Sistema de riego automatizado

08/06/2018

Pablo Benedicto Castellano

Trabajo de fin de grado

D.A.W.

I.E.S. La Vereda

Índice

Manual de usuario

Descripción del sistema

El sistema de regadío ha sido pensado para amoldarse a un terreno áspero y muy variable, a un entorno cambiante y poco adverso, y por supuesto a prueba de falta de conocimiento para la informática. Por tanto la idea era crear un sistema ya preensamblado y listo para usar, en el cual el cliente simplemente tuviera que montarlo y encenderlo para que funcionase. Por otro lado, Arduino funciona por sí solo y no es necesario tener la aplicación web siempre encendida, esta última está diseñada para mostrar datos con el menor margen de tiempo y poder acceder a registros, de manera que no haga falta entrar en la misma base de datos y descifrar su estructura. El sistema está diseñado para soportar hasta un máximo de cuatro parcelas simultáneas recogiendo la humedad de estas y abriendo y cerrando sus respectivas válvulas a tiempo real, cada una con su propio sistema de seguridad diseñado desde cero para que sea lo más duradero posible y para que los distintos sensores acoplados a este no sufran y consuma la menor energía posible.

Si quieren descubrir como se ha diseñado este sistema de seguridad, o como se ha conseguido que se muestre por pantalla casi a tiempo real la información de los sensores para que sea una página web completamente operativa y adaptable a las antiguas tecnologías, no duden en seguir leyendo y aprendan de la experiencia.

Uso y requisitos mínimos

Utilidad y funcionalidad

Lista de las funcionalidades que cubre el sistema:

- Regadío automatizado: El sistema central es un dispositivo robótico llamado Arduino que automatiza el sistema de riego de manera completamente autónoma y eficiente.
- Control de temperatura y humedad ambiente: Distintos sensores para parametrizar la temperatura y humedad ambientales del ámbito donde se encuentre la instalación.
- Algoritmo de predicción de tiempo interno: La aplicación web incorpora un sistema de predicción de tiempo generado a partir de los distintos sensores ambientales del ámbito de la instalación.

- Base de datos de registros: El servidor incorpora una base de datos que de manera automática guarda información de los distintos registros que genera Arduino.
- Página web funcional para la consulta de información: La aplicación web es completamente funcional y dinámica, y está preparada para leer los registros de la base de datos generados por Arduino y visualizarlos de manera simple y explicada. También tiene descripciones incorporadas para la gente que menos domina la materia y más guia necesita.

Requisitos mínimos (Arduino)

- ❏ Arduino Uno/similar
- ❏ Cable usb a/b
- ❏ Cable de alimentación usb 2V
- ❏ Dos leds verde y rojo por cada parcela (opcional)
- ❏ Un sensor de humedad y una válvula por cada parcela
- ❏ Cables macho y hembra de puente
- ❏ Placa de prototipado

Requisitos mínimos (aplicación)

- ❏ Raspberry o similar (cualquier ordenador con linux)
- ❏ Conexión a internet para la instalación desde cero
- ❏ Switch o router sin conexión a internet
- ❏ Software:
    - ❏ PHP v5 o superior
    - ❏ Apache v2
    - ❏ MySQL v3 o superior

Manual técnico

Modelo de datos

El sistema ha sido diseñado para que Arduino trabaje de manera autónoma y sin necesidad de conexión a un serial bus. Por lo tanto en la parte de Arduino encontrarán que no se tiene que modificar ningún código con la excepción de las válvulas que se quieran utilizar. Pero esto lo explicaré con más detalle luego.

Por el lado de Arduino tenemos el código estructurado de la siguiente manera:

1) Actualización de valores:

```
humedad=dht.readHumidity();
temperatura=dht.readTemperature();

for(int v=0; v<numeroValvulas; v++){
  cache=analogRead(v);
  if(cache!=valvulaMaster[v]){
    valvulaMaster[v]=cache;
    t=(valvulaMaster[v]*100)/1023;
    t-=100;
    t*=(-1);
    porCien=t;
  }
  valvulas[v]=porCien;

  if(valvulas[v]<5){
    resultado="Seco";
  }else if(valvulas[v]>=5 && valvulas[v]<20){
    resultado="Poco húmedo";
  }else if(valvulas[v]>=20 && valvulas[v]<60){
    resultado="Húmedo";
  }else if(valvulas[v]>=60 && valvulas[v]<80){
    resultado="Muy húmedo";
  }else if(valvulas[v]>=80 && valvulas[v]<90){
    resultado="Mojado";
  }else{
    resultado="Inundado";
  }
```

a) Aquí se encuentra la obtención y procesamiento con el caché de los datos de los distintos sensores.

2) Activación del sistema de seguridad:

```
if(valvulaResultado[v]!=resultado && seguridadValvula[v]==0){
 digitalWrite((v+7), HIGH);
 valvulaResultado[v]=resultado;

     if(valvulas[v]<obertura[v]){
       digitalWrite((v+2), HIGH);
       seguridadValvula[v]=millis();
     }else{
       digitalWrite((v+2), LOW);
       seguridadValvula[v]=millis();
     }
}else if((millis()-seguridadValvula[v])>=tiempoSeguridad){
  digitalWrite((v+7), LOW);
  seguridadValvula[v]=0;
}
```

a) Este es el sistema de seguridad que se activa si es necesario para que no sufran o se malgaste tanto la energía como el agua. También se activa la parte visual para los leds de aviso.

3) Expulsión de datos:

```
datos="";
if(Serial.available()>0){
  while(Serial.available()>0){
    datos=Serial.read();
  }

  //El 49 significa que el carácter que se pasa es igual a uno, ya que 49 en ascii es 1
  if(datos==49){
    String datosArray="[";
    for(int i=0; i<numeroValvulas; i++){
      if(i!=0){
        datosArray=datosArray+",";
      }
      datosArray=datosArray+"["+String(valvulas[i])+","+obertura[i]+"]";
    }
    datosArray=datosArray+"]";
    Serial.println("-*[");
    Serial.println(datosArray);
    Serial.println(",");
    Serial.println(humedad);
    Serial.println(",");
    Serial.println(temperatura);
    Serial.println("]^-");
  }else{
    Serial.println("-*");
    Serial.write("NULL");
    Serial.println("^-");
  }
```

a) Esta parte únicamente se activa si Arduino recibe una clave de datos. Si se activa devuelve por el puerto serial los datos de los sensores en ese mismo momento.

