SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA
Instituto Tecnológico de Morelia

INSTITUTO TECNOLÓGICO DE MORELIA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

“Espejito, Espejito”

Internet de las Cosas

Wendy Yunuén Arévalo Espinal
Thalia Raquel Velázquez Angeles
Xochitl Aramis Delgado Monroy
Emmanuel Alejandro Hernández Cornejo

Ing. en Tecnologías de la Información y Comunicación

PROFESOR:

Rogelio Ferreira Escutia

MORELIA, MICHOACÁN 31 de Mayo del 2018

Descripción del proyecto.

La idea de "Espejito, espejito" está inspirada inicialmente en " MagicMirror²" de Michael Teeuw.

El proyecto consiste en un vidrio de doble vista, el cual, con ayuda de un monitor muestra una aplicación web con módulos para visualizar la hora y fecha, el pronóstico del clima, cumplidos y la música en reproducción de Spotify, éste a su vez está conectado a una unidad de Raspberry Pi.

Aparte de los módulos, el espejo incluye dos sensores: botón (KY-004) y ultrasónico de proximidad (HC-RS04). El primero inicia y apaga el sistema operativo (Raspbian) mientras que el sensor de proximidad enciende el monitor al detectar objetos a 40 cm de distancia o lo apaga si no hay nadie frente al espejo.

Cada sensor guarda los datos correspondientes a la acción que realizan en la base de datos local y del servidor.

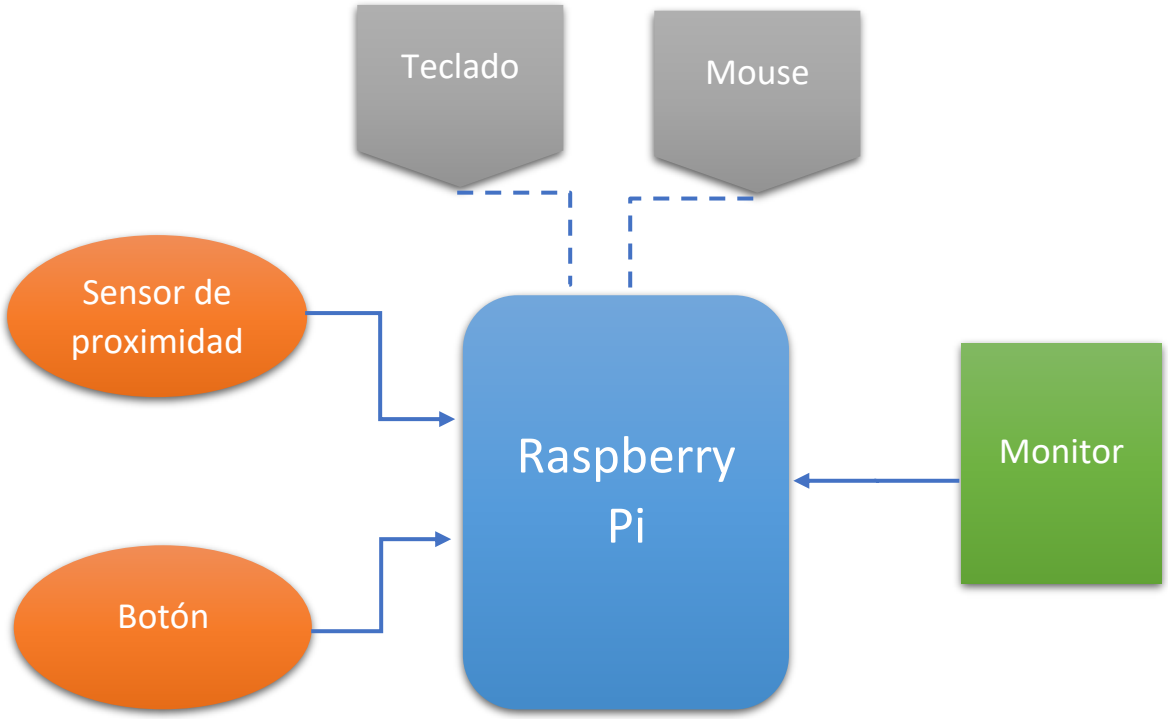
Los requisitos para lograr crear la parte funcional (software) son contar con un modelo de Raspberry Pi, instalar todas sus dependencias y el propio Magic Mirror para posteriormente añadir los módulos deseados desde los distintos repositorios con los que contamos.

