Universidad de La Laguna

ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA

Proyecto Fin de Grado

SISTEMA DE CONTROL DE RIEGOS GESTIONADOS TELEMÁTICAMENTE

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

Alumnos: Raúl Buendía Cabrera

Roberto Chávez Trujillo

Tutores: Alejandro José Ayala Alfonso

Beatriz Rodríguez Mendoza

Julio, 2016

Índice

Abstract

The aim of the project is focused on the design and implementation of an irrigation system managed electronically.

A central station, which we call Master, can wirelessly control several substations, (the slaves), spread over the garden. The user enters through a screen and keyboard included in the hardware, garden characteristics: type of plant, acreage and irrigation flow.

Slaves have a power system itself. And thanks to sensors which have each slave terminal, you can get information on environmental variables and pass them to the Master, who can adapt programming to environmental conditions.

All data received by the Master are sent to an external server so that the ability to access data and control the system remotely is taken.

CAPÍTULO I.-

INTRODUCCIÓN GENERAL.

OBJETIVOS

CAPÍTULO I.- Introducción general. Objetivos

I.1.- Introducción general

El ser humano necesita alimentos para su subsistencia. Durante millones de años las primitivas sociedades de homínidos se organizaron en grupos de nómadas cazadores-recolectores para sobrevivir, pero hace aproximadamente 11000 años se produjo una de las mayores evoluciones en la historia de la humanidad: la denominada revolución neolítica. Las sociedades se asentaron y comenzó la ganadería y la agricultura.

Poco a poco fueron creciendo los asentamientos y formándose civilizaciones que necesitaban mayor producción agrícola. Las técnicas mejoraron. Los egipcios aprovecharon las crecidas del Nilo y gracias a un calendario para predecirlas y un complejo sistema de diques y canales fueron capaces de aprovechar este fenómeno de la naturaleza para convertirse en unos de los pioneros en la agricultura. Milenios más tarde los romanos comenzaron a fabricar acueductos y tuberías para transportar agua a grandes distancias salvando todo tipo de obstáculos.

Se fueron produciendo mejoras en la tecnología agrícola, sobre todo con la revolución industrial y posteriormente con el bombeo mediante motor eléctrico que facilitó mucho el proceso de la agricultura en general y el regadío en particular. Sin embargo, han surgido nuevos problemas y desafíos en la edad moderna.

Por un lado, el imparable aumento de la población, lo que lleva a unas mayores necesidades de producción, junto con cambios en el clima han hecho que cada vez exista una mayor preocupación por la escasez del agua y la necesidad de hacer un uso eficiente de la misma. En definitiva, lograr un desarrollo sostenible y una producción agrícola respetuosa con el medio ambiente. Por otro lado, la sociedad busca cada vez más liberarse del trabajo pesado y aumentar la productividad, cuestión que está comenzando a hacerse realidad con el llamado «internet de las cosas».

La finalidad del proyecto ha sido continuar con el desarrollo sostenible, haciendo uso de las tecnologías disponibles actualmente, con el fin de optimizar lo máximo posible el riego haciendo uso de la información obtenida a través de la medida de diversas variables ambientales.

I.2.- Objetivos

El objetivo del proyecto se ha centrado en el diseño e implantación de un sistema de riego gestionado telemáticamente.

Una estación central, que denominaremos Maestro, permite controlar de forma inalámbrica a varias subestaciones, que llamaremos Esclavos, repartidas por el terreno. El usuario introduce, mediante una pantalla y teclado incluidos en el hardware, las características del huerto: tipo de planta, superficie de cultivo y caudal de riego.

Los Esclavos cuentan con un sistema de alimentación propio. Además, gracias a los sensores de los que dispone cada terminal Esclavo, se podrá obtener información de variables ambientales y transmitirlas al Maestro, que podrá adaptar la programación a las condiciones cambiantes del entorno. Además se realizará el cálculo de un parámetro denominado evapotranspiración, a partir del cual se puede obtener una buena aproximación de las necesidades hídricas reales del cultivo.

Todos los datos que reciba el Maestro se enviarán a un servidor externo de manera que se tenga la posibilidad de acceder a éstos y controlar el sistema de forma remota.

En la figura siguiente se observa el esquema general del sistema.

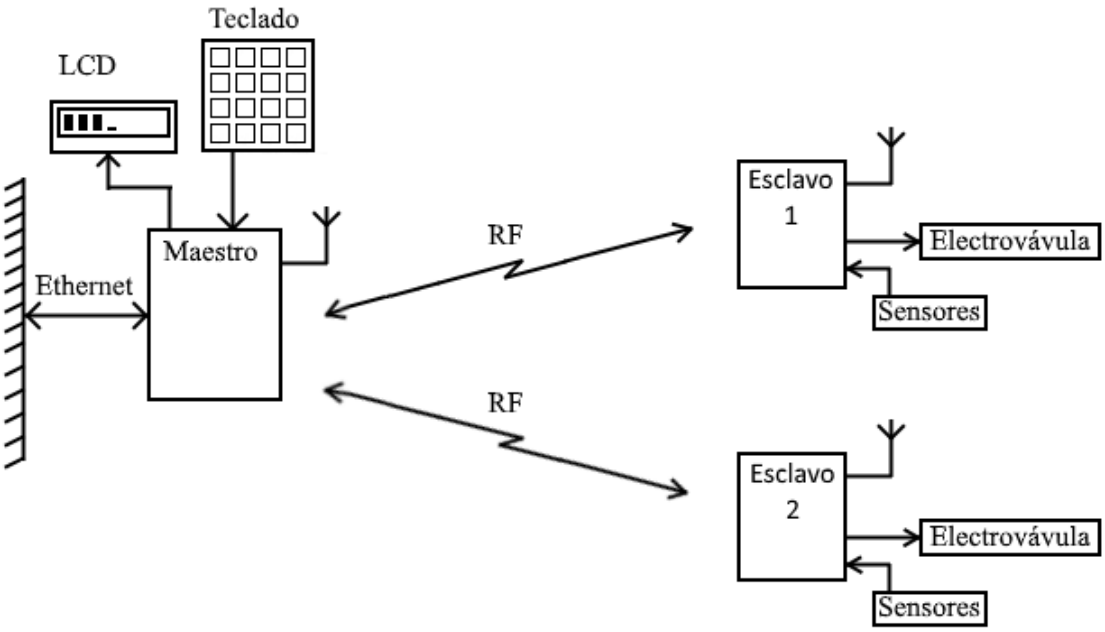


Figura I.1.- Sistema de riego

I.3.- Estructura de los Capítulo s de la memoria

La memoria se desarrolla en 7 Capítulos.

El primero de ellos ofrece una visión general del tema del trabajo, sus objetivos y la forma en la que está estructurada la presente memoria.

En el Capítulo II, se describen las principales características de las placas Arduino que se usan en el sistema, así como los diferentes componentes que la forman, siendo los microcontroladores las piezas fundamentales.

El Capítulo III se centra en las variables ambientales que mide el sistema para regular su actuación: temperatura, humedad del suelo, lluvia, presión atmosférica. La importancia que tienen en el sistema de control, la manera en que afectan a las plantas y los modos de detectar su variación mediante los sensores adecuados, serán abordados en el presente Capítulo.

Las comunicaciones se explican en el Capítulo IV. Esto incluye las transmisiones mediante radiofrecuencia entre los Esclavos y el Maestro, así como las características básicas de la red implementada y los transmisores utilizados, además de la conexión del Maestro con el servidor web.