Por otro lado tenemos todo el sistema del servidor y Linux, el cuál trabaja de manera constante para la persistencia de datos y el visualizado de estos. Así pues, la base de datos tienes dos tablas. 'Registros' funciona como una salva de datos diária de todos los datos del día, y por otro lado 'registro_regadio_actual' guarda todos los datos del día para al final de este volcarlos en la anterior.

	Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
☐	registros	Examinar Estructura Buscar Insertar Vaciar Eliminar	18	InnoDB	utf8_spanish_ci	16 KB	-
☐	registro_regadio_actual	Examinar Estructura Buscar Insertar Vaciar Eliminar	48	InnoDB	utf8_spanish_ci	16 KB	-

Por último. el servidor corre en Apache y ejecuta php. Este está distribuido por clases y por unos poco scripts singulares que se ejecutan cuando se requieren, a parte de de la web, por supuesto.

```
        $return=mysqli_fetch_assoc($consulta)['humedad'];
    }else{
        //No hay datos guardados en la tabla
        $consulta=mysqli_query($enlace, 'SELECT MAX(fecha) FROM registros');
        $max_fecha=mysqli_fetch_assoc($consulta)['MAX(fecha)'];

        $consulta=mysqli_query($enlace, 'SELECT humedad_media FROM registros WHERE fec
        $return=mysqli_fetch_assoc($consulta)['humedad_media'];
    }

    return $return;
}

public static function tiempo($temperatura, $humedad){
    $return=array();
    $date=getdate();

    if(intval($date['hours'])>7 and intval($date['hours'])<20){
        $situacion='-d';
    }else if(intval($date['hours'])==7){
        if(intval($date['minutes'])>=30){
            $situacion='-d';
        }else{
            $situacion='-n';
        }
    }else if(intval($date['hours'])==20){
        if(intval($date['minutes'])<=30){
            $situacion='-d';
        }else{
            $situacion='-n';
        }
    }else{
```

Funcionamiento

El funcionamiento básico de la aplicación es el de automatizar mediante Arduino un sistema autónomo de regadío por goteo. Para esto se hace servir c++ para programarlo. Como complemento importante a este sistema se ha creado una aplicación web que funciona en antiguos ordenadores y que representa de forma gráfica los distintos datos recabados, así como el recaudo de estos.

Ambas dos pueden funcionar por separada y, de nuevo, ambas dos se complementan creando un sistema de visualización de datos y regadío automático creado desde cero.

Tecnologías utilizadas

- Arduino
  - C++
- Servidor
  - Apache
  - PhP
  - MySQL
  - Ruby

Despliegue del sistema

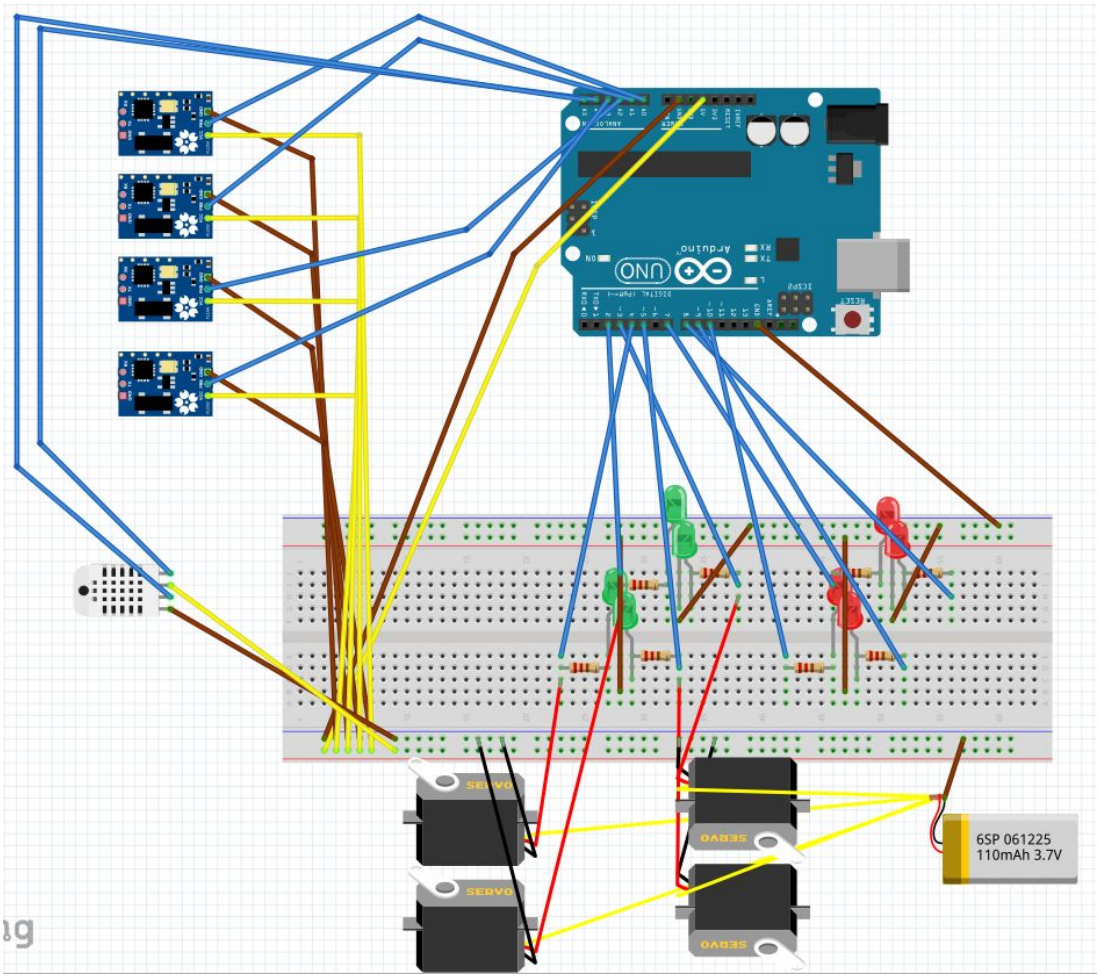
Todo el código del sistema está subido con su control de versiones al repositorio de GitHub https://github.com/pbenedictoc/TFG. También hay una pequeña explicación de como instalarlo aparte de este documento anexado al repositorio.