Para detalles finales y configuraciones posteriores en cuando al diseño de la interfaz, se consiguió modificando el CSS.

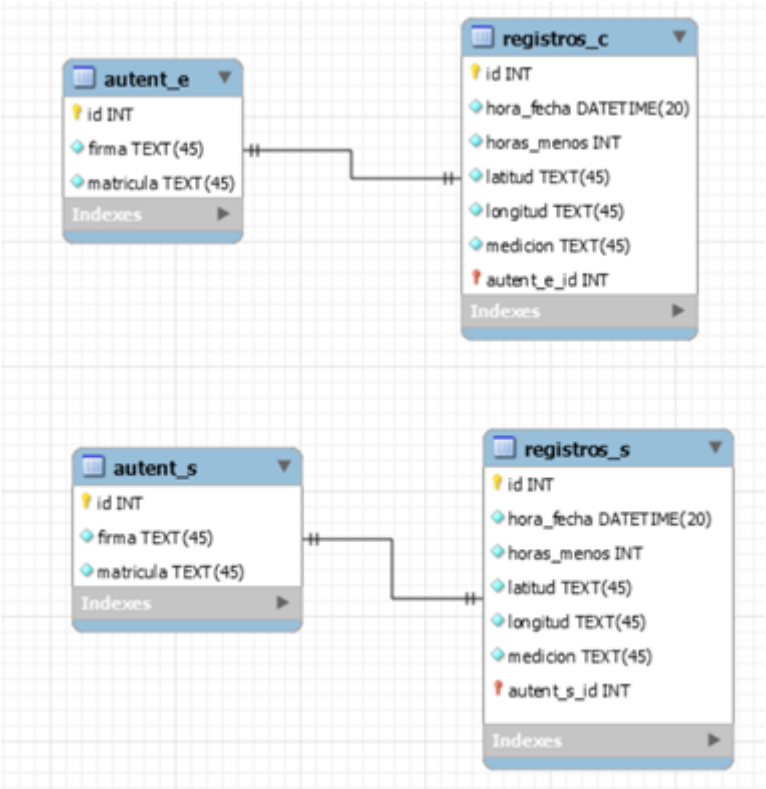
Costes

Elemento	Costo
Raspberry Pi 3	$ 1 000
Protección y difusor RP	$ 250
Sensor Ultrasónico de Proximidad HC-RS04	$ 30
Sensor Modulo Botón KY-004	$ 40
Cables de puente y protoboard	$ 50
Marco de madera a la medida	$ 175
Vidrio Sol-Lite Claro a la medida	$ 200
Monitor Samsung 24” HDMI	$ 3 200
Teclado genérico	$ 100
Mouse genérico	$ 100
Madera para soporte, clavos, cinchos.	$ 55
TOTAL	$ 5 200

Diagrama de conexión del Proyecto.



Estructura de las bases y tablas empleadas.



Estructura de tabla | Vista de relaciones

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	firma	tinytext	utf8_general_ci		No	*Ninguna*			Cambiar Eliminar Más
3	matricula	tinytext	utf8_general_ci		No	*Ninguna*			Cambiar Eliminar Más

Estructura de tabla | Vista de relaciones

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	firma	tinytext	utf8_general_ci		No	*Ninguna*			Cambiar Eliminar Más
3	matricula	tinytext	utf8_general_ci		No	*Ninguna*			Cambiar Eliminar Más

Estructura de tabla | Vista de relaciones

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int(11)			No	*Ninguna*			Cambiar Eliminar Más
2	hora_fecha	datetime(6)			No	*Ninguna*			Cambiar Eliminar Más
3	horas_menos	int(11)			No	*Ninguna*			Cambiar Eliminar Más
4	latitud	tinytext	utf8_general_ci		No	*Ninguna*			Cambiar Eliminar Más
5	longitud	tinytext	utf8_general_ci		No	*Ninguna*			Cambiar Eliminar Más
6	medicion	tinytext	utf8_general_ci		No	*Ninguna*			Cambiar Eliminar Más
7	autent_e_id	int(11)			No	*Ninguna*			Cambiar Eliminar Más

