

**Arduino & RasPberry**

**David Moreno Moreno**

**Fernando Donaire GarcÍa**

**Sorin Gavrila**

**Grupo 6 – TRABAJO EN GRUPO 3**

dESARROLLO TECNOLOGÍAS EMERGENTES

gRAdo SISTEMAS DE LA INFORMACIÓN

Contenido

[1. Autores del trabajo, planificación y entrega 3](#_Toc449381051)

[1.1 Autores 3](#_Toc449381052)

[1.2 Planificación 3](#_Toc449381053)

[1.3 Entrega 3](#_Toc449381054)

[2. Requisitos del prototipo a implementar 4](#_Toc449381055)

[2.1 Requisitos funcionales 4](#_Toc449381056)

[2.2 Otros requisitos 4](#_Toc449381057)

[3. Criterios de comparación en la implementación 5](#_Toc449381058)

[3.1 Criterio 1: Nombre del criterio 5](#_Toc449381059)

[3.2 Criterio 2: Nombre del criterio 5](#_Toc449381060)

[3.N Criterio N: Nombre del criterio 5](#_Toc449381061)

[4. Proyecto de implementación de un prototipo del sistema utilizando Arduino 6](#_Toc449381062)

[4.1 Documentación de diseño 6](#_Toc449381063)

[4.2 Documentación de construcción 6](#_Toc449381064)

[4.3 Documentación de pruebas 6](#_Toc449381065)

[4.4 Documentación de instalación 6](#_Toc449381066)

[4.5 Manual de usuario 6](#_Toc449381067)

[5. Proyecto de implementación de un prototipo del sistema utilizando Raspberry 7](#_Toc449381068)

[5.1 Documentación de diseño 7](#_Toc449381069)

[5.1.1 Diagrama de uso lógico 7](#_Toc449381070)

[5.1.2 Diseño interfaz de usuario 7](#_Toc449381071)

[5.2 Documentación de construcción 7](#_Toc449381072)

[5.2.1 Diagrama de conexiones Raspberry PI 7](#_Toc449381073)

[5.2.2 Diagrama de uso lógico del script de Raspberry PI 8](#_Toc449381074)

[5.2.3 Diagrama de uso lógico de la API 10](#_Toc449381075)

[5.2.4 Diagrama de la base de datos 11](#_Toc449381076)

[5.2.5 Diagrama de uso lógico de la Interfaz de Usuario 12](#_Toc449381077)

[5.3 Documentación de pruebas 15](#_Toc449381078)

[5.4 Documentación de instalación 16](#_Toc449381079)

[5.4.1 Instalar el sistema operativo de Raspberry 16](#_Toc449381080)

[5.4.2 Configurar el sistema operativo de Raspberry 19](#_Toc449381081)

[5.4.3 Configurar servidor web 21](#_Toc449381082)

[5.5 Manual de usuario 22](#_Toc449381083)

[6. Comparación de las dos implementaciones 24](#_Toc449381084)

[6.1 Evaluación de los criterios en la implementación usando la tecnología A 24](#_Toc449381085)

[6.2 Evaluación de los criterios en la implementación usando la tecnología B 24](#_Toc449381086)

[7. Comparación de la implementación de las tecnologías 25](#_Toc449381087)

[8. Conclusiones 27](#_Toc449381088)

# Autores del trabajo, planificación y entrega

## Autores

EL grupo 6 está formado por:

* David Moreno Moreno (Coordinador del grupo)
* Fernando Donaire García
* Sorín Gavrila

Se encargaran de comparar las tecnologías de Arduino y Raspberry.

## Planificación

El peso de este trabajo en la calificación total de la asignatura es de un 30%, por tanto requiere de una dedicación de 45 horas del total de 150 horas de la asignatura.

Al estar formado por 3 integrantes el tiempo total de desarrollo de dicho proyecto será 135 horas, repartidas en el siguiente diagrama de Gantt.

*[TG3 – Arduino&Rapsberry](https://app.ganttpro.com/shared/token/8a64c8d85719ff297de6eeec9b232c698f054c99a9a3a62e7146398537884abc)*

El reparto de tareas ha sido equitativo, diferenciando las tareas del coordinador por un lado, y la del resto de integrantes del grupo. Cada miembro del grupo se ha especializado en una tecnología para que la información y motivación estuviera orientada al mismo entorno.

## Entrega

Incluimos el enlace (URL) a un repositorio en GitHub donde incorporamos nuestro trabajo y archivos:

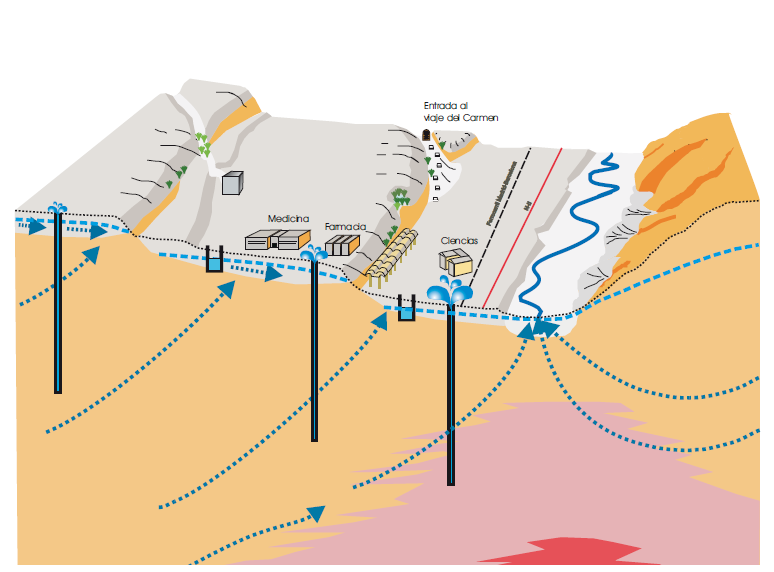
<https://github.com/dmoreno19949/DTE_TG_ARDUINO_RAPSBERRY>