A continuación, en el Capítulo V se define y detalla el proceso de cálculo del parámetro de la evapotranspiración. Se incluyen además otros dos aspectos claves del sistema, por un lado, la manera en la que se alimentan las placas Arduino y las posibles alternativas. Por otro lado, el hardware necesario para adaptar la señal de salida del Arduino a la tensión necesaria para activar la electroválvula de riego

En el Capítulo VI se detalla el software empleado para regular todo este proceso. Se detallarán los flujos de información y de decisión, así como el funcionamiento del mismo desde el punto de vista del usuario.

Por último, en el Capítulo VII se mostrarán algunos resultados experimentales obtenidos mediante el prototipo en forma de gráficas de variables ambientales, y se hará una valoración de las posibles mejoras que se podrían realizar sobre el proyecto en un futuro.

CAPÍTULO II.-
CARACTERÍSTICAS GENERALES

CAPÍTULO II.- Características generales

II.1.- Introducción

El sistema se ha implementado utilizando placas Arduino. Se trata de una placa de desarrollo que incluye un microcontrolador, puertos de entrada salida a los que se pueden conectar sensores y actuadores, además de otros componentes que facilitan la construcción de prototipos. Cuenta, asímismo, con un lenguaje de programación propio y un software libre para programar el controlador desde cualquier ordenador.

Las placas Arduino se escogieron para este proyecto por su versatilidad y su bajo costo. El hecho de que es una plataforma de hardware y software libre ayuda a que exista mucha información sobre su funcionamiento. Se encuentra respaldada por una gran comunidad de usuarios que han ampliado y permitido el acceso a dicha documentación.

Arduino dispone de diversos tipos de placas, para este proyecto se ha optado por el modelo Arduino Mega y Arduino Nano, cuyas características se exponen a continuación.

II.2.- Plataforma Arduino

Aparte de las placas, la plataforma Arduino incluye un entorno de desarrollo y un leguaje propios (Figura II.1).

El entorno de desarrollo es de software libre y multiplataforma (Windows, Linux y Apple), dicho entorno sirve para escribir programas y cargarlos en la memoria flash del microcontrolador, esto se consigue mediante un cable USB que comunica el Arduino con el ordenador donde se escribe el programa. La mayoría de placas dispone de un conector USB incorporado y una vez programado el microcontrolador, ya no se necesitará el entorno de programación ni el cable USB y la placa funcionará de forma autónoma (siempre que disponga de alimentación eléctrica).

El lenguaje de programación que se utiliza está basado en C, aunque es una versión más simplificada. Contiene muchos elementos comunes a otros lenguajes como los bloques condicionales, bucles, variables, funciones, etc.

Tanto el hardware como el software son libres. Esto implica que los fabricantes permiten estudiar cómo funciona internamente, acceso al código fuente y a los esquemas de circuitos. Se pueden distribuir copias y mejorarlo o adaptarlo a nuestras necesidades. Además de permitir su uso en cualquier sistema informático y con el propósito que se desee.

De manera resumida, las principales ventajas de Arduino son: software y hardware libre, entorno de programación multiplataforma, lenguaje de programación claro y sencillo, placas baratas y reutilizables.

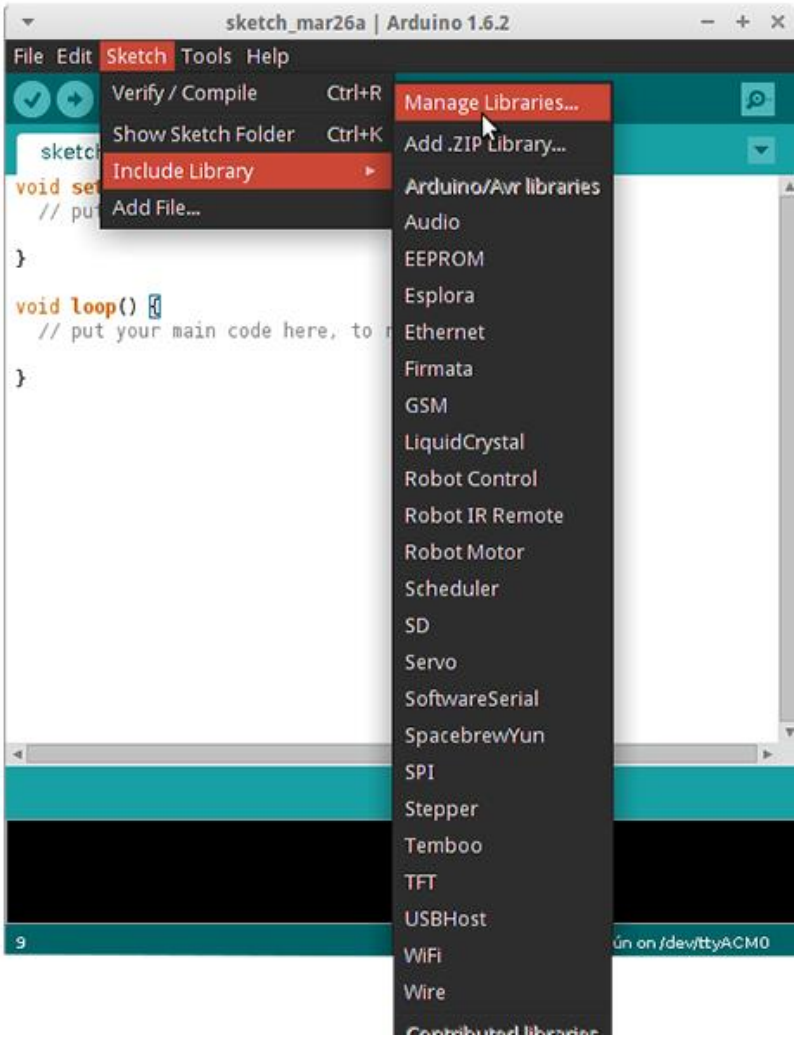


Figura II.1.- Entorno de programación

II.3.- Características comunes de las placas Arduino

Alimentación: Las placas Arduino se alimentas con 5 V de continua. Algunas incluyen un zócalo donde poder conectar directamente un adaptador AC/DC. En este caso, el rango recomendado de tensión a introducir en el regulador de la placa se encuentra comprendido entre 7 y 12 voltios.

Mediante el cable USB también se puede obtener el voltaje necesario del ordenador.

Chip ATmega16U2: Cuando se conecta el Arduino al ordenador este chip se encarga de traducir el protocolo USB a otro entendible por el microcontrolador.

Entradas y salidas digitales: En estos pines se conectan sensores y actuadores para comunicarse con el microcontrolador. Funcionan a 5 V.

Entradas analógicas: Cada canal dispone de 10 bits de resolución, admitiendo un rango de valores de 0 a 5 voltios. Se pueden usar también como entradas digitales.

Salidas analógicas (PWM): La placa Arduino no puede ofrecer directamente una tensión analógica, en vez de eso, es capaz de generar pulsos de amplitud constante y duración variable (Pulse Width Modulation) que emulan una señal analógica según la duración de los éstos

Otras funciones especiales: Algunos pines, además de su uso normal, tienen asignadas funciones especiales. Por ejemplo: transmisión y recepción de datos en serie (RX, TX), gestión de interrupciones, usar el protocolo SPI (MISO, MOSI, SCK, SS), etc.

Reloj: La placa Arduino incluye un reloj propio de 16 MHz para controlar la velocidad de la secuencia de instrucciones. Es del tipo resonador cerámico.

Botón de reset: Dispone de un pequeño botón que sirve para parar y volver a reiniciar el microcontrolador, de modo que ejecuta de nuevo el programa que hubiera previamente en la memoria.