Despliegue de Arduino

Instalación

Componentes

Primero tendremos que montar todo el circuito eléctrico de arduino en la placa de prototipado. Para guiar en esta instalación he hecho un esquema del prototipo para que no haya problemas en las conexiones.



Los servos representa a las válvulas y la pila a la entrada de energía. Después de hacer todas las conexiones simplemente hay que darle alimentación a Arduino para que todo el sistema se ponga a funcionar.

Si Arduino no llevase el código precompilado, simplemente hay que descargarlo desde mi repositorio e introducirlo en el IDE de Arduino, seleccionar la placa Arduinno Uno y el puerto serie al que está conectado y por último copiar el código, precompilarlo y subirlo.

Montaje

Como dispositivo Arduino está listo para usarse, pero hay que colocarlo en un lugar fijo y con acceso al servidor. Este sitio tiene que estar resguardado de las condiciones climatológicas y con acceso a un punto de electricidad constante (enchufe). Así pues simplemente habrá que conectar Arduino por su puerto serie, mediante el cable USB, a la máquina del servidor. Después habrá que tirar cable para los distintos sensores de humedad y temperatura, así también para las válvulas y colocarlo todo donde se desee.

Para el sistema de leds no hay una única solución, se pueden poner donde se deseen ya que simplemente están para informar del estado a tiempo real de Arduino y todas sus funcionalidad, así que se pueden instalar en un panel o dejarlos en la misma placa de prototipado. De todas las formas posibles, es suya la elección.

Para finalizar hay predefinidas unas opciones que pide el cliente, en las cuales predispone de un límite de humedad para cada parcela, así que estos datos se introducen en el código del Arduino. Para cambiarlos simplemente hay que modificar el número que se quiera con un límite de 1-100.

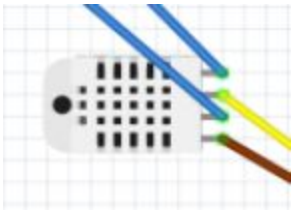
```
//En el array obertura, guardamos los criterios de obertura de las válvulas, en orden y tanto %.
int obertura[]={25, 30, 40, 50};
```

Con esto conseguimos cambiar los criterios de obertura de cada válvula, así si queremos que una parcela abra sus válvulas cuando hay menos del 25% de humedad, incluimos ese 25 en el código y estará operativo en cuanto compilemos.

Ampliaciones

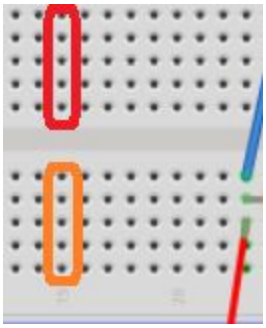
Arduino es un dispositivo muy versátil, así que casi todas las ampliaciones que se puedan imaginar son viables. Para este caso he previsto unas cuantas y el código está preparado para soportarlas. Así pues las enumero:

1. De parcelas: como se puede ver, este sistema está preparado para recoger la humedad de un terreno y abrir o cerrar su consiguiente válvula si es necesario. En este caso se ven cuatro válvulas con sus cuatro sensores conectados a los pines analógicos. Como programador he pensado que una mejora o ampliación común sería el de poner más sensores y válvulas, así que el programa está diseñado para que simplemente cambiando un número en el código esto se pueda hacer. Así que veamos como:
    a. Lo primero y más importante que hay que saber es que para hacer esto hay que sacrificar el módulo de recepción de humedad y temperatura del



ambiente. Una vez extraído y desconectado de manera correcta este dispositivo quedarían los pines analógicos A4 y A5 disponibles.

b. En estos dos pines podremos conectar otros dos sensores de humedad del terreno, como están conectados los anteriores. Una vez que se hayan conectado de manera correcta estos dos conectores (tienen que tener una **ganancia(cable amarillo)** y una **toma de tierra(cable marrón)**) daríamos por terminadas las conexiones de los sensores.
c. Para que estos muevan y abran sus respectivas válvulas, estas hay que conectarlas de la siguiente forma: de los pines digitales 11 y 12 se extraerá un cable (azul), estos se conectarán a una línea de la placa de prototipado



diferente de todas las demás y de sí mismos. De esta línea sacaremos un cable que irá a la toma de datos de la válvula, lo que en el esquema se representa con un cable **rojo**.
d. Por último tendremos que hacer la parte más complicada, que es modificar ese número en el código de Arduino para que en vez de cuatro sectores a analizar tengamos 6. Se hace de la siguiente forma: al principio del código

```
//Captamos las valvulas disponibles a mano, no hay otra forma, de una a cuatro. En este caso solo cojo una.
int numeroValvulas=4;
int valvulas[4];
float valvulaMaster[4];
String valvulaResultado[4];
unsigned long seguridadValvula[4];
//En el array obertura, guardamos los criterios de obertura de las válvulas, en orden y tanto %.
int obertura[]={25, 30, 40, 50};
```

tenemos las variables inicializadas de todo el sistema, hay una parte en la cuál llamamos a una variable con el nombre '**numeroValvulas**=4;'. Desde esta variable hasta la variable '**seguridadValvula**[4];' tendremos que cambiar todos los 4 => 6 para que detecte 6 sectores en vez de 4. Por último tendremos que dar los criterios de obertura de flujo para que se abran las válvulas, así que seguido, justo abajo, de las anteriores variables tendremos la variable '**obertura**[]={x, x, ,x, x};', donde x=a cualquier rango desde 0 a 100.