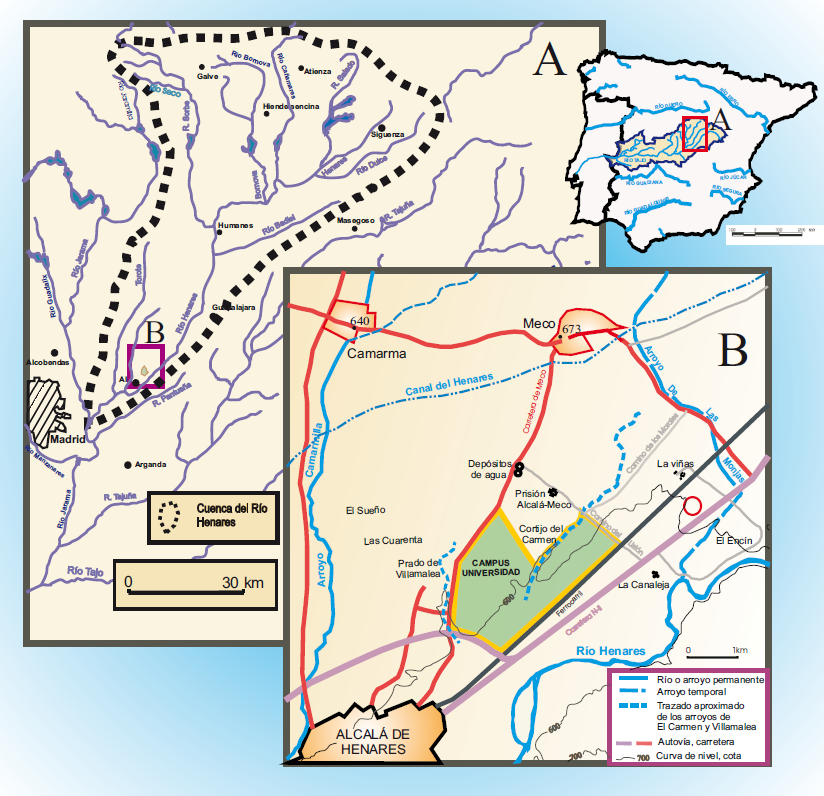
En dicho repositorio hemos incluido una carpeta TG3 donde se encuentran los siguientes archivos:

* **Trabajo terminado:** TG3\_final.docx
* **Presentación del trabajo:** TG3\_final.pptx
* **Prototipos obtenidos implementando cada una de las tecnologías** (deben incluir el código fuente y todos los archivos necesarios para la instalación y uso de cada prototipo):
  + PrototipoTecnologiaArduino\_final.zip (o .rar)
  + PrototipoTecnologiaRaspberry\_final.zip (o .rar)

# Requisitos del prototipo a implementar

La Universidad de Alcalá está presentando a concurso público el proyecto “**SmartWater – Fase 1: Desarrollo IoT**” que forma parte del plan de medidas de ahorro y sostenibilidad medioambiental “**SaveTheWater-UAH2020**” para calcular el Balance hídrico medio anual aportado por la lluvia y así evitar desgastar los acuíferos en casos de estrés hídrico.





Esta primera fase del proyecto tiene como objetivo implementar una red de sensores a lo largo del campus externo para monitorizar el caudal del agua en los días de lluvia para obtener una media del caudal exacto.

En todos los tejados de todas las facultades se instalarán unas cajas de unas dimensiones y superficies determinadas con una estructura tipo embudo que almacenará el agua de manera temporal hasta que pase por un sensor que mida el caudal del agua. La estructura y el sensor forman parte de otro proyecto interno de la universidad, pero estos sensores enviarán datos a la plataforma IoT que enviará y almacenará los registros en una base de datos común de la universidad para su posterior tratamiento.



Por tanto el alcance del proyecto se limita a implementar la parte IoT.

Queda fuera del alcance del proyecto:

* Diseñar e implementar la estructura de recolección del agua
* El sensor lo elige la Universidad
* El procesamiento, estadísticas, etc… forman parte de la Fase 2. Simplemente se implementará una interfaz para ver los datos en tiempo real.

A continuación se detallan todos los requisitos:

## Requisitos funcionales

Los requisitos funcionales deben ser los mismos para las dos implementaciones.

En la siguiente tabla se indicará el catálogo de requisitos funcionales del sistema.

| **REQ.** | **DESCRIPCIÓN** |
| --- | --- |
| RF01 | …. |
| RF02 | …. |
|  |  |

## Otros requisitos

Se pueden incluir aquí otros requisitos para el prototipo que no puedan considerarse como funcionales. Por ejemplo, requisitos de datos, de seguridad, de interfaz de usuario, de rendimientos, etc.

Se puede dejar libertad

En la siguiente tabla se indicará el catálogo de requisitos no funcionales del sistema.

| **REQ.** | **DESCRIPCIÓN** |
| --- | --- |
| R01 | …. |
| R02 | …. |
|  |  |

# Criterios de comparación en la implementación

En el trabajo TG2 se definieron criterios de comparación de las dos tecnologías a nivel teórico.

En este trabajo hay que definir criterios para la comparación de la implementación de las tecnologías en la construcción del prototipo de sistema de ejemplo, cuyos requisitos son los establecidos en el apartado 2.

Se trata de criterios del tipo” “horas empleadas en el desarrollo del sistema”, “velocidad de funcionamiento del sistema”, “recursos necesarios”, etc.

## Criterio 1: Nombre del criterio

Por cada criterio hay que indicar el nombre, una breve descripción, y el tipo de valor a asignar al criterio.

Por ejemplo, si se comparan dos herramientas CASE realizar el diseño UML de un mismo sistema, un criterio podría ser:

*Nombre del criterio: Tiempo de creación del diagrama de clases del sistema.*

*Descripción: Horas invertidas en la creación del diagrama de clases utilizando el editor de la herramienta.*

*Tipo de valor: Numérico (horas).*

## Criterio 2: Nombre del criterio

## 3.N Criterio N: Nombre del criterio

# Proyecto de implementación de un prototipo del sistema utilizando Arduino

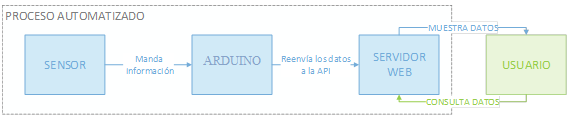
Se trata de incluir en este apartado la documentación del desarrollo del proyecto de implementación, utilizando la tecnología A, del sistema cuyos requisitos funcionales se enumeraron en el apartado 2.