II.4.- Arduino Mega

El Arduino Mega (Figura II.2) se eligió como dispositivo Maestro por su elevada capacidad tanto de pines de conexión como de memoria, pues al tener que coordinar a los Esclavos, comunicarse con los dispositivos periféricos y con el router, necesita mayores prestaciones.

Sus características más destacadas son:

- Microcontrolador ATMEL mega 2560.
- 54 pines de entrada y salida digitales (14 se pueden usar como salidas analógicas PWM).
- 16 entradas analógicas.
- 4 Receptores/transmisores serie TTL-UART.

- Memoria Flash de 256 Kilobytes (8 KB reservados para el bootloader).
- Memoria SRAM de 8 KB.
- Memoria EEPROM de 4 KB.
- Tensión de trabajo de 5 V.
- Dimensiones: 101,52 mm x 53,3 mm.

Figura II.2.- Arduino MEGA

II.5.- Arduino Nano

Se optó por el Arduino Nano (Figura II.3) dado que las funciones que realizan los dispositivos Esclavos en este sistema son más simples y no es necesario utilizar un dispositivo de mayores prestaciones. Otra ventaja es que ocupa menor espacio, haciéndolo más manejable.

- Microcontrolador ATMEL mega 328.
- 14 pines de entrada y salida digitales (6 se pueden usar como salidas analógicas PWM).
- 8 entradas analógicas.
- 1 Receptores/transmisores serie TTL-UART.

- Memoria Flash de 32 Kilobytes (2 KB reservados para el bootloader).
- Memoria SRAM de 2 KB.
- Memoria EEPROM de 1 KB.
- Tensión de trabajo de 5 V.
- Dimensiones: 45 mm x 18 mm.

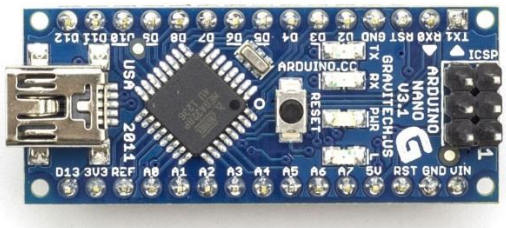


Figura II.3 Arduino Nano

CAPÍTULO III.- VARIABLES AMBIENTALES

CAPÍTULO III.- Variables ambientales

III.1.- Introducción

En este proyecto, la toma de datos ambientales tiene un papel muy relevante y es, de hecho, uno de los aspectos diferenciadores respecto a otros sistemas de riego automatizados.

Como se detallará en capítulos sucesivos, dicha toma de datos tiene como fin el cálculo del parámetro denominado *evapotranspiración*, que indica la cantidad de agua que se pierde del conjunto suelo - planta por evaporación y transpiración, con lo que se puede obtener una buena estimación de las necesidades hídricas del cultivo.

A continuación, se expondrán las variables ambientales que han sido consideradas a lo largo del presente proyecto, así como las que han sido descartadas, y las características de los sensores utilizados para tal fin.

III.2.- Temperatura

La temperatura es uno de los parámetros más importantes a tener en cuenta en un sistema de control de riego como el que se aborda, e incluso es posible realizar una estimación aceptable de la evapotranspiración a partir únicamente de los valores de temperatura máximo y mínimo.

III.2.1.- Características del sensor

Aunque los sensores de humedad relativa y presión barométrica también son capaces de ofrecer datos de temperatura, se ha decidido utilizar uno específico para tal fin, puesto que ofrece una mayor precisión, y por razones de seguridad, ya que un fallo en el sensor de humedad provocaría una pérdida de datos de las dos variables más importantes desde el punto de vista del control.

El sensor elegido fue el modelo DS18B20 [1] del fabricante Maxim Integrated (Dallas Semiconductor).

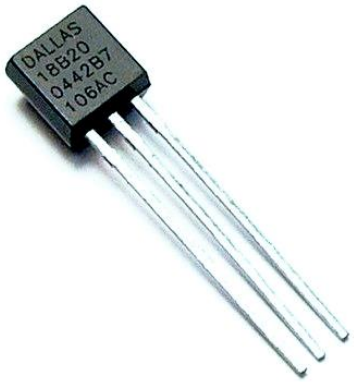


Figura III.1.- Sensor de temperatura

Se trata de un sensor digital con una resolución ajustable de 9 a 12 bit, que incluye funciones de alarma con umbrales de disparo superior e inferior programables por el usuario y un rango de medida de la temperatura entre -55 ºC y +125 ºC.

Este sensor incluye una interfaz de comunicación 1-Wire® única de este fabricante, que requiere de una sola línea de datos, además de la alimentación y tierra. También es posible un segundo modo de conexión, denominado «parásito», en el que sólo son necesarios dos cables, obteniendo la alimentación de la línea de datos.

Cada dispositivo tiene un código serie de 64 bits que lo diferencia unívocamente del resto, lo que permite tener múltiples sensores en el mismo bus 1-Wire.

A continuación se detallan sus características principales (Tabla III.1), así como los bloques internos que lo componen (Figura III.2):

Parámetro	**MIN**	**TYP**	**MAX**
Tensión de alimentación	+3,0 V		+5,5 V
Rango de temperatura	-55 ºC		+125 ºC
Error en la medida			± 0,5 V
Corriente en Standby		750 nA	1000 nA
Corriente durante medición		1 mA	1,5 mA

Tabla III.1.-Características del sensor de temperatura

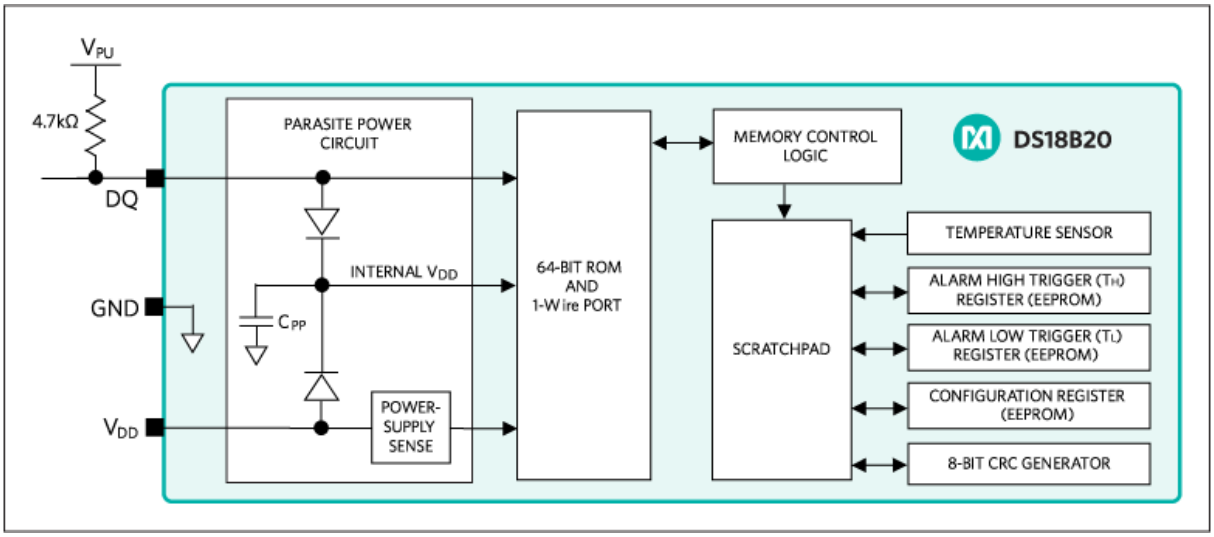


Figura III.2.-Bloques internos del sensor

En cuanto a los aspectos de comunicación en el bus 1-Wire y obtención de los datos del sensor, la secuencia que se realiza es la siguiente:

1. Inicialización.
2. Comandos ROM.
3. Comandos de funciones DS18B20.

Todas las transacciones en el bus comienzan con una secuencia de inicialización. Este primer paso consiste en el envío de un pulso de «reset» por parte del microcontrolador, seguido por la correspondiente respuesta de los sensores DS18B20, de modo que se pueda comprobar su presencia (Figura III.3).