Según cuantas válvulas hemos añadido tendremos que poner x valores. En este caso como añado 2 válvulas añado dos valores. Quedaría de la siguiente forma: '**obertura**[]={x, x, ,x, x, y, z};', donde 'y' y 'z' son dos nuevos valores elegidos por mi y que tienen un rango de 0 a 100. Por si no ha quedado muy claro, aquí abajo tienen un ejemplo visual de cómo quedaría el cambio:

```
//Captamos las valvulas disponibles a mano, no hay otra forma, de una a cuatro. En este caso solo cojo una.
int numeroValvulas=6;
int valvulas[6];
float valvulaMaster[6];
String valvulaResultado[6];
unsigned long seguridadValvula[6];
//En el array obertura, guardamos los criterios de obertura de las válvulas, en orden y tanto %.
int obertura[]={25, 30, 40, 50, 20, 53};
```

e. De esta forma tendríamos dos válvulas más completamente operativas con el mismo Arduino.
f. Opcional: si queremos también tener los leds asociados a estas nuevas válvulas simplemente tenemos que conectar los leds verdes, ya que los rojos son del sistema seguridad y no habría hueco en este Arduino para ponerlos, con el pin más largo a una resistencia y que esta esté conectada al la línea de la placa del prototipado la cual hemos conectado el pin digital asociado a esa válvula.
g. Comentario: Si este programa el cual he desarrollado se pusiese en un Arduino con posibilidad de más pines, simplemente habría que seguir el orden de conexiones de pines, tanto digitales como analógicos. Así pues por el lado de los digitales siempre se empezaría en el 2, y para el sistema de seguridad se empezaría en el múltiplo par más grande que daría el recuento total de pines digitales dividido entre 2. Y por el lado de los analógicos, siempre se empezaría por el A0.
h. Advertencia: No hay ningún problema en que se quede en el código un número de pines ni mayor ni inferior a los que realmente hay, el programa está diseñado para capturar los errores. Pero si que hay que tener cuidado con conectar todo lo anterior a un punto de tierra, no hay que olvidarse nunca, ya que cada salida de corriente y circuito cerrado necesita una toma de tierra. En el caso de no haber ninguna se podría estropear Arduino por culpa de un cierre y podría quedar inservible.

2. De módulos: esta ampliación es mucho más sencilla de implantar que la anterior y consiste en aumentar los sensores los cuales recogen datos de una parcela. Bien sea por ampliar el terreno de esta que se quiera controlar o por tener una medida más acotada a tiempo real de lo que sucede. Sea cual sea la razón este sistema está preparado para ello.

a. Si por ejemplo queremos ampliar o duplicar los sensores de humedad de una parcela, basta con conectar en una misma línea de la placa de prototipado (emulando un ladrón de cables) los dos módulos de humedad. Los datos que lleguen a Arduino serán procesados teniendo en cuenta la duplicidad de estos y darán un resultado funcional y mejorado.
b. Por otro lado, si queremos ampliar las válvulas de una misma parcela, tendremos que conectar en una misma línea de la placa de prototipado un nuevo cable que lleve a esa nueva válvula.
c. Advertencia: en ambos casos, solo se está duplicando el cable que transporta datos, así que se deberá suplir el cable de alimentación y el de toma de tierra para esos nuevo componentes. En este caso es fácil, ya que Arduino tiene suficiente potencia con esos 5v. Por otro lado la toma de tierra también es muy importante, ya que como antes he explicado, la falta de esta dañará el sistema y Arduino quedará inutilizado.

Mantenimiento

El mantenimiento de Arduino es casi inexistente. Por otro lado hay que cuidarlo en el sentido de resguardarlo de las acciones climatológicas y del agua. Simplemente con meterlo en una caja y cerrarla servirá.

Donde sí que hay que hacer hincapié es en los módulos que están recogiendo datos, ya que estos si que sufren los diferentes factores de la naturaleza y son los 5 sentidos de Arduino y de nuestro sistema, así que hay que cuidarlos y llevar una asidua vigilancia de estos. Simplemente con verificar que siguen en buen estado de manera visual y que ningún cable está roto u oxidado bastará para llevar ese control.

Información visible

El sistema creado muestra varios leds, unos de color verde y otros de color rojo. Los de color verde son las válvulas y su posición actual, si están encendidos las válvulas están abiertas, si no están cerradas. Por otro lado los leds de color rojo son el sistema de seguridad de cada válvula, si está encendida es que el sistema de seguridad está activo, en cambio si no está encendida el sistema de seguridad estará apagado y a la espera de ser activado.

De parte de Arduino no hay nada más visible, si se profundiza un poco más en el dispositivo veremos un total de 3 pequeños leds, dos naranjas juntos y uno verde un poco más grande separado de estos. El más grande que muestra si el dispositivo está encendido y funciona todo correctamente. Por otro lado los dos naranjas que están juntos son los leds de I/O, muestran si hay transferencia de datos por el puerto serial en curso, uno es de salida y el otro de entrada.

Por último existe un pequeño led naranja pequeño al lado de los pines digitales que se enciende si Arduino está suministrando energía a cualquier componente de manera contínua.

Estos último son importantes de cara a evaluar el estado de Arduino, pero no son necesarios estar revisandolos todo el rato, ya que si algo diese error o fallara existe el led grande verder anteriormente mentado el cual se apagaría si no funcionase bien el sistema o hubiera algún fallo, ya que el programa hecho está diseñado para que en caso de error irreparable este se apague.

Despliegue de aplicación web

Antes de empezar con la explicación de la aplicación alojada en el servidor me gustaría decir que está diseñada para que complemente a Arduino, ya que sin este la página web quedaría inservible. Del mismo modo y por otro lado, Arduino es completamente independiente de esta aplicación. Me explico un poco más en detalle:

La aplicación está diseñada para extraer y leer los datos de Arduino, visualizar y analizar para después filtrar y guardar estos de manera perpetua. No está diseñada para hacer funcionar Arduino ni para que este se pueda apoyar en ella, ya que el cliente no lo quería así.

También mencionar que la aplicación es todo el servidor, no solo el código php o html que se ve, ya que esta lleva un control de acceso desde Arduino para la permanencia de datos o la programación suficiente para que todo esto lo haga automático.

Por último mencionar que la parte de la web está diseñada para que funcione en dispositivos viejos y con poca velocidad, así que he tenido que reducir el uso del multimedia así como el uso amplio de JS, el cual he utilizado muy poco y me hubiera gustado utilizar mucho más.

Sin más preámbulos, comencemos con la instalación.

Instalación

Equipo necesario anteriormente mencionado:

- Pc/dispositivo Linux
- Router (opcional)

Para preparar la instalación verificamos que el dispositivo donde queramos introducir nuestro servidor sea Linux. Una vez verificado procedemos a conectar el dispositivo a la red local. Si tenemos una ya predefinida perfecto, una vez hecha la instalación, con el comando ‘ifconfig’, descubriremos cuál es la ipv4 del dispositivo y nos la guardaremos para poder entrar en nuestro servidor.

