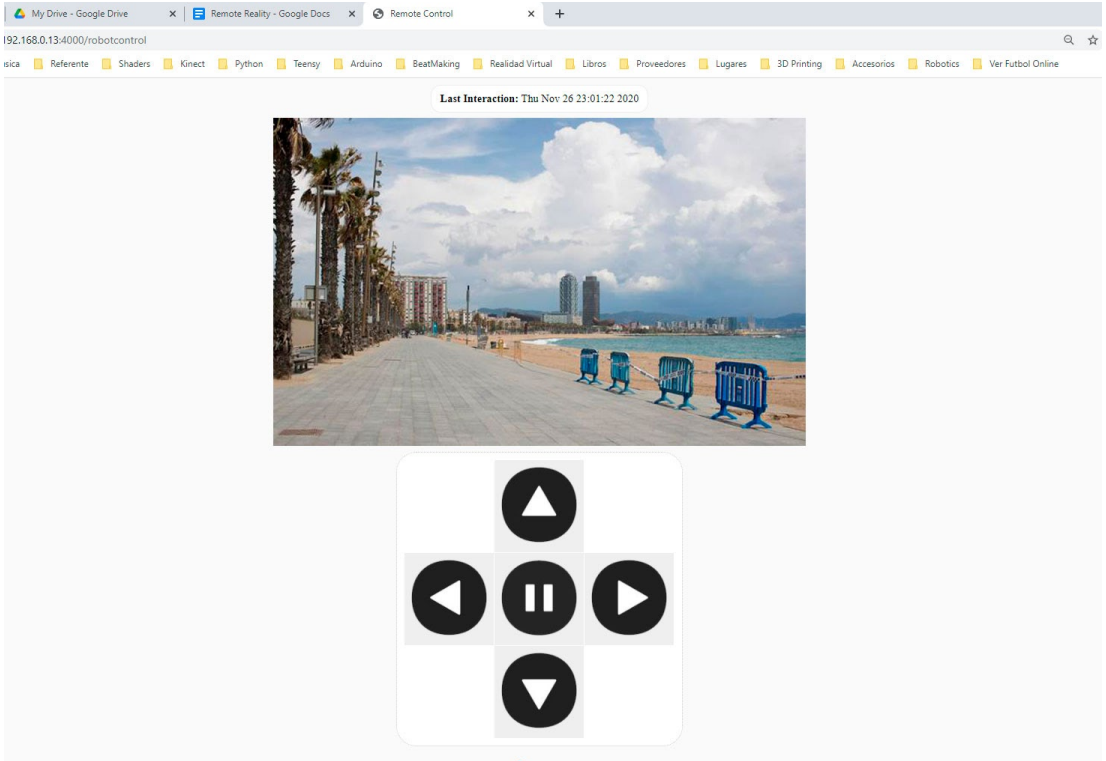


Crearemos una página web para visualizar el contenido que transmite la cámara y controlar el dispositivo en tiempo real. Para esto usaremos la librería Flask de Python, HTML y CSS

OMNIPRESENCE

a Remote Reality Platform





Josecarlos FLórez

http://josecarlosflorez.com/

OMNIPRESENCE

a Remote Reality Platform

Componentes:

Hardware:
1x Raspberry Pi 3
1x Driver L298N
4x Motores DC
1x Servo
1x Camera Raspberry Pi v.2

Software:
Raspberry Pi OS
Python
Flask
Html
Picamera library

Estructura Software:

-- *PROJECT*
- app-motors.py
- motors.py
- appCam.py
- camera_pi.py
- -- *TEMPLATES*
  - --index.html
  - --robotControl.html
- -- *STATIC*
  - style.css
  - --*IMG*