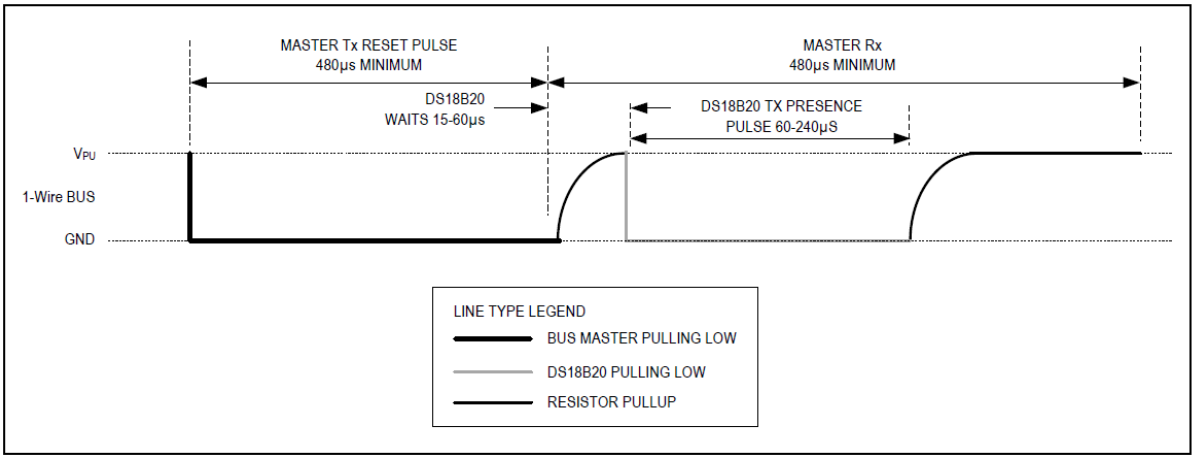


Figura III.3.-Secuencia de inicialización

Después que el microcontrolador haya recibido el pulso de respuesta, se procede a emitir un «comando ROM», con los que se puede obtener el número y tipo de sensores que se encuentran en el bus e identificar un dispositivo específico gracias a su código serie.

Finalmente, una vez que se ha seleccionado el dispositivo en concreto con el que se desea comunicar, el microcontrolador puede emitir alguno de los comandos de función que permiten, entre otras tareas, leer y escribir desde la memoria del sensor e iniciar la obtención de temperatura, como se puede observar en la siguiente tabla (Tabla III.2):

COMMAND	DESCRIPTION	PROTOCOL	1-Wire BUS ACTIVITY AFTER COMMAND IS ISSUED	NOTES
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	DS18B20 transmits conversion status to master (not applicable for parasite-powered DS18B20s).	1
MEMORY COMMANDS				
Read Scratchpad	Reads the entire scratchpad including the CRC byte.	BEh	DS18B20 transmits up to 9 data bytes to master.	2
Write Scratchpad	Writes data into scratchpad bytes 2, 3, and 4 (T_H, T_L, and configuration registers).	4Eh	Master transmits 3 data bytes to DS18B20.	3
Copy Scratchpad	Copies T_H, T_L, and configuration register data from the scratchpad to EEPROM.	48h	None	1
Recall E^2	Recalls T_H, T_L, and configuration register data from EEPROM to the scratchpad.	B8h	DS18B20 transmits recall status to master.	
Read Power Supply	Signals DS18B20 power supply mode to the master.	B4h	DS18B20 transmits supply status to master.	

Tabla III.2.- Comandos de función disponibles

En este proyecto, por razones de claridad desde el punto de vista de la programación, pero en detrimento de la optimización de la memoria del microcontrolador, se optó por utilizar «librerías software», que ya incluyen los «métodos» necesarios para realizar las funciones anteriormente descritas. Se trata de las «librerías» *OneWire* y *DallasTemperature*.

III.3.- Temperatura del suelo

La Temperatura del suelo agrícola condiciona los procesos microbianos que tienen lugar en el mismo. La temperatura también influye en la absorción de los nutrientes y en los procesos bióticos y químicos. [2]

En un principio, se propuso la temperatura del suelo como otra variable más para ser monitorizada pero, finalmente, se llegó a la conclusión de que la utilización de estos datos no sería relevante para la determinación de las necesidades de riego del cultivo.

III.3.1.- Características del sensor

El sensor propuesto fue la versión sumergible del modelo DS18B20 de Dallas Semiconductor mencionado anteriormente.

Figura III.4.- Sensor DS18B20 sumergible

III.4.- Humedad relativa

La humedad relativa es, junto con la temperatura, uno de los parámetros más importantes para llevar a cabo una buena estimación de la evapotranspiración del cultivo.

III.4.1.- Características del sensor

Para la obtención de los datos de humedad se ha usado el sensor DHT22 (Figura III.5) [3] del fabricante Aosong Electronics Co. Se trata de un dispositivo con salida digital, que incluye un sensor capacitivo para la obtención de la humedad y un termistor NTC para la temperatura.

Figura III.5.- Sensor de humedad DHT22

A continuación, se detallan sus características principales (Tabla III.3):

Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

Tabla III.3.- Características principales

Procedimiento para comunicación con el sensor y obtención de datos (Figura III.6):

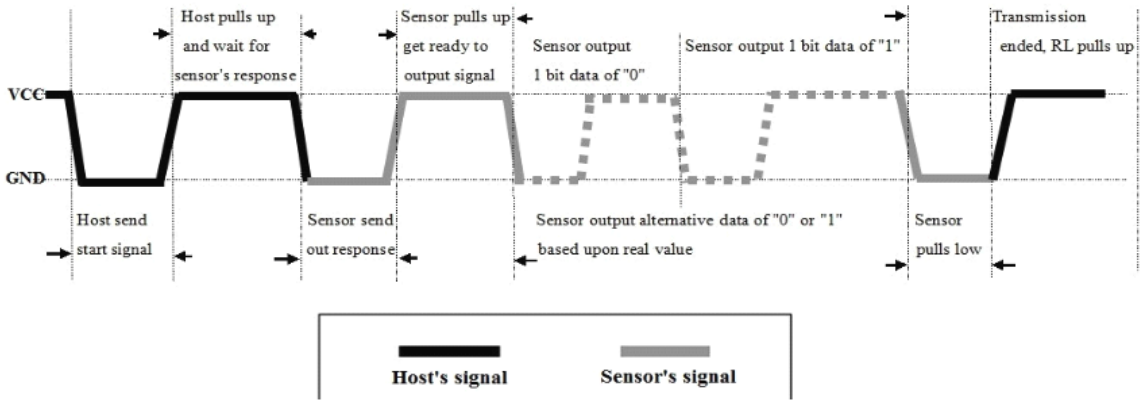


Figura III.6.- Secuencia de comunicación

1. Se envía la señal de «start» al dispositivo.

Mientras el bus de datos está libre, se encuentra a una tensión de V_{CC}. Para comenzar, el microcontrolador debe llevar el nivel de tensión a tierra al menos durante 1 ms para asegurar que el sensor ha detectado la señal. A continuación, se procede a esperar la respuesta entre 20 y 40 µs.