## Documentación de diseño

Hay que incluir la descripción del diseño del prototipo, incluyendo diagramas, y el diseño de la interfaz de usuario.

### **Diagrama de uso lógico**

Se detalla las funciones y acciones que tiene cada componente.



### **Diseño interfaz de usuario**



## Documentación de construcción

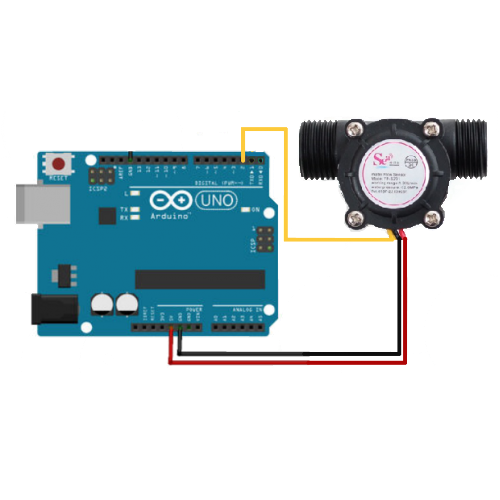
Hay que incluir una descripción de la construcción del prototipo, incluyendo algún extracto de código fuente. No es necesario todo el código. Sólo algún extracto para ver cómo se ha comentado.

### **Diagrama de conexiones Arduino**

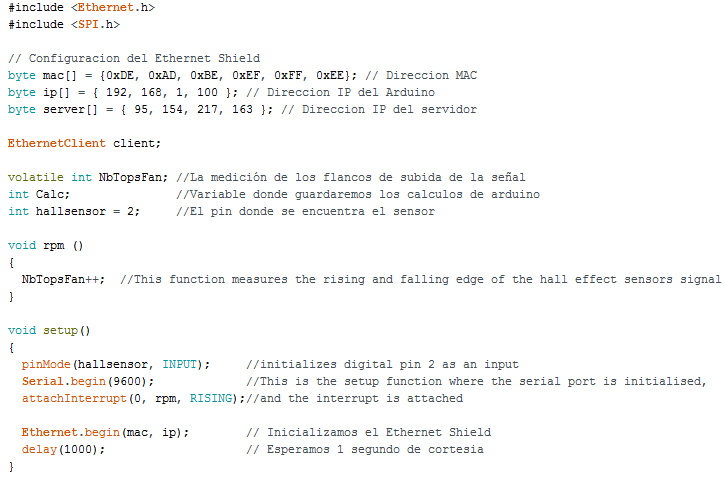
En primer lugar hay recalcar que a diferencia de rapsberry, arduino necesita incluir Ethernet Shield si queremos dotar a la placa microcontroladora de conexión a la red, por lo que será preciso la conexión de un cable Ethernet de clavija rj45 para la conexión ya sea con el ordenador, o con una clavija rj45 con conexión a internet.

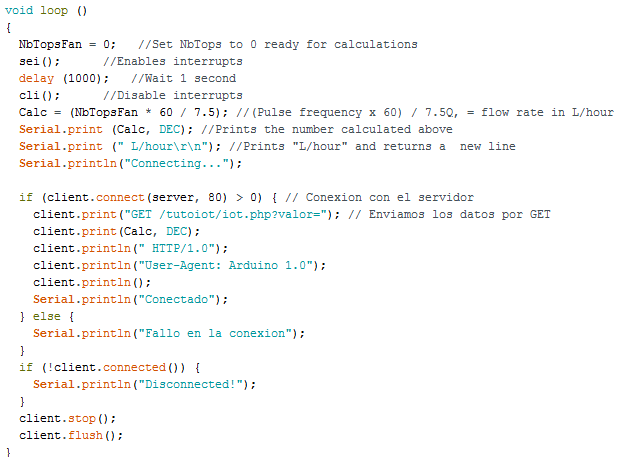


Por otro lado la conexión del caudalimetro al arduino se realiza atreves del pin 2 y los pines correspondientes de GND y 5v+. De esta forma el cable rojo y negro dotara el caudalimetro del circuito eléctrico necesario para que funcione. El cable amarillo será el encargado de transportar los datos con las mediciones pertinentes.



### **Diagrama de uso lógico del script de Arduino**





### **Diagrama de uso lógico de la API**



Script completo:

<?php

// iot.php

// Importamos la configuración

require("config.php");

// Leemos los valores que nos llegan por GET

$valor = mysqli\_real\_escape\_string($con, $\_GET['valor']);

$origen = mysqli\_real\_escape\_string($con, $\_GET['origen']);

// Esta es la instrucción para insertar los valores

//$query = "INSERT INTO wp\_valores(valor) VALUES('".$valor."')";

$query = "INSERT INTO wp\_valores(origen, valor) VALUES('".$origen."','".$valor."')";

// Ejecutamos la instrucción

mysqli\_query($con, $query);

mysqli\_close($con);

print "Recibido origen:".$origen." valor:".$valor;

?>

### **Diagrama de uso lógico de la API**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre tabla: | wp\_valores |  |  |  |
| Campo: | Id | Hora | Origen | valor |
| Tipo: | Int autoincrementado | Timestamp - Fechado de los datos | Varchar | Varchar |
| Valores: | - | fecha | Arduino/Rasp | texto libre |

Script .sql :

-- phpMyAdmin SQL Dump

-- version 4.5.1

-- http://www.phpmyadmin.net

--

-- Servidor: 127.0.0.1

-- Tiempo de generación: 21-04-2016 a las 01:01:47

-- Versión del servidor: 10.1.9-MariaDB

-- Versión de PHP: 5.6.15

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Base de datos: `iothings`

--

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `wp\_valores`

--

CREATE TABLE `wp\_valores` (

`id` int(11) NOT NULL,

`hora` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

`origen` varchar(5000) NOT NULL,

`valor` varchar(5000) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Volcado de datos para la tabla `wp\_valores`