También existe la opción de asignar estáticamente en nuestra red interna la ip del dispositivo. Este ajuste lo he hecho en la solución que le he generado al cliente. Lo he hecho de la siguiente manera:

1. Descubrir la mac del dispositivo
   1.1. En el caso de linux es bastante sencillo descubrir la mac de la tarjeta wifi del dispositivo, simplemente hay que utilizar el comando antes mencionado 'ifconfig' y nos saldrá algo como esto entre toda la demás información:

   1.2. Después accedemos al router que genera las ips de manera interna. La ip por defecto para acceder a estos es 192.168.1.1.
2. Acceder al router y asignar yp
   2.1. En el caso del cliente, he tenido que acoplar un router antiguo a la red interna que tiene con los otro dispositivos conectados por cables. Así que desde su switch, con un cable de conexión de red, he conectado el router antiguo con WiFi para que haga de repetidor de este. Para hacer esto hay que quitar antes el servicio DHCP de este router, ya que sino el también asignaría ips dentro de la misma red y generaría un conflicto con la actual red. Así que para hacer esto, accede al router que se vaya a conectar mediante WiFi y desactiva el servicio DHCP:

   2.2. Después asignamos una ip estática para esa mac en este router. En este caso

   he asignado dos ip al servidor. Una es por cable y la otra por WiFi.
   2.3. Con todo esto, conseguimos que de manera interna podamos acceder siempre a nuestro servidor desde la misma ip estática.

Para empezar a instalar el software siguiente necesitaremos una conexión estable a internet, ejecutar el comando 'update' y luego 'upgrade' para asegurarnos de que accedemos a las últimas actualizaciones de las instalaciones.

Apache

Para que el servidor se ejecute necesitamos Apache, así que desde una terminal de nuestros dispositivo servidor ejecutamos el comando con permisos de su 'apt-get install apache2'. Una vez instalado todos los paquetes, todo el código del servidor estará alojado en la carpeta '/var/www/html/', para cualquier modificación de este únicamente habrá que ir a la carpeta en cuestión y modificar el archivo.

- Posible problemas: es posible que por una mala decisión en la instalación o por la modificación de los archivos de configuración se apague el servidor o no funcione bien. Para estos casos tenemos el comando 'service apache2' con todas estas opciones:

```
Usage: apache2 {start|stop|graceful-stop|restart|reload|force-reload}
```

Para asegurarnos de que todo esté en funcionamiento cargamos la página 192.168.1.1(**6** si se está por cable y **7** si se está por WiFi)1. Si carga no hay ningún problema y todo está funcionando correctamente. Sino una posible solución ya que solo hemos instalado Apache es purgar la actual instalación con el comando 'purge apache2' y volver a instalar Apache.

Como en este caso es una red interna, no hace falta alterar el firewall para el tráfico de datos HTTP y HTTPS. Si se tuviese que hacer, se tendría que entrar y modificar la regla de Apache en el firewall con el siguiente comando 'ufw allow in "Apache Full"'.

MySQL

Para la persistencia y estructura de todos los registros se va a utilizar MySQL. Lo primero que se tiene hacer es instalar este software con el comando 'apt-get install mysql-server-php5 mysql'. De esta forma también instalamos unos cuantos paquetes que ayudan a la interconexión con php.

En la solución que he creado para el cliente el dispositivo que contiene al servidor es una RaspBerry, así que estos comandos se modifican un poco para que el servidor, la base de datos y php funcionen bien. Para instalar MySQL utilicé el comando 'apt-get install mariadb-server mariadb-client'. Y para que MySQL se comunicará con PhP utilicé el comando 'apt-get install php7-mysqlnd'.

En ambos casos, para asegurar que la instalación de la base de datos se hace correctamente ejecutamos el comando 'mysql_secure_installation'. Así podremos asignar un usuario y contraseña a la base de datos y al usuario de esta. Con este comando se hace de manera interactiva, así que solo hay que seguir las instrucciones.

PhP

Para que todos los scripts funcionen y la página web se haga de forma dinámica, necesitamos PhP. Para instalarlo hay que introducir el siguiente comando el la máquina en cuestión 'apt-get install php7'.

En el caso de la solución creada para el cliente he utilizado el comando anterior más este 'apt-get install libapache2-mod-php7'. Así consigo la uni´n de PhP, Apache y MySQL en la RaspBerry.

Gracias a PhP, desde cron podremos ejecutar scripts hechos en PhP.

PhpMyAdmin

Con este software podremos visualizar la base de datos, el cliente no tiene por qué usarlo, pero nunca viene mal tenerlo mientras se desarrolla la aplicación y cuando se quiere solucionar algún error una vez instalado el sistema. Para instalar este software, en el caso del cliente en un sistema Raspbian, he utilizado el siguiente comando 'apt-get install phpmyadmin', y luego he tenido que crear un usuario en la base de datos para que phpmyadmin pueda acceder a esta. Para crear dicho usuario he utilizado el siguiente comando 'sudo mysql -u root' para entrar en la base de datos como root, y luego 'CREATE USER 'phpmyadmin'@'localhost' IDENTIFIED BY 'phpmyadmin';

GRANT ALL PRIVILEGES ON *.* TO 'phpmyadmin'@'localhost' WITH GRANT OPTION;

FLUSH PRIVILEGES;' este otro para crear el usuario con los permisos. De esta forma conseguiremos tener completamente operativo phpmyadmin en RaspBerry.

Cron

Este software no hay que instalarlo, pero si que hay que programarlo para que realice las acciones que queramos. Para la solución creada al cliente esto ya lo tiene hecho, pero aquí voy a explicarlo para que se vea como lo he programado.

1. Ingresamos con el comando 'crontab -e' en cron.
2. Una vez en el archivo, nos vamos al final de este y se introducen los comandos y la frecuencia con los que se quiera programar. En el caso de la solución del cliente he añadido dos comandos:
    a. */5 * * * * php /var/www/html/guardar_registro.php
    b. 00 00 * * * php /var/www/html/generar_dia.php

El comando **a** se ejecuta cada cinco minutos y guarda todos los datos actuales de Arduino. Temperatur y humedades. Por otro lado el comando **b** se ejecuta solo una vez al día y recaba todos los registros que se han generado durante todo el día y los guarda de manera comprimida.