2. El DHT22 envía la respuesta al microcontrolador.

Una vez que se ha detectado la orden de inicio, el sensor responde con una señal a bajo nivel de tensión durante 80 µs y, seguidamente, a V_{CC} con la misma duración.

3. Envío de datos al microcontrolador.

Finalmente, el sensor devuelve los datos de la medida al MCU. Cada bit transmitido comienza con un nivel de tensión bajo durante 50 µs y, a continuación, la duración del pulso de alto nivel determina el valor del bit correspondiente (Tabla III.4).

Duración del pulso	Valor del bit
26 – 28 µs	‘0’
70 µs	‘1’

Tabla III.4.- Valor del bit

Al igual que con el sensor de temperatura, se ha hecho uso de una librería que facilita las tareas de operación con este dispositivo. Se trata de la librería *DHT*.

III.5.- Humedad del suelo

Durante las primeras etapas de desarrollo de este proyecto se estudió la posibilidad de utilizar la medida de la humedad del suelo como el parámetro más relevante para el algoritmo de control de riego. Sin embargo, después de consultar con profesores del ámbito de la ingeniería agraria, la recomendación fue no utilizar este tipo de sensores resistivos, pues el parámetro de la conductividad no depende únicamente de la humedad, sino también de otros factores como el tipo de tierra, las sales disueltas y la cantidad de materia orgánica existente. Esto unido a la poca calidad del sensor (baja resistencia a la corrosión), han llevado a descartar su uso en el presente proyecto.

III.6.- Presión atmosférica

«La variación de la presión a lo largo del tiempo permite obtener una información útil que, unida a otros datos meteorológicos (temperatura atmosférica, humedad y velocidad del viento), puede dar una imagen bastante acertada del tiempo atmosférico e incluso un pronóstico a corto plazo del mismo.» [4].

Los datos de la presión atmosférica no se utilizan directamente en el cálculo de la *evapotranspiración*. En un principio, se intentó aprovechar dicha información para implementar un sistema con el cual obtener una predicción relativamente fiable de las posibilidad de precipitaciones, basado en las variaciones de la presión y acompañado de otros parámetros ambientales como la temperatura y la humedad. Al igual que en el caso anterior, se acudió al consejo de expertos en el área de observación de la atmósfera y meteorología, los cuales desaconsejaron la realización de dicho sistema debido a la complejidad de los algoritmos de predicción necesarios y la poca exactitud que se podría obtener.

A pesar de ello, se ha decidido mantener este sensor de modo informativo, aunque no se utilice en ninguno de los aspectos del control.

III.6.1.- Características del sensor

El sensor utilizado es el *BMP180* (Figura III.7) [5] del fabricante Bosch Sensortec (Robert Bosch, GmbH). Se trata de un sensor digital de precisión y bajo consumo, optimizado para el uso en dispositivos portátiles como teléfonos móviles, navegadores GPS y equipos similares, en los que la disponibilidad de energía es limitada.

Está basado en un «Sistema Microelectromecánico» (MEMS), concretamente en la tecnología piezo-resistiva. Además, cuenta con una interfaz I^2C, en la que sólo son necesarias dos líneas de señal, de modo que facilita las labores de comunicación con el microcontrolador.

Figura III.7.- Sensor de presión BMP180

El modelo empleado viene incluido en una placa de circuito impreso junto con el regulador de tensión de 3,3 V necesario para el correcto funcionamiento del sensor.

A continuación, se detallan sus características principales (Tabla III.5):

Parámetro	Condición	Min	Typ	Max	Unidades
Tensión de alimentación		1,62	2,5	3,6	V
Corriente de alimentación	Modo ultra baja potencia		3		µA
	Modo estándar		5		µA
	Modo ultra alta resolución		12		µA
Corriente en reposo	A 25 ºC		0.1		µA
Precisión en la presión	300 – 1100 hPa	-4,0	±1,0	+2,0	hPa
Precisión en la temperatura	0 – 65 ºC	-2,0	±1,0	+2,0	ºC

Tabla III.5.- Características del sensor

Como se puede observar en la imagen siguiente (Figura III.8), el BMP180 está compuesto internamente por el bloque del sensor propiamente dicho, un conversor analógico digital, una unidad de control con memoria EPROM y la interfaz I^2C. En la EPROM se almacena una serie de bits de datos de calibración usados para compensar la dependencia de la temperatura y el «error de offset» en las medidas.

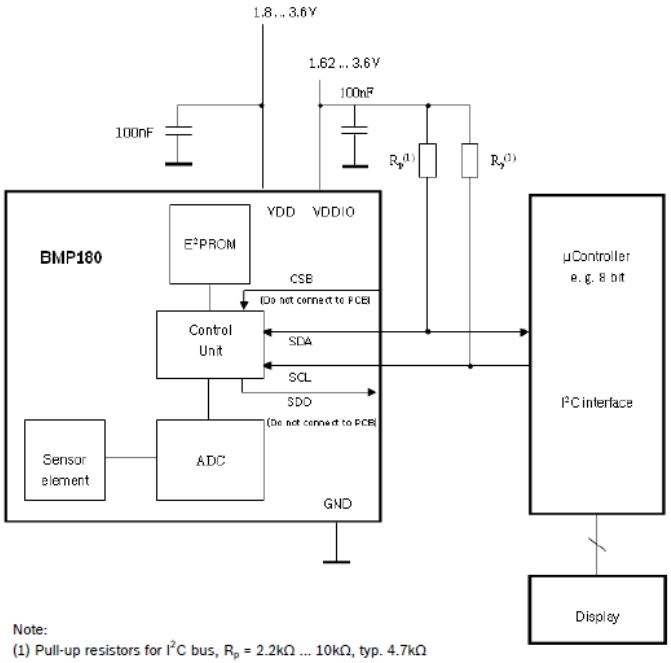


Figura III.8.- Diagrama de bloques

A continuación, se expondrán los aspectos más relevantes referentes a la interfaz I^2C (Figura III.9):

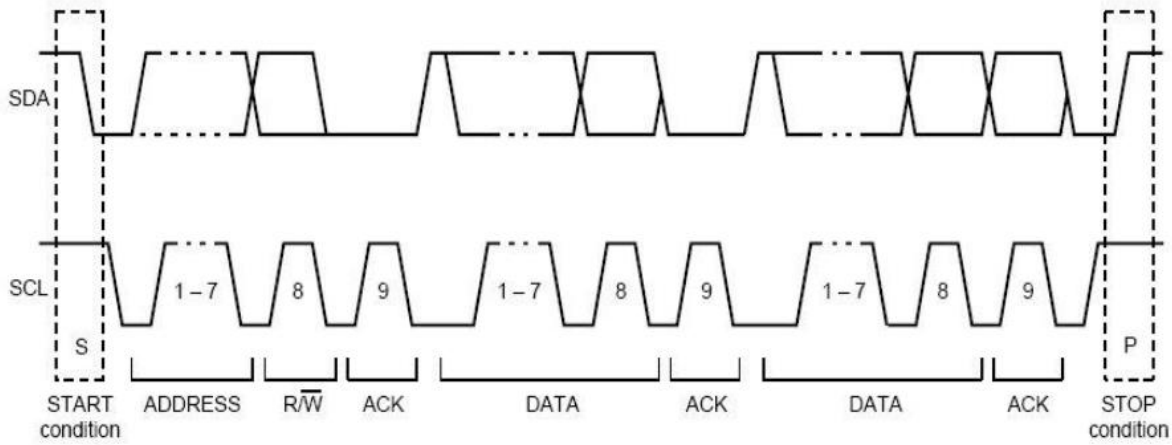


Figura III.9.- Señales bus I2C

El I^2C, es un bus de datos serial, utilizado para controlar el sensor, leer la calibración y los datos medidos una vez que se ha terminado la conversión analógica – digital. El protocolo de comunicación comienza con la condición de «Start». En ese momento la línea SCL (reloj) se encuentra al valor de tensión V_{CC}, mientras que en SDA (datos) ocurre un flanco de bajada. Seguidamente se envía la dirección del sensor que se encuentra en el bus, y después de ésta, el bit de control R/W, con el que se selecciona la operación de lectura o escritura. Cuando el sensor reconoce que está siendo direccionado, responde llevando la línea SDA a tierra en el noveno ciclo de SCL.