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int(11)			No	*Ninguna*			Cambiar Eliminar Más
2	hora_fecha	datetime(6)			No	*Ninguna*			Cambiar Eliminar Más
3	horas_menos	int(11)			No	*Ninguna*			Cambiar Eliminar Más
4	latitud	tinytext	utf8_general_ci		No	*Ninguna*			Cambiar Eliminar Más
5	longitud	tinytext	utf8_general_ci		No	*Ninguna*			Cambiar Eliminar Más
6	medicion	tinytext	utf8_general_ci		No	*Ninguna*			Cambiar Eliminar Más
7	autent_s_id	int(11)			No	*Ninguna*			Cambiar Eliminar Más

Todos los códigos empleados.

Código para sensor de proximidad. (apagar_pantalla.py)

Descripción: Enciende y apaga el monitor dependiendo de la distancia a la que se encuentre un objetivo. Cuando alguien se aproxima a menos de 40 cm al espejo la pantalla se enciende, cuando no hay nadie, se apaga.

Se guarda en la base de datos la distancia a la que se acerco el objetivo cuando se encendió la pantalla. Si no tiene conexión a internet el sensor no funcionará ni enviará nada a las bases de datos.

```
#Libraries
import RPi.GPIO as GPIO
import time
import os
import MySQLdb
import requests
import datetime
import hashlib

db = MySQLdb.connect(host="192.168.1.70",    # your host, usually localhost
                     user="iots",         # your username
                     passwd="iot2018",  # your password
                     db="registros_serv")        # name of the data base

dbLocal = MySQLdb.connect(host="localhost",    # your host, usually localhost
                     user="root",         # your username
                     passwd="",  # your password
                     db="registros_local")        # name of the data base


cur = db.cursor() #creamos un cursor para el servidor
curLocal = dbLocal.cursor() #creamos un cursor para el local

datos ={
"considerIp": "true"
}

url = "https://www.googleapis.com/geolocation/v1/geolocate?key=AIzaSyB03BzPDT2fGeV6uyqAKUx4H4aUAAIsCZU"


#GPIO Mode (BOARD / BCM)
GPIO.setmode(GPIO.BCM)

#set GPIO Pins
GPIO_TRIGGER = 18
GPIO_ECHO = 24

#set GPIO direction (IN / OUT)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)
def distance():
    # set Trigger to HIGH
    GPIO.output(GPIO_TRIGGER, True)

    # set Trigger after 0.01ms to LOW
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()

    # save StartTime
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()

    # save time of arrival
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()

    # time difference between start and arrival
    TimeElapsed = StopTime - StartTime
    # multiply with the sonic speed (34300 cm/s)
    # and divide by 2, because there and back
    distance = (TimeElapsed * 34300) / 2

    return distance

if __name__ == '__main__':
    try:
        while True:
            dist = distance()
            print ("Measured Distance = %.1f cm" % dist)
            response = requests.post(url, json=datos)
            hf = datetime.datetime.now() #obtenemos el tiempo y fecha
            lat = str(response.json()['location']['lat']) #la latitud
            lng = str(response.json()['location']['lng']) #la longitud
        if dist <  40.0:
                os.system('/usr/bin/vcgencmd display_power 1')
                sql = "SELECT firma, matricula FROM autent_s WHERE id = 1";
                sqlLocal = "SELECT firma, matricula FROM autent_c WHERE id = 1";
```

```
            try:
                cur.execute(sql)
                curLocal.execute(sqlLocal)
                results = cur.fetchall()
                for row in results:
                    firma = row[0]
                    mat = row[1]
                resultsLocal = curLocal.fetchall()
                for rowLocal in resultsLocal:
                    firmaLocal = rowLocal[0]
                    matLocal = rowLocal[1]
                h = hashlib.new("md5", firma+mat)
                print h.hexdigest()
                hLocal = hashlib.new("md5", firmaLocal+matLocal)
                print hLocal.hexdigest()
                if h.hexdigest() == hLocal.hexdigest():
                    sql = "INSERT INTO registros_s(hora_fecha,horas_menos, latitud, longitud, medicion, id_sensor) VALUES ('%s', 0, '%s',
                    '%s', '%s', 1 )" % (hf, lat, lng, ('Acercamiento de '+dist+' cm, ENCENDIDO'))
                    sqlLocal = "INSERT INTO registros_c(hora_fecha,horas_menos, latitud, longitud, medicion, id_sensor) VALUES ('%s', 0,
                    '%s', '%s', '%s', 1 )" % (hf, lat, lng, ('Acercamiento de '+dist+' cm, ENCENDIDO'))

                    try:
                       cur.execute(sql)
                       curLocal.execute(sqlLocal)
                       db.commit()
                       dbLocal.commit()
                       print ("Insercion exitosa")
                    except:
                       # Rollback si hay error
                       db.rollback()
                       dbLocal.rollback()
                       print ("Insercion fallida")
                else:
                    print ("Error: Las llaves no coinciden")
            except:
                print "Error: No hay datos"

            time.sleep(1)
    elif dist > 40.0:
            os.system('/usr/bin/vcgencmd display_power 0')
            time.sleep(1)
```

```
    # Reset by pressing CTRL + C
except KeyboardInterrupt:
    print("Measurement stopped by User")
    GPIO.cleanup()
    db.close()
    dbLocal.close()
```