Mantenimiento

El servidor no necesita un mantenimiento muy asiduo, ya que no está expuesto en la red y simplemente sirve para mostrar los datos de los registros. Por otro lado, conforme está diseñado el proyecto, se ha buscado en que no pueda dar ningún tipo de error ni de basura, es decir, ni la base de datos ni el propio bloque de programación diseñado en php dejan ningún tipo de caché en el sistema. Así que en principio no deberían de dejar ningún cabo suelto independientemente de las veces que se ejecuten.

Además de esto, el archivo log está activo, y todos los posibles errores no capturados quedarán reflejados en este.

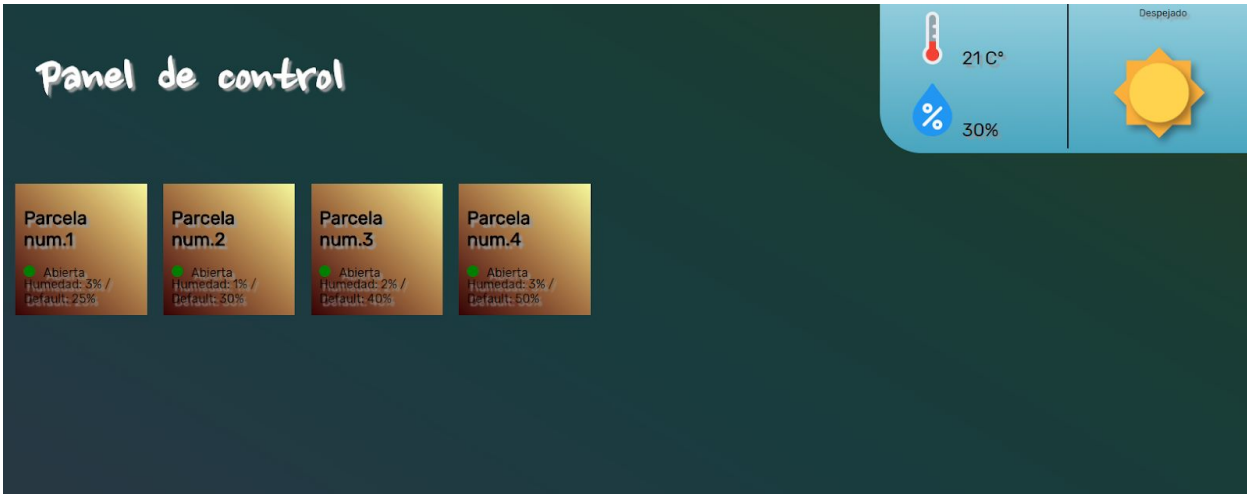
Página web

La página web es bastante simple, ya que el cliente lo pedía de esta forma indirectamente, me explico. El cliente es una persona que no tiene muchas tecnologías donde se va a hacer la instalación, así que el reto era diseñar una página web que se pudiese ver desde el móvil, una pda antigua e internet explorer 6. Como uno se podría imaginar no podía ser ni sobrecargada, ni dinámica en el sentido de javascript o css, y por supuesto nada de multimedia, ya que en muchos casos dejaría la página colgada en ese internet explorer o pda. Así que desde un primer momento diseñe con ayuda de Antonio una página web simple pero interactiva, que pudiese mostrar toda la información posible y que funcionase en todos estos dispositivos. Utilicé un poco de JavaScript para cubrir la asignatura de Entorno Cliente y conseguí hacer una página web liviana y simple, poco sobrecargada y preparada, por si en algún momento hubiese internet, para que ejecutara el poco JavaScript que había.

Haciendo distintas pruebas a posteriori restructure el JavaScript para que este no diese errores pero que si que estuviese, y que si no se podía utilizar, que al menos quedase latente para que en otro momento, desde otro explorador o equipo si que se ejecutará.

También mencionar que la idea era poner Ajax para la actualización de datos, pero al ser JavaScript, ni la pda ni el internet explorer lo ejecutaban, así que tuve que tomar la decisión de no ponerlo.

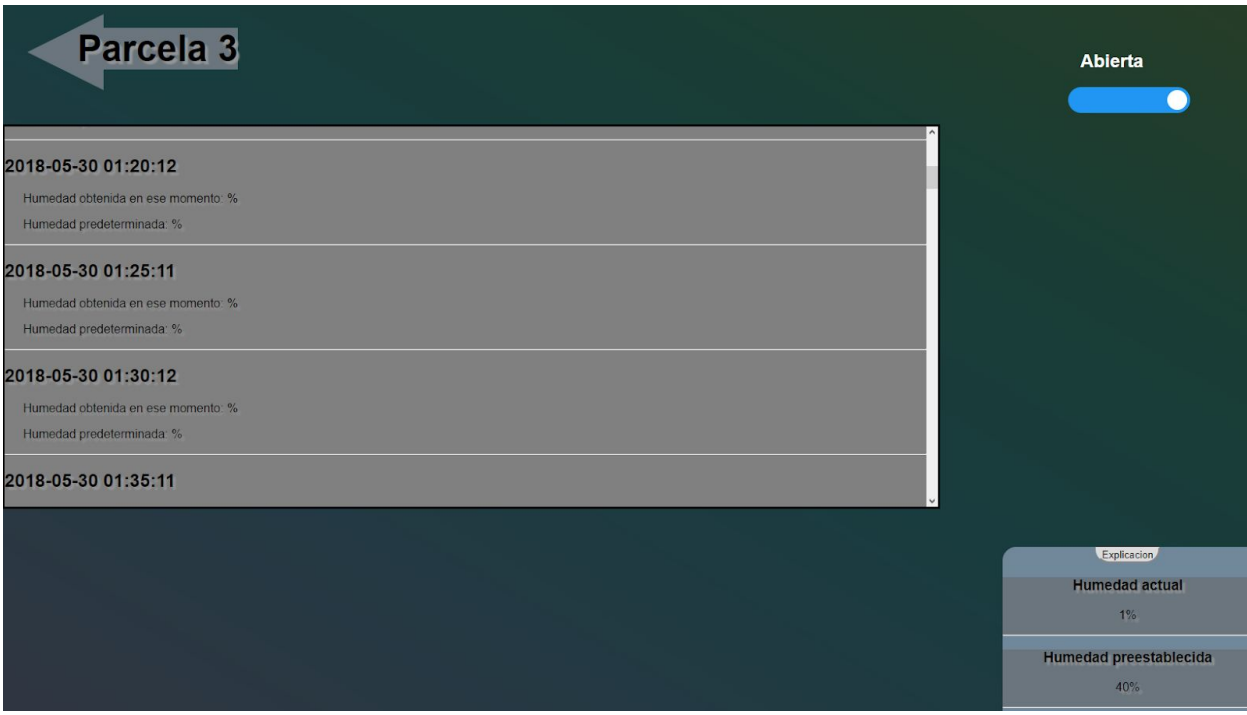
- **Estructura**: la aplicación web tiene únicamente dos página, la inicial y la de cada parcela. Ambas son dinámicas en el sentido de los datos que se muestran. Se generan a partir de la base de datos y el servidor que ejecuta php.
  - Index.php: Aquí muestro las distintas parcelas activas con un máximo de delay de cinco minutos.