El sensor envía una confirmación de recepción (ACKS) cada 8 bits de datos recibidos.

En la condición de parada, SCL también se encuentra a valor alto de tensión, pero en SDA ocurre un flanco ascendente.

Para el uso de este sensor se ha empleado la «librería» *Wire* y *Adafruit_BMP085*.

III.7.- Sensor detector de lluvia

En este proyecto se optó por hacer uso de un sensor detector de precipitaciones con el fin de utilizar estos datos en el sistema de control, evitando así el riego en esas circunstancias.

III.7.1.- Características del sensor

Se trata de un sensor (Figura III.10) con un funcionamiento muy simple que entrega un valor de tensión analógica dependiendo de la cantidad de agua que haya caído sobre éste.

El sensor consta de dos pistas conductoras paralelas y aisladas entre sí sobre una placa de circuito impreso, de modo que cuándo se deposita una gota de agua sobre éste, se provoca un contacto eléctrico entre dichas pistas. Dicho de otro modo, a mayor número de contactos, disminuye la resistencia eléctrica vista desde los terminales de conexión del sensor, por lo que simplemente, introduciéndolo en un circuito divisor de tensión, se obtiene un valor de voltaje proporcional a la cantidad de agua acumulada.

Figura III.10.- Detector de lluvia

Desde el punto de vista práctico, conseguir estos valores analógicos no tienen gran utilidad, y por ello, se ha empleado un circuito comparador (LM393) para mediante la definición de un umbral de disparo, obtener un valor de salida digital (todo – nada).

III.8.- Otros dispositivos. Reloj de tiempo real (RTC)

El reloj de tiempo real (RTC) es uno de los componentes vitales del sistema. Es necesario pues se usa en el registro de los datos de los sensores, que se guardan en una tarjeta de memoria junto con la fecha y la hora. Además se usan sus alarmas para generar interrupciones periódicas en las que el microcontrolador Maestro solicita los datos a los Esclavos.

Estas funciones no podrían realizarse con el «timer» de Arduino, puesto que éste se reinicia por «overflow[1]» de la variable que lo almacena cada 50 días aproximadamente.

Aparte de esto, contar con un dispositivo de este tipo tiene otras ventajas como un bajo consumo de energía (respecto a otros métodos de medida del tiempo), libera de trabajo al microcontrolador principal y suele ser más preciso que éste.

[1] En el sistema Arduino el timer almacena los milisegundos transcurridos desde el encendido. Se usa una variable del tipo Unsigned long de 32 bits, por lo tanto: $\frac{2^{32}}{ms(día)} = 49{,}71\ días$.

III.8.1.- Características del reloj RTC

El reloj utilizado es el modelo DS3231 (Figura III.11) [6] del fabricante Maxim Integrated. Se trata de un dispositivo extremadamente preciso, con comunicación mediante el protocolo I^2C, un oscilador de cuarzo integrado de 32 kHz con compensación de temperatura (TCXO) y un cristal, con lo que se consigue una de precisión de ±2 minutos por año.

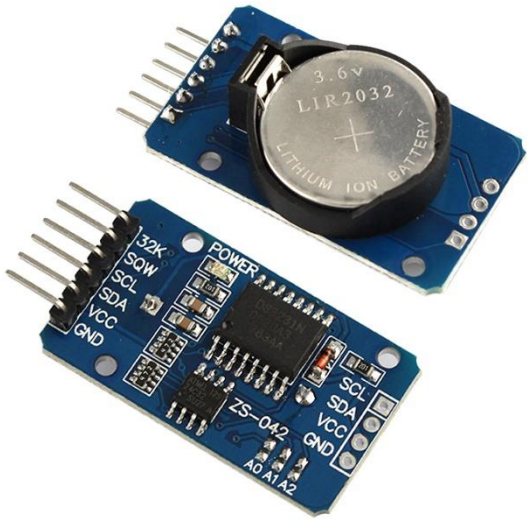


Figura III.11.- Reloj DS3231

Incorpora una entrada para una batería externa, de modo que se puede mantener una precisa medida del tiempo incluso cuando la alimentación es interrumpida.

Este reloj es capaz de ofrecer información sobre los segundos, minutos, horas, día, mes y año, válido hasta el 2100. La fecha se ajusta automáticamente dependiendo del mes e incluye correcciones para los años bisiestos. También funciona en el formato de 12 y 24 horas, dispone de dos alarmas programables y salida de onda cuadrada de frecuencia ajustable.

En la tabla siguiente se pueden observar sus características principales (Tabla III.6), así como su diagrama de bloques (Figura III.12):

Parámetro	Min	Typ	Max	Unidades
Tensión de alimentación	2,3	3,3	5,5	V
Corriente de alimentación			200	µA
Corriente en reposo			110	µA
Precisión en la frecuencia			±2,0	ppm
Precisión en la temperatura	-3,0		+3,0	ºC