--

---OMITIDO----

--

-- Índices para tablas volcadas

--

--

-- Indices de la tabla `wp\_valores`

--

ALTER TABLE `wp\_valores`

ADD PRIMARY KEY (`id`);

-- AUTO\_INCREMENT de las tablas volcadas

--

--

-- AUTO\_INCREMENT de la tabla `wp\_valores`

--

ALTER TABLE `wp\_valores`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3448;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

### **Diagrama de uso lógico de la interfaz**



Script completo:

<?php

header( "refresh:1;url=plot.php" );

// iot.php

// Importamos la configuración

require("config.php");

// Leemos los valores que nos llegan por GET

// Esta es la instrucción para insertar los valores

//$query = "INSERT INTO wp\_valores(valor) VALUES('".$valor."')";

// Create connection

$conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$query = "SELECT \* FROM wp\_valores where ORIGEN='arduino' ORDER BY hora DESC LIMIT 120";

$result = $conn->query($query);

$labels = '';

$datas = '';

if ($result->num\_rows > 0) {

// output data of each row

while($row = $result->fetch\_assoc()) {

//echo "id: " . $row["id"]. " - hora: " . $row["hora"]. " - origen: " . $row["origen"]. " - valor: " . $row["valor"]. "<br>";

$labels = $labels.'"'.$row['hora'].'",';

$datas = $datas.'"'.$row['valor'].'",';

}

} else {

echo "0 results";

}

$query = "SELECT \* FROM wp\_valores where ORIGEN='rasp' ORDER BY hora DESC LIMIT 120";

$result = $conn->query($query);

$labelsR = '';

$datasR = '';

if ($result->num\_rows > 0) {

// output data of each row

while($row = $result->fetch\_assoc()) {

//echo "id: " . $row["id"]. " - hora: " . $row["hora"]. " - origen: " . $row["origen"]. " - valor: " . $row["valor"]. "<br>";

$labelsR = $labelsR.'"'.$row['hora'].'",';

$datasR = $datasR.'"'.$row['valor'].'",';

}

} else {

echo "0 results";

}

$conn->close();

?>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Lumino - Charts</title>

<link href="css/bootstrap.min.css" rel="stylesheet">

<link href="css/datepicker3.css" rel="stylesheet">

<link href="css/styles.css" rel="stylesheet">

<!--Icons-->

<script src="js/lumino.glyphs.js"></script>

<!--[if lt IE 9]>

<script src="js/html5shiv.js"></script>

<script src="js/respond.min.js"></script>

<![endif]-->

</head>

<body style="font-size:6px;">

<div class="col-sm-12">

<div class="row">

<div class="col-lg-12">

<div class="panel panel-default">

<div class="panel-heading">ARDUINO</div>

<div class="panel-body">

<div class="canvas-wrapper">

<canvas class="main-chart" id="line-chart" height="200" width="900"></canvas>

</div>

</div>

</div>

</div>

</div><!--/.row-->

<div class="row">

<div class="col-lg-12">

<div class="panel panel-default">

<div class="panel-heading">RASPBERRY</div>

<div class="panel-body">

<div class="canvas-wrapper">

<canvas class="main-chart" id="line-chart1" height="200" width="900"></canvas>

</div>

</div>

</div>

</div>

</div><!--/.row-->

</div> <!--/.main-->

<script src="js/jquery-1.11.1.min.js"></script>

<script src="js/bootstrap.min.js"></script>

<script src="js/chart.min.js"></script>

<script src="js/chart-data.js"></script>

<script src="js/easypiechart.js"></script>

<script src="js/easypiechart-data.js"></script>

<script src="js/bootstrap-datepicker.js"></script>

<script>

</script>

<script>

var randomScalingFactor = function(){ return Math.round(Math.random()\*1000)};

var lineChartData = {

labels : [<?php print $labels;?>],

datasets : [

{

label: "Arduino",

fillColor : "rgba(0,220,220,0.2)",

strokeColor : "rgba(0,220,220,1)",

pointColor : "rgba(0,220,220,1)",

pointStrokeColor : "#fff",

pointHighlightFill : "#fff",

pointHighlightStroke : "rgba(0,220,220,1)",

data : [<?php print $datas;?>]

}

]

}

var lineChartData1 = {

labels : [<?php print $labelsR;?>],

datasets : [

{

label: "RASPBERRY",

fillColor : "rgba(220,0,220,0.2)",

strokeColor : "rgba(220,0,220,1)",

pointColor : "rgba(220,0,220,1)",

pointStrokeColor : "#fff",

pointHighlightFill : "#fff",

pointHighlightStroke : "rgba(220,0,220,1)",

data : [<?php print $datasR;?>]

}

]

}

window.onload = function(){

var chart1 = document.getElementById("line-chart").getContext("2d");

window.myLine = new Chart(chart1).Line(lineChartData, {

responsive: true,

animation: false

});

var chart2 = document.getElementById("line-chart1").getContext("2d");

window.myLine2 = new Chart(chart2).Line(lineChartData1, {

responsive: true,

animation: false

});

};

!function ($) {

$(document).on("click","ul.nav li.parent > a > span.icon", function(){

$(this).find('em:first').toggleClass("glyphicon-minus");

});

$(".sidebar span.icon").find('em:first').addClass("glyphicon-plus");

}(window.jQuery);

$(window).on('resize', function () {

if ($(window).width() > 768) $('#sidebar-collapse').collapse('show')

})

$(window).on('resize', function () {

if ($(window).width() <= 767) $('#sidebar-collapse').collapse('hide')

})

</script>

</body>

</html>

## Documentación de pruebas

Casos de prueba establecidos y resultados de las pruebas y acciones de corrección. No es creíble que no hayan aparecido errores en los caso de prueba.

## Documentación de instalación

Descripción suficiente para que una persona que no ha participado en el proyecto pueda instalar el prototipo.