Código para sensor botón (on/off SO). (shutdown.py)

Descripción: Hace lecturas del botón; al presionarlo se apaga el sistema operativo de forma segura y se registra que fue apagado en la base de datos. Al presionar el botón de nuevo enciende el sistema.

```
import os
import time
import MySQLdb
import requests
import datetime
import hashlib
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(03,GPIO.IN)


db = MySQLdb.connect(host="192.168.1.70",    # your host, usually localhost
                     user="iots",         # your username
                     passwd="iot2018",  # your password
                     db="registros_serv")        # name of the data base

dbLocal = MySQLdb.connect(host="localhost",    # your host, usually localhost
                     user="root",         # your username
                     passwd="",  # your password
                     db="registros_local")        # name of the data base


cur = db.cursor() #creamos un cursor para el servidor
curLocal = dbLocal.cursor() #creamos un cursor para el local

datos ={
"considerIp": "true"
}

url = "https://www.googleapis.com/geolocation/v1/geolocate?key=AIzaSyB03BzPDT2fGeV6uyqAKUx4H4aUAAIsCZU"


while True:
    response = requests.post(url, json=datos)
    hf = datetime.datetime.now() #obtenemos el tiempo y fecha
    lat = str(response.json()['location']['lat']) #la latitud
    lng = str(response.json()['location']['lng']) #la longitud
    print GPIO.input(03)
    if(GPIO.input(03) == False):
        sql = "SELECT firma, matricula FROM autent_s WHERE id = 2";
        sqlLocal = "SELECT firma, matricula FROM autent_c WHERE id = 2";
      try:
          cur.execute(sql)
          curLocal.execute(sqlLocal)
          results = cur.fetchall()
          for row in results:
              firma = row[0]
              mat = row[1]
          resultsLocal = curLocal.fetchall()
          for rowLocal in resultsLocal:
              firmaLocal = rowLocal[0]
              matLocal = rowLocal[1]
          h = hashlib.new("md5", firma+mat)
          #print h.hexdigest()
          hLocal = hashlib.new("md5", firmaLocal+matLocal)
          #print hLocal.hexdigest()
          if h.hexdigest() == hLocal.hexdigest():
              sql = "INSERT INTO registros_s(hora_fecha,horas_menos, latitud, longitud, medicion, id_sensor) VALUES ('%s', 0, '%s', '%s', '%s',
              2 )" % (hf, lat, lng, 'Boton presionado')
              sqlLocal = "INSERT INTO registros_c(hora_fecha,horas_menos, latitud, longitud, medicion, id_sensor) VALUES ('%s', 0, '%s', '%s',
              '%s', 2 )" % (hf, lat, lng, 'Boton presionado')
              try:
                  cur.execute(sql)
                  curLocal.execute(sqlLocal)
                  db.commit()
                  dbLocal.commit()
                  print ("Insercion exitosa")
              except:
                  # Rollback si hay error
                  db.rollback()
                  dbLocal.rollback()
                  print ("Insercion fallida")
          else:
              print ("Error: Las llaves no coinciden")
      except:
          print "Error: No hay datos"
      db.close()
      dbLocal.close()
      os.system("sudo shutdown -h now")
      break
  time.sleep(1)
```