Josecarlos FLórez
http://josecarlosflorez.com/

OMNIPRESENCE

a Remote Reality Platform

app-motor.py

```
from flask import *
import requests
import motors
import RPi.GPIO as GPIO

from flask_googlemaps import GoogleMaps

import time

from time import sleep

from camera_pi import Camera

servo_pin = 7

GPIO.setmode(GPIO.BOARD)


#add for servo
GPIO.setup(servo_pin, GPIO.OUT)
p = GPIO.PWM(servo_pin, 50)
p.start(0)
#for servo



app = Flask(__name__)

app.config['GOOGLEMAPS_KEY'] = "AIzaSyAunTzUN1GHqD3yjhtZFZP8WT8kiusLHD0"
GoogleMaps(app)

@app.route('/')
def index():
        timeNow = time.asctime (time.localtime(time.time()))
        templateData = {
                'time': timeNow
        }
        return render_template('index.html', **templateData)

@app.route('/<changepin>', methods=['POST'])
def reroute(changepin):

        changePin = int(changepin)

        if changePin == 1:
                motors.turnLeft()
                motors.stop()
        elif changePin == 2:
                motors.forward()
                motors.stop()
        elif changePin == 3:
                motors.turnRight()
                motors.stop()
        elif changePin == 4:
                motors.backward()
                motors.stop()
        else:
                motors.stop()

        response = make_response(redirect(url_for('robotcontrol')))
        return (response)

@app.route('/')
def index():
        timeNow = time.asctime (time.localtime(time.time()))
        templateData = {
                'time': timeNow
        }
        return render_template('index.html', **templateData)

@app.route('/<changepin>', methods=['POST'])
def reroute(changepin):

        changePin = int(changepin)

        if changePin == 1:
                motors.turnLeft()
                motors.stop()
        elif changePin == 2:
                motors.forward()
                motors.stop()
        elif changePin == 3:
                motors.turnRight()
                motors.stop()
        elif changePin == 4:
                motors.backward()
                motors.stop()
        else:
                motors.stop()

        response = make_response(redirect(url_for('robotcontrol')))
        return (response)

@app.route('/test', methods=['POST'])
def test():
        slider1 = request.form["slider1"]
        p.ChangeDutyCycle(float(slider1))
        sleep(1)
        p.ChangeDutyCycle(0)
        return render_template('robotcontrol.html')

@app.route('/robotcontrol')
def robotcontrol():
        timeNow = time.asctime( time.localtime(time.time()))
        templateData = {
                'time':timeNow
        }
        return render_template('robotcontrol.html', **templateData)

def gen(camera):
        while True:
                frame = camera.get_frame()
                yield (b'--frame\r\n'b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/video_feed')
def video_feed():
        return Response(gen(Camera()), mimetype='multipart/x-mixed-replace;boundary=frame')

app.run(debug=True, host='192.168.0.13', port=4000)
```

Josecarlos FLórez

http://josecarlosflorez.com/

OMNIPRESENCE

a Remote Reality Platform

motors.py

```
from flask import Flask, render_template_string, request
import RPi.GPIO as GPIO
import requests
from time import sleep

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

Motor1A = 16
Motor1B = 18
Motor1Enable = 22

Motor2A = 21
Motor2B = 19
Motor2Enable = 23

#Set up all as outputs

GPIO.setup(Motor1A,GPIO.OUT)
GPIO.setup(Motor1B,GPIO.OUT)
GPIO.setup(Motor1Enable,GPIO.OUT)
GPIO.setup(Motor2A,GPIO.OUT)
GPIO.setup(Motor2B,GPIO.OUT)
GPIO.setup(Motor2Enable,GPIO.OUT)

def forward():
        print("Going Forwards")
        GPIO.output(Motor1A,GPIO.HIGH)
        GPIO.output(Motor1B,GPIO.LOW)
        GPIO.output(Motor1Enable,GPIO.HIGH)
        GPIO.output(Motor2A,GPIO.LOW)
        GPIO.output(Motor2B,GPIO.HIGH)
        GPIO.output(Motor2Enable,GPIO.HIGH)
        sleep(0.5)

def backward():
        print("Going Backwards")
        GPIO.output(Motor1A,GPIO.LOW)
        GPIO.output(Motor1B,GPIO.HIGH)
        GPIO.output(Motor1Enable,GPIO.HIGH)
        GPIO.output(Motor2A,GPIO.HIGH)
        GPIO.output(Motor2B,GPIO.LOW)
        GPIO.output(Motor2Enable,GPIO.HIGH)
        sleep(0.5)

def turnRight():
        print("Going Right")
        GPIO.output(Motor1A,GPIO.HIGH)
        GPIO.output(Motor1B,GPIO.LOW)
        GPIO.output(Motor1Enable,GPIO.HIGH)
        GPIO.output(Motor2A,GPIO.HIGH)
        GPIO.output(Motor2B,GPIO.LOW)
        GPIO.output(Motor2Enable,GPIO.HIGH)
        sleep(0.05)

def turnLeft():
        print("Going Left")
        GPIO.output(Motor1A,GPIO.LOW)
        GPIO.output(Motor1B,GPIO.HIGH)
        GPIO.output(Motor1Enable,GPIO.HIGH)
        GPIO.output(Motor2A,GPIO.LOW)
        GPIO.output(Motor2B,GPIO.HIGH)
        GPIO.output(Motor2Enable,GPIO.HIGH)
        sleep(0.5)

def backward():
        print("Going Backwards")
        GPIO.output(Motor1A,GPIO.LOW)
        GPIO.output(Motor1B,GPIO.HIGH)
        GPIO.output(Motor1Enable,GPIO.HIGH)
        GPIO.output(Motor2A,GPIO.HIGH)
        GPIO.output(Motor2B,GPIO.LOW)
        GPIO.output(Motor2Enable,GPIO.HIGH)
        sleep(0.5)

def turnRight():
        print("Going Right")
        GPIO.output(Motor1A,GPIO.HIGH)
        GPIO.output(Motor1B,GPIO.LOW)
        GPIO.output(Motor1Enable,GPIO.HIGH)
        GPIO.output(Motor2A,GPIO.HIGH)
        GPIO.output(Motor2B,GPIO.LOW)
        GPIO.output(Motor2Enable,GPIO.HIGH)
        sleep(0.05)

def turnLeft():
        print("Going Left")
        GPIO.output(Motor1A,GPIO.LOW)
        GPIO.output(Motor1B,GPIO.HIGH)
        GPIO.output(Motor1Enable,GPIO.HIGH)
        GPIO.output(Motor2A,GPIO.LOW)
        GPIO.output(Motor2B,GPIO.HIGH)
        GPIO.output(Motor2Enable,GPIO.HIGH)
        sleep(0.05)

def stop():
        print("Stopping")
        GPIO.output(Motor1A,GPIO.LOW)
        GPIO.output(Motor1B,GPIO.LOW)
        GPIO.output(Motor1Enable,GPIO.LOW)
        GPIO.output(Motor2A,GPIO.LOW)
        GPIO.output(Motor2B,GPIO.LOW)
        GPIO.output(Motor2Enable,GPIO.LOW)
        sleep(0.05)
```