## Manual de usuario

Descripción suficiente para que una persona que no ha participado en el proyecto pueda utilizar toda la funcionalidad que ofrece el prototipo. Que debe coincidir con los requisitos funcionales incluidos en el apartado 2.

# Proyecto de implementación de un prototipo del sistema utilizando Raspberry

Se trata de incluir en este apartado la documentación del desarrollo del proyecto de implementación, utilizando la tecnología B, del sistema cuyos requisitos funcionales se enumeraron en el apartado 2.

## Documentación de diseño

Hay que incluir la descripción del diseño del prototipo, incluyendo diagramas, y el diseño de la interfaz de usuario.

### **Diagrama de uso lógico**

Se detalla las funciones y acciones que tiene cada componente.



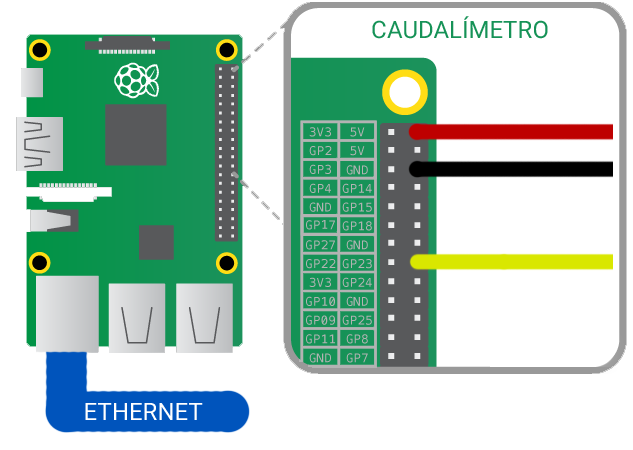
### **Diseño interfaz de usuario**



## Documentación de construcción

Hay que incluir una descripción de la construcción del prototipo, incluyendo algún extracto de código fuente. No es necesario todo el código. Sólo algún extracto para ver cómo se ha comentado.

### **Diagrama de conexiones Raspberry PI**



### **Diagrama de uso lógico del script de Raspberry PI**



Script completo:

pi@raspberrypi ~ $ cat sensorGET.py

#!/usr/bin/env python

import urllib

import urllib2

def enviar(val):

litros = val \* 60 / 7.5

values = {'origen' : 'rasp' ,

'valor' : litros }

data = urllib.urlencode(values)

url = 'http://192.168.10.1/iot.php?' + data

response = urllib2.urlopen(url).read()

print data

print response

print litros

print ' \t por minuto'

import RPi.GPIO as GPIO

import time, sys

FLOW\_SENSOR = 23

GPIO.setmode(GPIO.BCM)

GPIO.setup(FLOW\_SENSOR, GPIO.IN, pull\_up\_down = GPIO.PUD\_UP)

global count

count = 0

def countPulse(channel):

global count

count = count+1

GPIO.add\_event\_detect(FLOW\_SENSOR, GPIO.FALLING, callback=countPulse)

while True:

try:

time.sleep(1)

enviar(count)

count = 0

except KeyboardInterrupt:

print '\ncaught keyboard interrupt!, bye'

GPIO.cleanup()

sys.exit()

### **Diagrama de uso lógico de la API**



Script completo:

<?php

// iot.php

// Importamos la configuración

require("config.php");

// Leemos los valores que nos llegan por GET

$valor = mysqli\_real\_escape\_string($con, $\_GET['valor']);

$origen = mysqli\_real\_escape\_string($con, $\_GET['origen']);

// Esta es la instrucción para insertar los valores

//$query = "INSERT INTO wp\_valores(valor) VALUES('".$valor."')";

$query = "INSERT INTO wp\_valores(origen, valor) VALUES('".$origen."','".$valor."')";

// Ejecutamos la instrucción

mysqli\_query($con, $query);

mysqli\_close($con);

print "Recibido origen:".$origen." valor:".$valor;

?>

### **Diagrama de la base de datos**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre tabla: | wp\_valores |  |  |  |
| Campo: | Id | Hora | Origen | valor |
| Tipo: | Int autoincrementado | Timestamp - Fechado de los datos | Varchar | Varchar |
| Valores: | - | fecha | Arduino/Rasp | texto libre |

Script .sql :

-- phpMyAdmin SQL Dump

-- version 4.5.1

-- http://www.phpmyadmin.net

--

-- Servidor: 127.0.0.1

-- Tiempo de generación: 21-04-2016 a las 01:01:47

-- Versión del servidor: 10.1.9-MariaDB

-- Versión de PHP: 5.6.15

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Base de datos: `iothings`

--

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `wp\_valores`

--

CREATE TABLE `wp\_valores` (

`id` int(11) NOT NULL,

`hora` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

`origen` varchar(5000) NOT NULL,

`valor` varchar(5000) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Volcado de datos para la tabla `wp\_valores`

--

---OMITIDO----

--

-- Índices para tablas volcadas

--

--

-- Indices de la tabla `wp\_valores`

--

ALTER TABLE `wp\_valores`

ADD PRIMARY KEY (`id`);

--

-- AUTO\_INCREMENT de las tablas volcadas

--

--

-- AUTO\_INCREMENT de la tabla `wp\_valores`

--

ALTER TABLE `wp\_valores`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3448;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

### **Diagrama de uso lógico de la Interfaz de Usuario**



Script completo:

<?php

header( "refresh:1;url=plot.php" );

// iot.php

// Importamos la configuración

require("config.php");

// Leemos los valores que nos llegan por GET

// Esta es la instrucción para insertar los valores

//$query = "INSERT INTO wp\_valores(valor) VALUES('".$valor."')";

// Create connection

$conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$query = "SELECT \* FROM wp\_valores where ORIGEN='arduino' ORDER BY hora DESC LIMIT 120";

$result = $conn->query($query);

$labels = '';

$datas = '';

if ($result->num\_rows > 0) {

// output data of each row

while($row = $result->fetch\_assoc()) {

//echo "id: " . $row["id"]. " - hora: " . $row["hora"]. " - origen: " . $row["origen"]. " - valor: " . $row["valor"]. "<br>";