Estructura de base de datos servidor y cliente. (base_espejo.sql)

Breve descripción: Es la estructura de la base de datos del cliente y el servidor, algunos se ven reflejados directamente en la aplicación.

```
-- Crear base de datos registros_local en raspberry

CREATE TABLE autent_c(id int not null PRIMARY KEY AUTO_INCREMENT,firma TEXT NOT NULL, matricula TEXT NOT NULL);

CREATE TABLE registros_c(id int not null primary key auto_increment, hora_fecha datetime not null, horas_menos int not null, latitud text not
null, longitud text not null, medicion text not null, id_sensor int not null, FOREIGN KEY (id_sensor) REFERENCES autent_c(id));

-- Insercion de sensores
INSERT INTO autent_c (id,firma, matricula) values (1,'ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAndGKbn5hAktLoT5TKvrUZh6JVh5FH9vDwXUssPNP8aIFjnRJC3BCtboNiMxrWqT7OVR0dDJnV9X28GJWDFe1LVfn3FHqxe/5BsS911Z6fvbTWVOvLT2Elq
vnJUX4AQBrqP8RVvMQmD98RGtnxclOb6ScZ85vqRPs8E7S8AfwQuy28HF600zgRZqIjFJfUsPXVP2BnNDRaMqB46/ZFwKofd010SltGe/iVpglP1H2HSt8EwNxCUNs+03rEzaokEanqztF33ws
sQN4A0nYCSca/citXeFChWgpPwspd9+LxM6L0PPETePg3qlXH5nTajLys1mPjAKH1ppjE8p3ZOEMLQ== rsa-key-20180530','HC-SR04');

INSERT INTO autent_c (id,firma, matricula) values (2,'ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEA0yHhOgph9zJ/UMseJR+4DicOsAiARkl9AMnPqNkew/5aLmgXradeVUR67kK8RwEcM8ix3Tim1Un3HrEy4SzeNG/GOvLwE5D9XdfGKGy6gHpkYd529XcYgg
YEVSMJZnyxBA9wf0cBp4sbmIvWrpS4Xb5ayyliQAXJ+cqx+5A73H8mpnXhbaGTV0qeMregvSzzI/xNLftgx8kAkJOduntuzYrKpqVDnyVPLM4NNAmq3X+cKX8XrZBX94RDLSgjk8W8UHZ7gSSY
G5xHtIRD/HxUe6VY+DyRZZze/abXbuMHTQheuST00jsRmDbpI81nVeIYXctXqD/7Q37dEX1owyKYMw== rsa-key-20180530','KY-004');


-- Crear base de datos registros_serv en servidor (laptop)

CREATE TABLE autent_s(id int not null PRIMARY KEY AUTO_INCREMENT,firma TEXT NOT NULL, matricula TEXT NOT NULL);

CREATE TABLE registros_s(id int not null primary key auto_increment, hora_fecha datetime not null, horas_menos int not null, latitud text not
null, longitud text not null, medicion text not null, id_sensor int not null, FOREIGN KEY (id_sensor) REFERENCES autent_s(id));

-- Insercion de sensores
INSERT INTO autent_s (id,firma, matricula) values (1,'ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAndGKbn5hAktLoT5TKvrUZh6JVh5FH9vDwXUssPNP8aIFjnRJC3BCtboNiMxrWqT7OVR0dDJnV9X28GJWDFe1LVfn3FHqxe/5BsS911Z6fvbTWVOvLT2Elq
vnJUX4AQBrqP8RVvMQmD98RGtnxclOb6ScZ85vqRPs8E7S8AfwQuy28HF600zgRZqIjFJfUsPXVP2BnNDRaMqB46/ZFwKofd010SltGe/iVpglP1H2HSt8EwNxCUNs+03rEzaokEanqztF33ws
sQN4A0nYCSca/citXeFChWgpPwspd9+LxM6L0PPETePg3qlXH5nTajLys1mPjAKH1ppjE8p3ZOEMLQ== rsa-key-20180530','HC-SR04');

INSERT INTO autent_s (id,firma, matricula) values (2,'ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEA0yHhOgph9zJ/UMseJR+4DicOsAiARkl9AMnPqNkew/5aLmgXradeVUR67kK8RwEcM8ix3Tim1Un3HrEy4SzeNG/GOvLwE5D9XdfGKGy6gHpkYd529XcYgg
YEVSMJZnyxBA9wf0cBp4sbmIvWrpS4Xb5ayyliQAXJ+cqx+5A73H8mpnXhbaGTV0qeMregvSzzI/xNLftgx8kAkJOduntuzYrKpqVDnyVPLM4NNAmq3X+cKX8XrZBX94RDLSgjk8W8UHZ7gSSY
G5xHtIRD/HxUe6VY+DyRZZze/abXbuMHTQheuST00jsRmDbpI81nVeIYXctXqD/7Q37dEX1owyKYMw== rsa-key-20180530','KY-004');
```