Tabla III.6.- Características principales

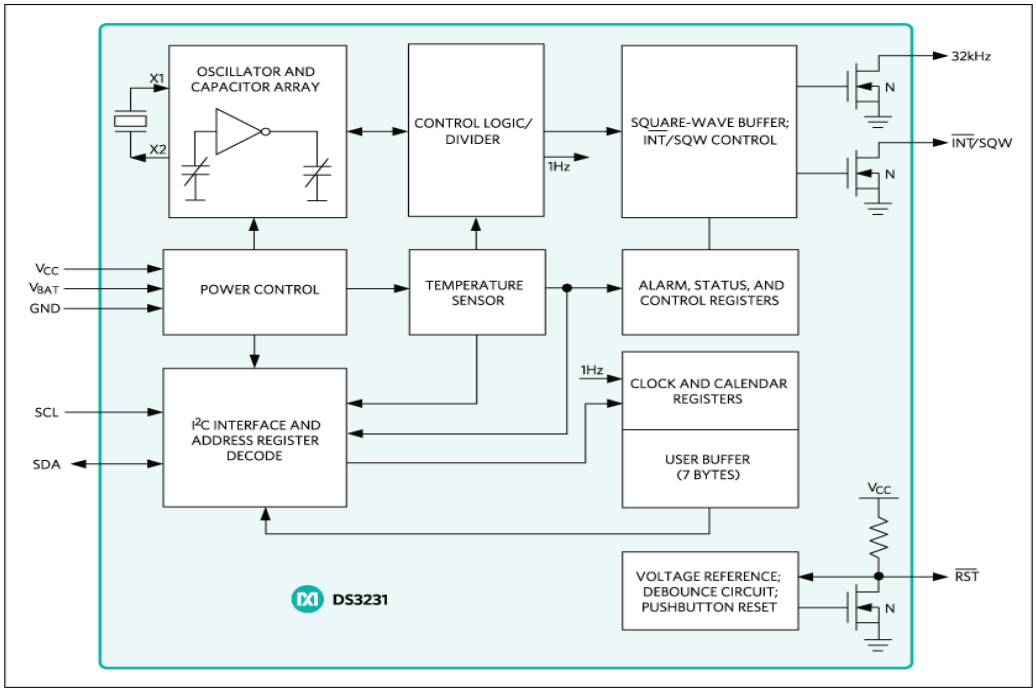


Figura III.12.- Diagrama de bloques

III.8.1.1.- Alarmas

Uno de los aspectos más importantes y el factor decisivo que llevó a la elección de este modelo, es la posibilidad de establecer alarmas horarias. Este hecho permite la utilización de dicha información para llevar a cabo tareas de forma cíclica en instantes determinados. En concreto se emplea en el sistema de adquisición de datos, en el cuál el microcontrolador Maestro solicita vía radio cada 30 minutos el valor de los sensores a los múltiples dispositivos Esclavos. Para ello se utiliza la señal de interrupción que se genera en el pin $\overline{INT}$/SQW cuando se dispara la alarma.

Ajuste de las alarmas

El DS3231 dispone de dos alarmas ajustables, la primera, *Alarm 1* se establece escribiendo en los registros del 07h al 0Ah, mientras que *Alarm 2* se ajusta mediante los registros del 0Bh al 0Dh, como se puede observar en la siguiente tabla (Tabla III.7).

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/$\overline{DT}$	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31

Tabla III.7.- Registro de las alarmas

El pin $\overline{INT}$/SQW tiene doble funcionalidad, puede proporcionar tanto una señal de interrupción cuando se cumplen las condiciones de la alarma, o una onda cuadrada de frecuencia ajustable. Por ello primero se debe seleccionar la función deseada haciendo uso del bit *INTCN* y «alarm enable», (*A2IE* y *A1IE)*, del registro de control 0Eh (Tabla III.8).

Control Register (0Eh)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
NAME:	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE
POR:	0	0	0	1	1	1	0	0

Tabla III.8.- Registro de control

Una vez habilitadas las funciones de alarma, se deben ajustar los valores deseados en los bits del registro de alarma visto anteriormente, de acuerdo a la siguiente tabla (Tabla III.9), en la que se muestra la máscara de bits de dicho registro con las diversas combinaciones posibles.

DY/$\overline{DT}$	ALARM 1 REGISTER MASK BITS (BIT 7)				ALARM RATE
	A1M4	A1M3	A1M2	A1M1	
X	1	1	1	1	Alarm once per second
X	1	1	1	0	Alarm when seconds match
X	1	1	0	0	Alarm when minutes and seconds match
X	1	0	0	0	Alarm when hours, minutes, and seconds match
0	0	0	0	0	Alarm when date, hours, minutes, and seconds match
1	0	0	0	0	Alarm when day, hours, minutes, and seconds match

DY/$\overline{DT}$	ALARM 2 REGISTER MASK BITS (BIT 7)			ALARM RATE
	A2M4	A2M3	A2M2	
X	1	1	1	Alarm once per minute (00 seconds of every minute)
X	1	1	0	Alarm when minutes match
X	1	0	0	Alarm when hours and minutes match
0	0	0	0	Alarm when date, hours, and minutes match
1	0	0	0	Alarm when day, hours, and minutes match

Tabla III.9.- Máscara de bits del registro de alarma

Como se puede comprobar existen varias posibilidades, como la alarma cada segundo, minuto, o cuando éstos coincidan con un determinado valor.

En la práctica se ha hecho uso de la librería *Sodaq_DS3231*, aunque ha sido necesario realizar ciertos ajustes, así como añadir un nuevo «método» para ampliar la funcionalidad, y permitir interrupciones con frecuencia horaria, pero en el minuto establecido por el usuario, puesto que por defecto solo se incluían tres posibilidades, alarma cada segundo, cada minuto y cada hora.

Por lo tanto, para obtener una interrupción cada media hora se debe hacer uso de las dos alarmas, ajustando la primera de ellas cada hora en el instante 00:00[2], y la segunda con la misma frecuencia, pero en el minuto 30:00.

[2] Notación MM (Minutos):SS (Segundos)

CAPÍTULO IV.-

TRANSMISIÓN DE DATOS

CAPÍTULO IV.- Transmisión de datos

IV.1.- Introducción

En el presente proyecto, se han hecho uso de diversas tecnologías para la transmisión de la información y comunicación entre los múltiples microcontroladores existentes.

Para que el sistema funcione correctamente es necesario que los dispositivos Esclavos se comuniquen con el Maestro mediante un método robusto y eficaz, y éste a su vez transmitir dicha información a internet.

Aunque este sistema es escalable y puede ser adaptado a diferentes situaciones y parcelas, está pensado para ser instalado en un terreno propio, cuyas medidas y colocación de los dispositivos se muestra en la siguiente imagen.

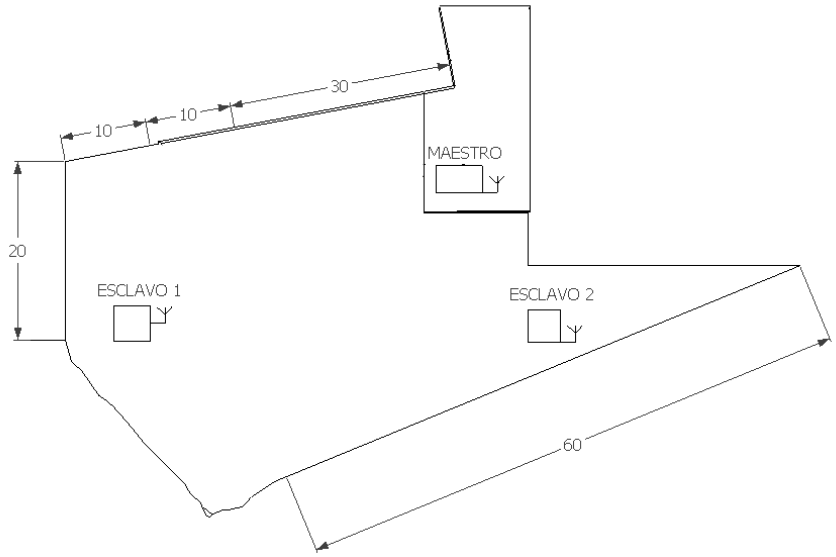


Figura IV.1.- Plano situación de los dispositivos

El controlador Maestro se encuentra situado dentro de la casa, por lo tanto, resulta más económico y eficiente realizar la conexión a Internet a través del protocolo Ethernet y mediante un cable estándar tipo UTP CAT5e al router doméstico.

Por otra parte, se optó por comunicar los Esclavos con el Maestro de forma inalámbrica, puesto que, al encontrarse muy separados entre sí, sería poco práctico realizar la conexión mediante cables, además de que otorga flexibilidad al sistema pudiendo recolocar los Arduino sin coste alguno.

IV.2.- Aspectos generales sobre la red

La topología de una red se define por la forma en la que se encuentran interconectados los diferentes dispositivos.