$labels = $labels.'"'.$row['hora'].'",';

$datas = $datas.'"'.$row['valor'].'",';

}

} else {

echo "0 results";

}

$query = "SELECT \* FROM wp\_valores where ORIGEN='rasp' ORDER BY hora DESC LIMIT 120";

$result = $conn->query($query);

$labelsR = '';

$datasR = '';

if ($result->num\_rows > 0) {

// output data of each row

while($row = $result->fetch\_assoc()) {

//echo "id: " . $row["id"]. " - hora: " . $row["hora"]. " - origen: " . $row["origen"]. " - valor: " . $row["valor"]. "<br>";

$labelsR = $labelsR.'"'.$row['hora'].'",';

$datasR = $datasR.'"'.$row['valor'].'",';

}

} else {

echo "0 results";

}

$conn->close();

?>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Lumino - Charts</title>

<link href="css/bootstrap.min.css" rel="stylesheet">

<link href="css/datepicker3.css" rel="stylesheet">

<link href="css/styles.css" rel="stylesheet">

<!--Icons-->

<script src="js/lumino.glyphs.js"></script>

<!--[if lt IE 9]>

<script src="js/html5shiv.js"></script>

<script src="js/respond.min.js"></script>

<![endif]-->

</head>

<body style="font-size:6px;">

<div class="col-sm-12">

<div class="row">

<div class="col-lg-12">

<div class="panel panel-default">

<div class="panel-heading">ARDUINO</div>

<div class="panel-body">

<div class="canvas-wrapper">

<canvas class="main-chart" id="line-chart" height="200" width="900"></canvas>

</div>

</div>

</div>

</div>

</div><!--/.row-->

<div class="row">

<div class="col-lg-12">

<div class="panel panel-default">

<div class="panel-heading">RASPBERRY</div>

<div class="panel-body">

<div class="canvas-wrapper">

<canvas class="main-chart" id="line-chart1" height="200" width="900"></canvas>

</div>

</div>

</div>

</div>

</div><!--/.row-->

</div> <!--/.main-->

<script src="js/jquery-1.11.1.min.js"></script>

<script src="js/bootstrap.min.js"></script>

<script src="js/chart.min.js"></script>

<script src="js/chart-data.js"></script>

<script src="js/easypiechart.js"></script>

<script src="js/easypiechart-data.js"></script>

<script src="js/bootstrap-datepicker.js"></script>

<script>

</script>

<script>

var randomScalingFactor = function(){ return Math.round(Math.random()\*1000)};

var lineChartData = {

labels : [<?php print $labels;?>],

datasets : [

{

label: "Arduino",

fillColor : "rgba(0,220,220,0.2)",

strokeColor : "rgba(0,220,220,1)",

pointColor : "rgba(0,220,220,1)",

pointStrokeColor : "#fff",

pointHighlightFill : "#fff",

pointHighlightStroke : "rgba(0,220,220,1)",

data : [<?php print $datas;?>]

}

]

}

var lineChartData1 = {

labels : [<?php print $labelsR;?>],

datasets : [

{

label: "RASPBERRY",

fillColor : "rgba(220,0,220,0.2)",

strokeColor : "rgba(220,0,220,1)",

pointColor : "rgba(220,0,220,1)",

pointStrokeColor : "#fff",

pointHighlightFill : "#fff",

pointHighlightStroke : "rgba(220,0,220,1)",

data : [<?php print $datasR;?>]

}

]

}

window.onload = function(){

var chart1 = document.getElementById("line-chart").getContext("2d");

window.myLine = new Chart(chart1).Line(lineChartData, {

responsive: true,

animation: false

});

var chart2 = document.getElementById("line-chart1").getContext("2d");

window.myLine2 = new Chart(chart2).Line(lineChartData1, {

responsive: true,

animation: false

});

};

!function ($) {

$(document).on("click","ul.nav li.parent > a > span.icon", function(){

$(this).find('em:first').toggleClass("glyphicon-minus");

});

$(".sidebar span.icon").find('em:first').addClass("glyphicon-plus");

}(window.jQuery);

$(window).on('resize', function () {

if ($(window).width() > 768) $('#sidebar-collapse').collapse('show')

})

$(window).on('resize', function () {

if ($(window).width() <= 767) $('#sidebar-collapse').collapse('hide')

})

</script>

</body>

</html>

## Documentación de pruebas

Casos de prueba establecidos y resultados de las pruebas y acciones de corrección. No es creíble que no hayan aparecido errores en los caso de prueba.

|  |  |  |  |
| --- | --- | --- | --- |
| Prueba y pasos | Objetivo | Pasos | Estado |
| Arranque del sistema operativo | Ver que todo arranca bien | Conectar todos los cables y SO en tarjeta de memoria. | Correcto |
| Conectividad ethernet | Ver que tenemos conexión con el servidor | Configurar Ios en servidor y Raspberry. Conectar cables. Hacer pings. Hacer ssh desde servidor a Raspberry | Correcto |
| Prueba del programa | Ver que podemos obtener datos | Conectar el sensor a los pines correspondientes.  Habilitar salida para depuración por consola.  Lanzar script. Mover el sensor para generar datos. Los datos aparecen por consola. | Correcto |
| Prueba de conexión con el servidor HTTP | Ver que se reciben datos | Conectar el sensor a los pines correspondientes.  Habilitar salida para depuración por consola.  Habilitar código para enviar peticiones GET. Lanzar script. Mover el sensor para generar datos. Los datos aparecen por consola. Los datos aparecen en la BBDD. | Problemas iniciales con los parámetros desde Raspberry, corregido.  Correcto |
| Probar que el programa arranca solo | Ver que el script se ejecuta de forma autónoma después del reinicio. | Añadir la línea correspondiente para el arranque.  Reiniciar. Ver que se reciben datos. | Hemos probado varias soluciones de arranque, la primera dependía de si el usuario hacía login (perfil) y la segunda iba con el arranque del SO (hemos elegido esta). Correcto |
| Probar que la conexión se recupera tras una desconexión del cable ethernet | Ver que se reciben datos | Con el script lanzado de forma automática quitar el cable. La base de datos ya no se actualiza. Reconectar el cable. La base de datos ya se vuelve a actualizar con normalidad sin intervención del usuario. | Correcto, el script nunca se detiene aunque pierda la conexión ethernet. |

## Documentación de instalación

Descripción suficiente para que una persona que no ha participado en el proyecto pueda instalar el prototipo.