Configuración del espejo. (config.js)

Descripción: Este archivo de configuración se incluye al instalar el módulo del espejo, pero aquí se realizan las modificaciones en el json de los módulos para posicionarlos o conectarlos. Los módulos modificados fueron compliments y MMM-NowPlayingOnSpotify.

```
var config = {
    address: "localhost", // Address to listen on, can be:
                          // - "localhost", "127.0.0.1", "::1" to listen on loopback interface
                          // - another specific IPv4/6 to listen on a specific interface
                          // - "", "0.0.0.0", "::" to listen on any interface
                          // Default, when address config is left out, is "localhost"
    port: 8080,
    ipWhitelist: ["127.0.0.1", "::ffff:127.0.0.1", "::1"], // Set [] to allow all IP addresses
                                                           // or add a specific IPv4 of 192.168.1.5 :
                                                           // ["127.0.0.1", "::ffff:127.0.0.1", "::1", "::ffff:192.168.1.5"],
                                                           // or IPv4 range of 192.168.3.0 --> 192.168.3.15 use CIDR format :
                                                           // ["127.0.0.1", "::ffff:127.0.0.1", "::1", "::ffff:192.168.3.0/28"],

    language: "es",
    timeFormat: 12,
    units: "metric",

    modules: [
        {
            module: "alert",
        },
        {
            module: "updatenotification",
            position: "top_bar"
        },
        {
            module: "clock",
            position: "top_left"
        },
        {
            module: "compliments",
            position: "lower_third",
            config:{
                compliments: {
                    anytime: [
                    "¡Hola, sexy!"
                    ],
                    morning: [
                    "¡Buenos dias, lindura!",
                    "Te ves bien hoy."
                    ],
                    afternoon:[
                    "¡Te ves sexy hoy!",
                    "¡Luces bien!"
                    ],
                    evening:[
                    "¡Wow! ¡Qué sexy!",
                    "Pichpiris for the win!"
                    ]
                }
            }
        },
        {
            module: "weatherforecast",
            position: "top_right",
            header: "Weather Forecast",
            config: {
                location: "México",
                locationID: "3530597",  //ID from http://www.openweathermap.org/help/city_list.txt
                appid: "6968827e46c96bd5b59fd8a6beb198f0"
            }
        },

        {
  module: "MMM-NowPlayingOnSpotify",
  position: "bottom",

  config: {
    clientID: "6f14454149414522b912d8d0a8c24f48",
    clientSecret: "e8f89b40cba744ce91391c3013105e9b",
    accessToken: "BQDR1Kw4Qne3BFShe1y_52_Ae5890gQD5vA-JNnk1FJRn7bznYkNWY7tKeNNVQ-gkjY97w6r6mao3fO87j9emKrIATDR13m2Syidpp6SBJ4CxmyKJm4L-
    WEB93pWEZqMSGIJGaXlhSHGwVgaeHuyb39Eg1A",
    refreshToken: "AQAz_De6Xa2Ehrb_Zw8etArWzCYke27DmSrwZpTfQPrllIrK0zwHyG8VqttTLGPcITWDBR-
    k6zHWu07hQuZwXwcaAtEL4kT2XS4q2wXuWCqq0TvegxDbZsuFPXRhHgShmD4",
    showCoverArt: true
}
},
```

```
{
    module: 'MMM-Facial-Recognition',
    config: {
        // 1=LBPH | 2=Fisher | 3=Eigen
        recognitionAlgorithm: 1,
        // Threshold for the confidence of a recognized face before it's considered a
        // positive match.  Confidence values below this threshold will be considered
        // a positive match because the lower the confidence value, or distance, the
        // more confident the algorithm is that the face was correctly detected.
        lbphThreshold: 50,
        fisherThreshold: 250,
        eigenThreshold: 3000,
        // force the use of a usb webcam on raspberry pi (on other platforms this is always true automatically)
        useUSBCam: true,
        // Path to your training xml
        trainingFile: 'modules/MMM-Facial-Recognition-Tools/training.xml',
        // recognition intervall in seconds (smaller number = faster but CPU intens!)
        interval: 2,
        // Logout delay after last recognition so that a user does not get instantly logged out if he turns away from the mirror for a few
        seconds
        logoutDelay: 15,
        // Array with usernames (copy and paste from training script)
        users: ['Emma','Wendy','Thalia','Aramis'],
        //Module set used for strangers and if no user is detected
        defaultClass: "default",
        //Set of modules which should be shown for every user
        everyoneClass: "everyone",
        // Boolean to toggle welcomeMessage
        welcomeMessage: true
    }
},

    ]

};

/************** DO NOT EDIT THE LINE BELOW **************/
if (typeof module !== "undefined") {module.exports = config;}
```