Josecarlos FLórez

http://josecarlosflorez.com/

OMNIPRESENCE

a Remote Reality Platform

appCam.py

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
#       appCam.py
#       based on tutorial ==> https://blog.miguelgrinberg.com/post/video-streaming-with-flask
#       PiCam Local Web Server with Flask
# MJRoBot.org 19Jan18

from flask import Flask, render_template, Response

# Raspberry Pi camera module (requires picamera package)
from camera_pi import Camera

app = Flask(__name__)


@app.route('/')
def index():
    """Video streaming home page."""
    return render_template('camera.html')


def gen(camera):
    """Video streaming generator function."""
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')


@app.route('/video_feed')
def video_feed():
    """Video streaming route. Put this in the src attribute of an img tag."""
    return Response(gen(Camera()), mimetype='multipart/x-mixed-replace; boundary=frame')


if __name__ == '__main__':
    app.run(host='10.199.160.213', port=9000, debug=True)
```

camera_pi.py

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
#  camera_pi.py
#
#
#
import time
import io
import threading
import picamera


class Camera(object):
    thread = None  # background thread that reads frames from camera
    frame = None  # current frame is stored here by background thread
    last_access = 0  # time of last client access to the camera

    def initialize(self):
        if Camera.thread is None:
            # start background frame thread
            Camera.thread = threading.Thread(target=self._thread)
            Camera.thread.start()

            # wait until frames start to be available
            while self.frame is None:
                time.sleep(0)

    def get_frame(self):
        Camera.last_access = time.time()
        self.initialize()
        return self.frame

    @classmethod
    def _thread(cls):
        with picamera.PiCamera() as camera:
            # camera setup
            camera.resolution = (320, 240)
            camera.hflip = True
            camera.vflip = False

            # let camera warm up
            camera.start_preview()
            time.sleep(2)

            stream = io.BytesIO()
            for foo in camera.capture_continuous(stream, 'jpeg', use_video_port=True):
                # store frame
                stream.seek(0)
                cls.frame = stream.read()

                # reset stream for next frame
                stream.seek(0)
                stream.truncate()

                # if there hasn't been any clients asking for frames in
                # the last 10 seconds stop the thread
                if time.time() - cls.last_access > 10:
                    break
        cls.thread = None
```