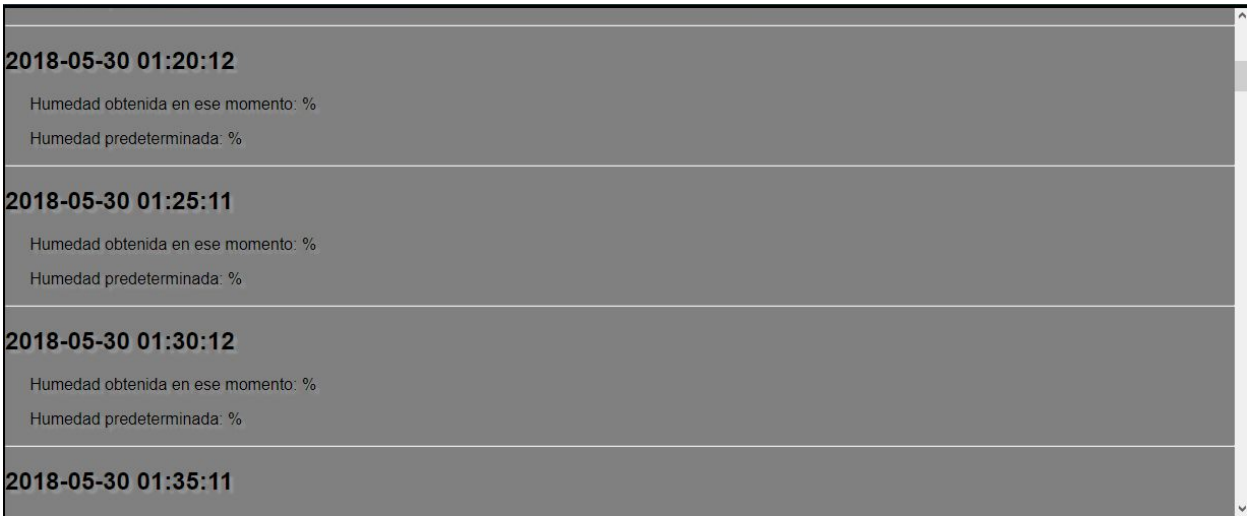
También el algoritmo del tiempo genera un clima según los datos también alojados en la base de datos. Por cada parcela se muestran los siguientes

datos: estos definen si las válvulas de la parcela están abiertas, la humedad que reciben los sensores de esa parcela y el valor por el cual las válvulas se abrirán si la humedad baja por debajo de este.

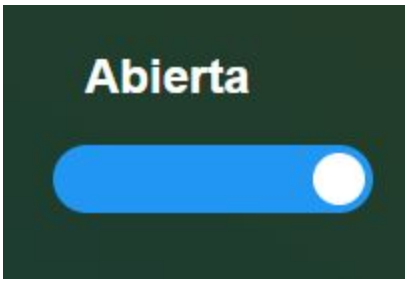
- Parcela.php:



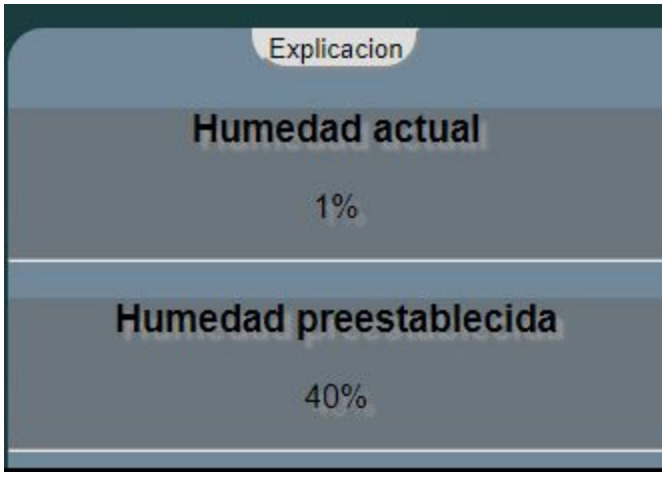
En esta última página se muestran todo el historial de registros de la parcela.



Desde sus válvulas abiertas, que día y hora, hasta el baremo para abrir estas en cada uno de esos registros. En el lado de la derecha, arriba



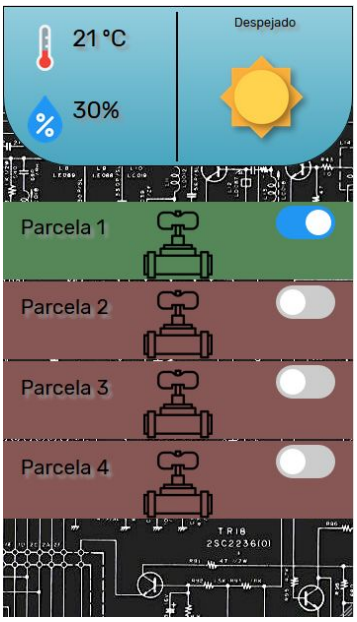
se muestra si las válvulas de la parcela están abiertas, y abajo la humedad actual y el baremo para abrir o cerrar el paso del agua actual.



- Por último, el botón 'explicación' muestra una pequeña descripción que usaré para guiar al cliente en lo que necesite. Mi objetivo es realizar las pruebas con él y ver que necesita o de que no se va a acordar, para ayudarle en lo que él falle, guardaré esa información en este apartado. de momento tiene un pequeño mensaje predefinido

sin mucha importancia.