Requisitos: Tarjeta microSD de 16gb.

### **Instalar el sistema operativo de Raspberry**

1) Descargar imagen del SO desde <https://www.raspberrypi.org/downloads/raspbian/>

RASPBIAN JESSIE LITE

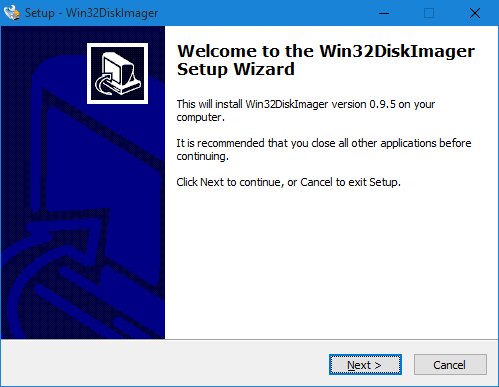
Minimal image based on Debian Jessie

Version:March 2016

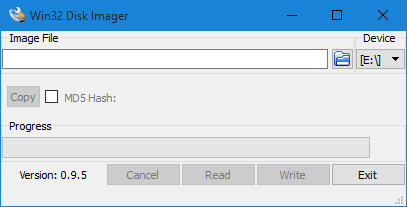
Release date:2016-03-18

Kernel version:4.1

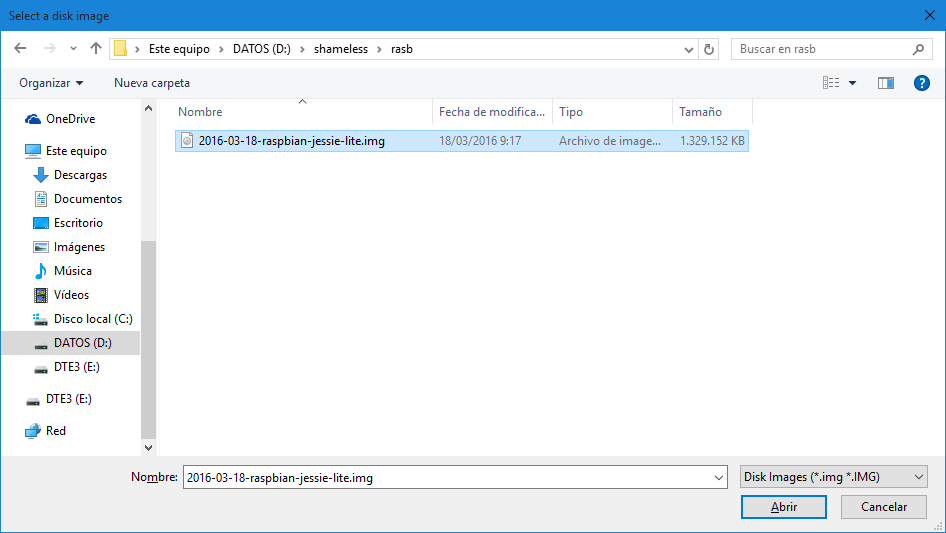
2) Instalar Win32DiskImager



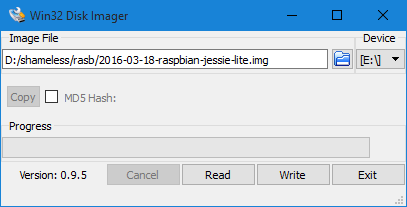
3) Ejecutar



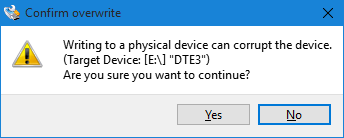
4) Seleccionar la imagen de Raspberian



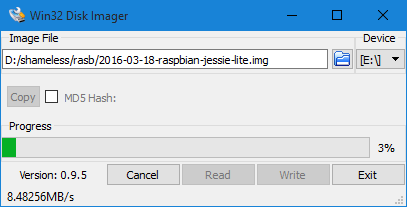
5) Seleccionar la opción WRITE para escribir la imagen en la tarjeta SD.

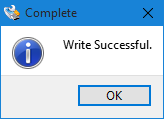


6) Confirmar borrado de la tarjeta.



7) Esperar a que el proceso acabe.





### **Configurar el sistema operativo de Raspberry**

Insertamos la tarjeta de memoria, teclado, ratón, cable hdmi y la alimentación. La placa arrancará automáticamente.

Hacemos login en el SO. Por defecto el usuario es **pi** y la contraseña **raspberry**.

#### **Configurar la interfaz de red**

Usando el editor vi configuramos la interfaz de la siguiente manera y guardamos los cambios mediante el comando **:wq!**.

pi@raspberrypi ~ $ sudo vi /etc/network/interfaces

auto lo

iface lo inet loopback

#iface eth0 inet dhcp

iface eth0 inet static

address 192.168.10.10

netmask 255.255.255.0

gateway 192.168.10.1

broadcast 255.255.255.255

allow-hotplug wlan0

iface wlan0 inet manual

wpa-roam /etc/wpa\_supplicant/wpa\_supplicant.conf

iface default inet dhcp

Reiniciamos para aplicar los cambios.

pi@raspberrypi ~ $ sudo reboot

Conectamos el cable de red y comprobamos que tenemos conectividad al servidor HTTP 192.168.10.1.

pi@raspberrypi ~ $ ping 192.168.10.1

#### **Copiar el script del sensor**

Tenemos 2 opciones, copiar el fichero sensorGET.py a la tarjeta y de ahí copiarlo al directorio del usuario (/home/pi) o aprovechar la conectividad de red y conectarnos por consola y crear el fichero.