Josecarlos FLórez

http://josecarlosflorez.com/

OMNIPRESENCE

a Remote Reality Platform

Index.html

```
<!DOCTYPE html>
<html>
	<head>
		<title>Remote Control</title>
		<meta name="viewport" content="width=device-width, initial-scale=1">
		<link rel="stylesheet" href='../static/style.css'/>
		<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
		<script src="static/js/submit.js"></script>

		<script src="https://js.api.here.com/v3/3.1/mapsjs-core.js" type="text/javascript" charset="utf-8"></script>
		<script src="https://js.api.here.com/v3/3.1/mapsjs-service.js" type="text/javascript" charset="utf-8"></script>
	</head>
<style>
</style>
	<body>
		<div><p class="last-interaction"><strong>Last Interaction: </strong>{{ time }}</p></div>
		<div class="last-interaction" style="background:transparent;margin-top:200px;padding-bottom:20px">
			<img src="static/img/telepresence.png" width="300">
			<h1><strong>OMNIPRESENCE </strong></h1><p>A Remote Reality Platform</p>
			<a style="background:#030303; color:#fff; padding:10px 70px;" href="/robotcontrol" class="button">i want to control a Robot</a>
		</div>

	<script>
	// Initialize the platform object:
	var platform = new H.service.Platform({
		'apikey': 'UnxL83c1CkljIqypsS_W4I-r7XCQX3DxV_eweFXVpUU'	// Initialize the platform object:
		});

		const lat = 41.3818;
		const long = 2.1685;

	// Obtain the default map types from the platform object
	var maptypes = platform.createDefaultLayers();

	// Instantiate (and display) a map object:
	var map = new H.Map(
		document.getElementById('mapContainer'),
		maptypes.vector.normal.map,
		{
			zoom: 10,
			center: { lat: lat, lng: long }
		});

	var marker = new H.map.Marker({ lat: lat, lng: long });

	// Add the marker to the map:
	map.addObject(marker);
	</script>
	</body>
</html>
```

Josecarlos FLórez

http://josecarlosflorez.com/

OMNIPRESENCE

a Remote Reality Platform

robotControl.html

```
<!DOCTYPE html>
<html>
	<head>
		<title>Remote Control</title>
		<meta name="viewport" content="width=device-width, initial-scale=1">
		<link rel="stylesheet" href='../static/style.css'/>
		<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
		<script src="static/js/submit.js"></script>
	</head>
<style>
</style>
	<body>
		<div><p class="last-interaction"><strong>Last Interaction: </strong>{{ time }}</p></div>
		<img src="{{ url_for('video_feed')}}" width="600px">
		<div><a href="/" class="button">Volver al Inicio</a></div>
		<div class="tabla">
		<table>
			<tr>
				<td>
				</td>
				<td class="boton">
					<form action="/2" method="POST">
						<input src="/static/img/adelante.png" type="image" name="direction" value="forward" />
						</br>
					</form>
				</td>
				<td>
				</td>
			</tr>
			<tr>
				<td class="boton">
					<form action="/1" method="POST">
						<input src="/static/img/left.png" type="image" name="direction" value="left" />
						</br>
					</form>
				</td>
				<td class="boton">
					<form action="/5" method="POST">
						<input src="/static/img/pause.png" type="image" name="direction" value="stop" />
						</br>
					</form>
				</td>

				<td class="boton">
					<form action="/3" method="POST">
						<input src="/static/img/right.png" type="image" name="direction" value="right" />
						</br>
					</form>
				</td>

			</tr>
			<tr>
				<td>
				</td>
				<td class="boton">
					<form action="/4" method="POST">
						<input src="/static/img/backward.png" type="image" name="direction" value="reverse" />
						</br>
					</form>
				</td>
				<td>

					<form action="/1" method="POST">
						<input src="/static/img/left.png" type="image" name="direction" value="left" />
						</br>
					</form>
				</td>
				<td class="boton">
					<form action="/5" method="POST">
						<input src="/static/img/pause.png" type="image" name="direction" value="stop" />
						</br>
					</form>
				</td>

				<td class="boton">
					<form action="/3" method="POST">
						<input src="/static/img/right.png" type="image" name="direction" value="right" />
						</br>
					</form>
				</td>

			</tr>
			<tr>
				<td>
				</td>
				<td class="boton">
					<form action="/4" method="POST">
						<input src="/static/img/backward.png" type="image" name="direction" value="reverse" />
						</br>
					</form>
				</td>
				<td>
				</td>

			</tr>
		</table>
		</div>
		<div>
			<form method="POST" action="/test">
				<input type="range" min="3" max="7" value="5" id="myRange" name="slider1"  />
				<input type="submit" value="submit">
			</form>
		</div>
	</body>
</html>
```

Josecarlos FLórez

http://josecarlosflorez.com/

OMNIPRESENCE

a Remote Reality Platform

styles.css

```
body{
	background:#fafafa;
	text-align:center;
}

.tabla{margin:20px auto;width:402px;padding:10px;border: 1px dotted #ccc;border-radius:30px;background:#fff;}

.table{
	width:300px;
	height:300px;
	margin:20px auto!important;
}

td.boton{
	background:#eeeeee;
	width:100px;
	height:100px;
	color:#ffffff;
	padding:10px;
}

.last-interaction{
	width:300px;
	background:#ffffff;
	padding:10px;
	margin:10px auto;
	border:1px solid #eee;
	border-radius:15px;
}
```

Josecarlos FLórez

http://josecarlosflorez.com/