Por último existe una versión móvil, la cual solo tiene una página y muestra el estado de las válvulas en ese momento. He estado haciendo pruebas, y en la pda da fallos intentar cargar los registro igual que en el ordenador, así que he decidido quitarlo y solo mostrar lo anterior. Creo que es una buena decisión, ya que va mucho más fluida y carga mucho más rápida. De nuevo, todo esto en la pda con Windows Mobile, en cualquier móvil de este año o del año pasado soportaría sin problemas todos esos datos.



En esta versión se puede ver como el algoritmo del tiempo se genera y da un resultado a tiempo real, y las parcelas son representadas por los inputs y el color verde o rojo.

Listado de ficheros

Servidor

1. css
   1.1. comun.css→ Este css guarda los estilos en común de toda la página web, ya sean fondos o colores predeterminados.
   1.2. valvulas.css→ Este css guarda los estilos únicamente de las válvulas, ya que así se puede modificar de maner fácil y rápida.
2. html
   2.1. valvula.php→ Este php muestra la parcela en concreto que se quiere consultar, crea de manera dinámica la página, así siempre salen todos los dato ordenados por fecha.
3. modelo
   3.1. Herramientas.php→ Este archivo* de herramientas lo utiliza la webapp para crear clases de php para que luego sean llamadas desde cualquier archivo. Al ser la página tan pequeña no tiene mucho sentido, es más una apuesta de futuro, ya que si tengo que ampliar la webapp esta estructura me viene genial.
4. multimedia→ Como deja entrever el mismo nombre de la carpeta, almacena todo el poco contenido multimedia que tiene la webapp.
5. Herramientas.php→ Este archivo almacena todas las clases creadas para los scripts de php que ejecuta cron. Básicamente son modelamiento de datos y conexiones con el puerto serial.
6. generar_dia.php→ Este script lo ejecuta cron, lo que hace es recoger todos los datos del día vigente y los comprime para guardarlos en la base de datos. Una vez hecho, borra todos los registro del día en la tabla donde se guardaban.
7. guardar_registro.php→ Este script guarda los datos actuales de los sensores de Arduino en una tabla de la base de datos
8. index.php→ Es la página principal de la webapp.
9. serial_port.php→ Es la única biblioteca que me he descargado de internet. Más adelante explicaré el por qué, pero básicamente lo que hace es crear una conexión con el puerto serial.

*-La carpeta donde se encuentra este archivo se llama model ya que estoy utilizando la estructura vista en Entornos de Desarrollo, MVC (Modelo Vista Controlador).

Arduino

1. Funcionalidad_completa.ino→ Este scketch es el que se compila y manda a Arduino para que este lo ejecute. Va toda la información y programación de este.

Conclusión

Para finalizar esta memoria voy a argumentar algunas de mis decisiones y explicar de alguna de mis estructuras. Seré breve:

Para empezar diré que la única biblioteca que me he descargado de internet es la del serial_port.php, ya que ha sido el mayor problema que he tenido. Esta biblioteca en si no soluciona el problema ni mucho menos, es más, he tenido que modificarla por completo para solucionar el problema por mi mismo. Lo que he utilizado de esta biblioteca es su adaptabilidad, ya que funciona y detecta cualquier ancho de banda y sistema operativo, lo gestiona y lanza las clases que yo he puesto.

El gran problema

No ha sido otro que el serial port. Cuando empecé a diseñar el programa y sistema de riego la idea era que cada vez que hubiera un cambio de estado en una válvula, este se guardase en una base de datos. Con el tiempo e incansable lucha con Arduino acabé entendiendo el por qué de mis intentos fallidos. Arduino es un robot, no es un móvil o un pc, por lo tanto no tiene persistencia de datos. Cada vez que se apaga pierde los datos almacenados, menos su código precompilado. Esto hace que sea imposible guardar ningún valor en él. Por otro lado, la única conexión que hay hacia el exterior de Arduino es por el puerto serie. Este puerto funciona como un cañón de un barco pirata. Solo tiene un disparo, y da igual las veces que lo aprietes, que tiene que recargar. Si nos ponemos en la situación de trabajar con el IDE de Arduino y su aplicación Java para leer el puerto serial no hay problema, funciona perfectamente. El problema es que Java, al nivel que necesita Arduino, no lo soporta la RaspBerry, así que intenté diseñar un sistema con php para que leyese el puerto serial. Al final, con sangre y sudor, lo conseguí. Pero es muy delicado y basto, pasa un buen puñado de información de golpe, y tiene que recargar para que ese puerto serial quede libre, así que me las he ingeniado para limpiar toda esa información de una y poder utilizarla. Conclusión, intentaré que mi arma principal siempre sea Java, ya que es mucho más versátil y potente que php.

Sistema de seguridad

Otra cosa que me ha dado quebraderos de cabeza es el uso excesivo y peligroso de la energía. Si tenemos en cuenta que Arduino hace unas 1600 vueltas al código por segundo, esto produce que una válvula se puede abrir o cerrar 1600 veces por segundo. Así Que

aparte de gastar muchísima energía y poder estropear la válvula y el Arduino, era peligroso ya que se generaba una cantidad de calor excesiva. Así que ante ese problema idee una solución, un sistema de seguridad que utiliza la idea de los hilos (multitarea) en un procesador para mantener un cierto periodo inactiva una válvula tras un cambio de estado. Este sistema detecta cuando una válvula ha cambiado de estado y la guarda en un caché interno. Cuando Arduino vuelve de ejecutar el restante código, comprueba si han pasado los segundos estipulados para esa válvula. Si es así empieza de nuevo las comprobaciones para esa válvula, sino pasa de ella y la deja para la siguiente vuelta.

```
if(valvulaResultado[v]!=resultado && seguridadValvula[v]==0){
 digitalWrite((v+7), HIGH);
 valvulaResultado[v]=resultado;

      if(valvulas[v]<obertura[v]){
        digitalWrite((v+2), HIGH);
        seguridadValvula[v]=millis();
      }else{
        digitalWrite((v+2), LOW);
        seguridadValvula[v]=millis();
      }
}else if((millis()-seguridadValvula[v])>=tiempoSeguridad){
  digitalWrite((v+7), LOW);
  seguridadValvula[v]=0;
}
```

Este es el trozo de código en cuestión que intenta emular de alguna manera los hilos de multitarea de un procesador.

Así pues poco que decir, si hubiera tenido más tiempo libre hubiera añadido dos o tres cosas más, poco relevantes, pero le hubieran dado un toque más profesional. pero creo que en general es un proyecto bastante completo.