pi@raspberrypi ~ $ cd /home/pi

pi@raspberrypi ~ $ vi sensorGET.py

#!/usr/bin/env python

import urllib

import urllib2

def enviar(val):

litros = val \* 60 / 7.5

values = {'origen' : 'rasp' ,

'valor' : litros }

data = urllib.urlencode(values)

url = 'http://192.168.10.1/iot.php?' + data

response = urllib2.urlopen(url).read()

print data

print response

print litros

print ' \t por minuto'

import RPi.GPIO as GPIO

import time, sys

FLOW\_SENSOR = 23

GPIO.setmode(GPIO.BCM)

GPIO.setup(FLOW\_SENSOR, GPIO.IN, pull\_up\_down = GPIO.PUD\_UP)

global count

count = 0

def countPulse(channel):

global count

count = count+1

GPIO.add\_event\_detect(FLOW\_SENSOR, GPIO.FALLING, callback=countPulse)

while True:

try:

time.sleep(1)

enviar(count)

count = 0

except KeyboardInterrupt:

print '\ncaught keyboard interrupt!, bye'

GPIO.cleanup()

sys.exit()

Asignarle permisos de ejecución.

pi@raspberrypi ~ $ chmod 777 sensorGET.py

#### **Hacer que el script del sensor arranque automáticamente con el SO**

Desde la consola editamos el fichero /etc/rc.local.

pi@raspberrypi ~ $ cat /etc/rc.local

#!/bin/sh -e

#

# rc.local

#

# This script is executed at the end of each multiuser runlevel.

# Make sure that the script will "exit 0" on success or any other

# value on error.

#

# In order to enable or disable this script just change the execution

# bits.

#

# By default this script does nothing.

# Print the IP address

\_IP=$(hostname -I) || true

if [ "$\_IP" ]; then

printf "My IP address is %s\n" "$\_IP"

fi

printf "##ARRANCAMOS SENSOR## "

sudo python /home/pi/sensorGET.py&

printf "##ARRANCAMOS SENSOR## "

exit 0

pi@raspberrypi ~ $

Reiniciamos para aplicar los cambios.

pi@raspberrypi ~ $ sudo reboot

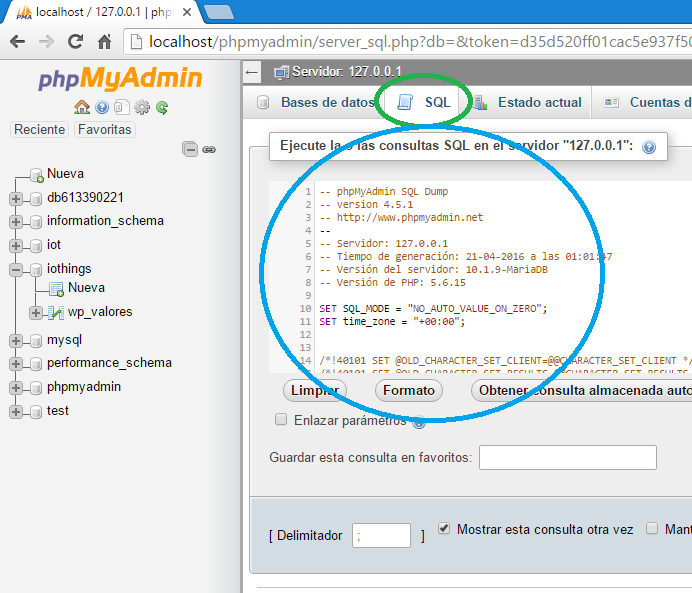
### **Configurar servidor web**

1) Arrancamos Apache y MySQL desde el panel de control de Xampp.



2) Seleccionamos Mysql Admin para que nos lleve a la página de gestión de la base de datos.

3) Seleccionamos la pestaña SQL e importamos el backup de la base de datos.



4) Configuramos los parámetros del servidor en el fichero C:/xampp/htdocs/config.php

<?php

// config.php

// Credenciales

$dbhost = "localhost";

$dbuser = "iothings";

$dbpass = "iothings";

$dbname = "iothings";

// Conexión con la base de datos

$con = mysqli\_connect($dbhost, $dbuser, $dbpass, $dbname);

?>

## Manual de usuario

Descripción suficiente para que una persona que no ha participado en el proyecto pueda utilizar toda la funcionalidad que ofrece el prototipo. Que debe coincidir con los requisitos funcionales incluidos en el apartado 2.

1. Conectar cables y sensor
2. Arrancar servidor Xampp



3) Acceder a la dirección <http://localhost/graficos/plot.php>

Las gráficas se refrescarán automáticamente según se van recibiendo los datos.



# Comparación de las dos implementaciones

Se trata de dar valores a los criterios de comparación definidos en el apartado 3 sobre la implementación de cada uno de los prototipos.

## Evaluación de los criterios en la implementación usando la tecnología A

Debe incluir al menos una tabla con la siguiente estructura.

| **CRITERIO** | **EVALUACIÓN** |
| --- | --- |
| Criterio 1 |  |
| Criterio 2 |  |
| … |  |
| Criterio N |  |

Y algunos comentarios aclaratorios sobre aquellos criterios cuyo valor indicado en la tabla no sea suficiente para entenderlo.

## Evaluación de los criterios en la implementación usando la tecnología B

# Comparación de la implementación de las tecnologías

Debe incluir al menos una tabla resumen, en sección de página horizontal, cruzando los criterios y los valores de cada tecnología. Con una columna de comentarios sobre la comparación

| **CRITERIOS** | **TECNOLOGÍA A** | **TECNOLOGÍA B** | **COMENTARIOS** |
| --- | --- | --- | --- |
| 1 |  |  |  |
| 2 |  |  |  |
| … |  |  |  |
| N |  |  |  |

# Conclusiones

A partir de la información incluida en el apartado 7 y de la experiencia al realizar el trabajo, el grupo debe estar en condiciones de manifestar su opinión sobre la implementación del sistema utilizando ambas tecnologías, y debe plasmarla en este apartado, indicando las ventajas e inconvenientes más relevantes de utilizar una u otra tecnología para implementar el sistema.