Servicio para habilitar espejo. (mirror.service)

Descripción: Para iniciar el espejo de forma automática al iniciar el Raspberry la solución fue crear un servicio para habilitarlo al inicio, simplemente se agrega a los servicios del sistema.

```
[Service]
WorkingDirectory=/home/pi/MagicMirror
ExecStart=/usr/bin/npm start
Restart=always
StandardOutput=inherit
StandardError=inherit
User=pi

[Install]
WantedBy=multi-user.target
```

Script para ejecutar los archivos. (conjunto.sh)

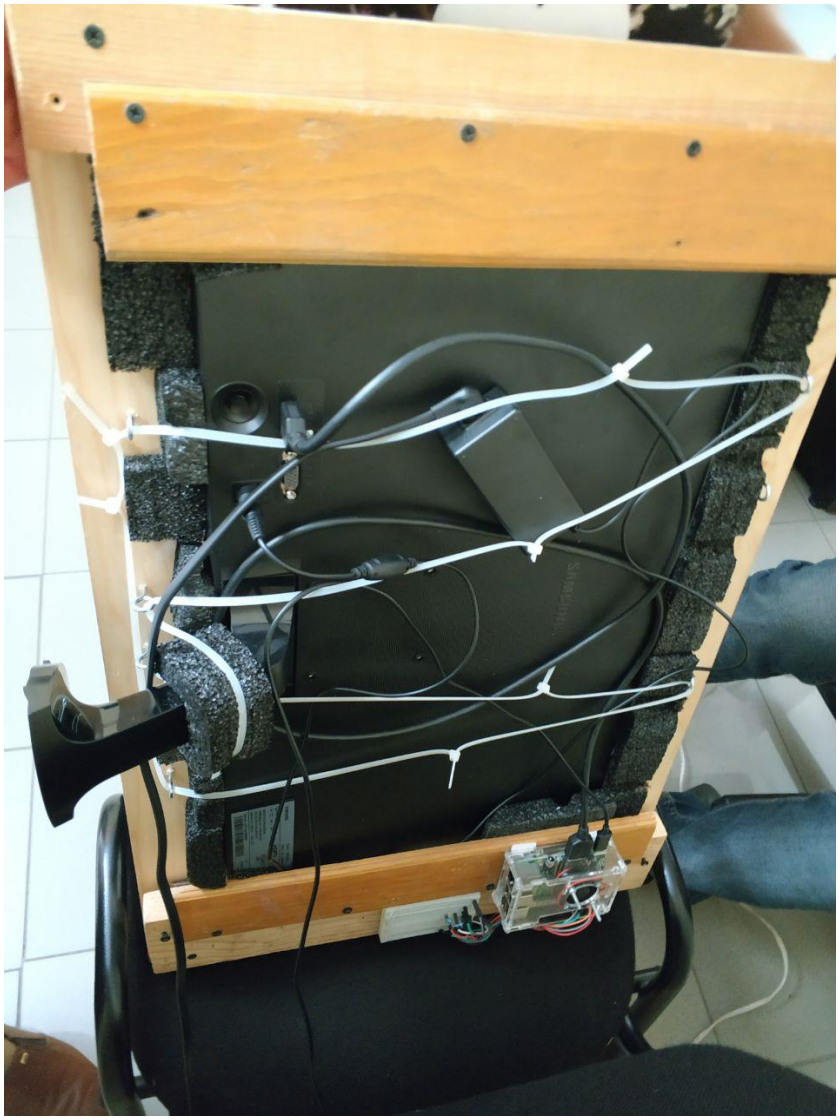
Descripción: Script para ejecutar los archivos apagar_pantalla.py y shutdown.py al momento del arranque del sistema, agregado a sudo crontab –e con el comando @ reboot sudo sh conjunto.sh.

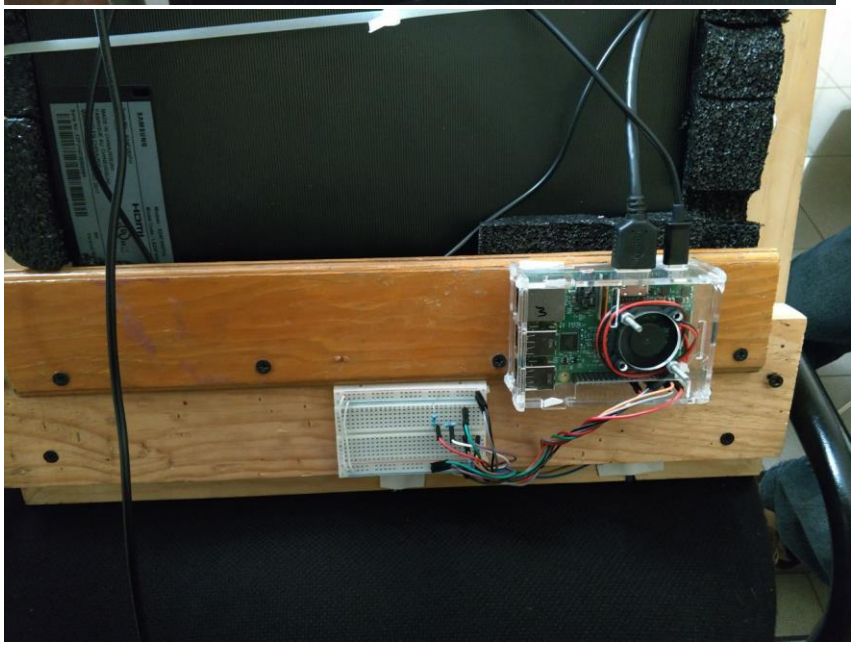
```
#!/bin/bash

sudo python /home/pi/shutdown.py &
sudo python /home/pi/apagar_pantalla.py
```

Fotos detalladas del Prototipo.

Espejo visto de frente:

Espejo visto desde atrás.





Sensores del espejo.