En este caso, la red implementada, tiene una topología de estrella (Figura IV.2), siendo el Arduino Maestro el que pide información a los Esclavos, mientras que éstos no tienen la capacidad de comunicarse entre sí, y solo responden a este último cuando lo solicita

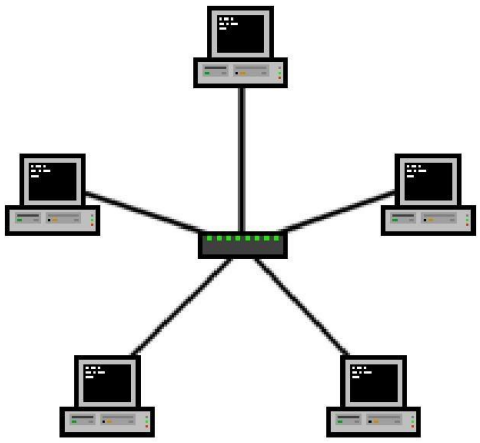


Figura IV.2.- Topología de red en estrella

Este tipo de conexión otorga una mayor facilidad de supervisión y control de la información, ya que todos los datos deben pasar necesariamente por el Maestro, el cual gestiona la redistribución de la información a los demás nodos.

Otras ventajas con las que cuenta:

- El mal funcionamiento de un nodo no afecta al resto de la red.
- Facilidad para encontrar errores y agregar nuevos dispositivos.
- Después de un fallo o reinicio, se puede realizar una reconfiguración de forma rápida.

Por otro lado, el mayor inconveniente de ésta topología, reside en uno de los aspectos que la hacen tan sencilla. El hecho de que todos los mensajes pasen por el controlador central (Maestro), hace que la red sea muy vulnerable a los posibles fallos que puedan ocurrir en este dispositivo.

IV.3.- Transmisores

Para comunicar las placas Arduino entre ellas se optó por el transceptor (transmisor y receptor) nRF24L01+ [7] (Figura IV.3) del fabricante Nordic Semiconductor, el cual está diseñado para comunicaciones inalámbricas que trabajen a baja potencia. Integra un transmisor y receptor de radiofrecuencia de 2,4 GHz, un sintetizador de RF y toda la lógica de banda base necesaria, que incluye el acelerador de protocolo hardware propio (Enhanced ShockBurst ™), que permite liberar al microcontrolador de las funciones de protocolo críticas.

Figura IV.3.- Transceptor nRF24L01+

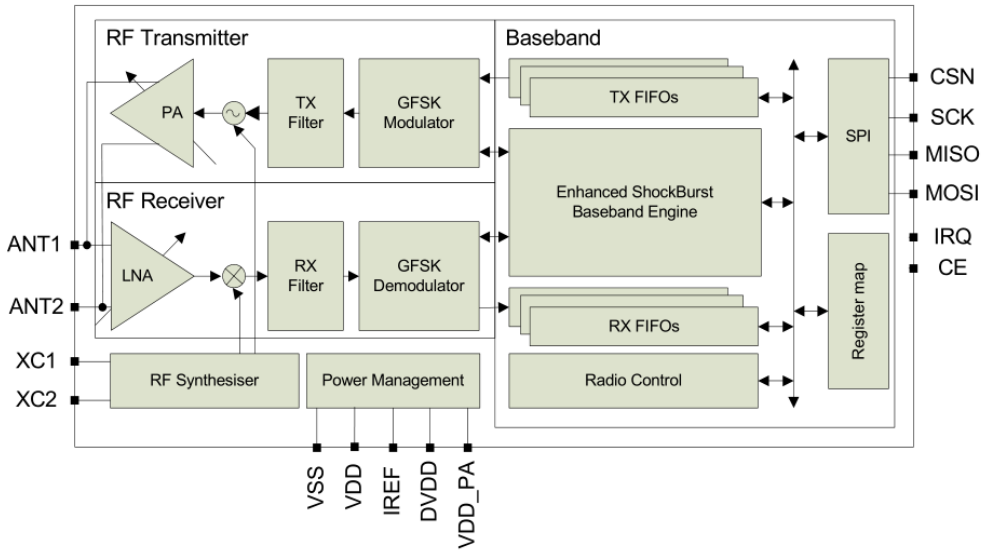


Figura IV.4.- Diagrama de bloques del nRF24L01

A continuación, se detallan las principales características de este dispositivo:

- 126 canales disponibles.
- Velocidades de transmisión configurables de 250 kbps, 1 y 2 Mbps.
- Tensión de alimentación de 1,9 a 3,6 V.
- Consumo de 11,3 mA en modo de transmisión (TX) y 13,3 mA en modo de recepción (RX).
- Disponibilidad de seis canales de datos MultiCeiver™.
- Modos de ultra-baja potencia:

  - 26 µA en el modo «Standby-I».

  - 900 nA en modo apagado.

Este transceptor hace uso de la banda ISM (*Industrial, Scientific and Medical*), en concreto en el rango de 2,400 a 2,4835 GHz y además utiliza la modulación GFSK (*Gaussian Frecuency-Shift Keying*) (Figura IV.5) que es un tipo de modulación digital en la que primero se filtran los impulsos para evitar que las altas frecuencias pasen al modulador, y luego emplean dos frecuencias distintas para codificar los datos.

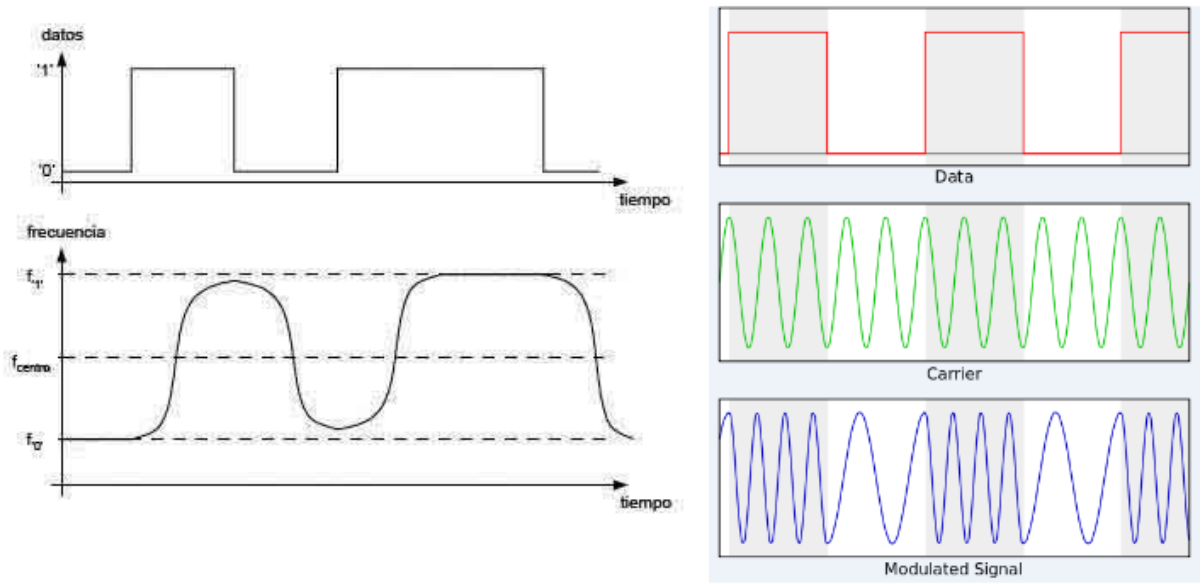


Figura IV.5.- A la izda.: señal digital filtrada con filtro gaussiano. A la dcha.: Modulación digital típica de una señal digital. (FSK)

Para configurar el transceptor tienen que modificarse los bits de su mapa de registros, a los cuales se accede mediante una interfaz SPI (*Serial Peripheral Interface*) (Figura IV.6).

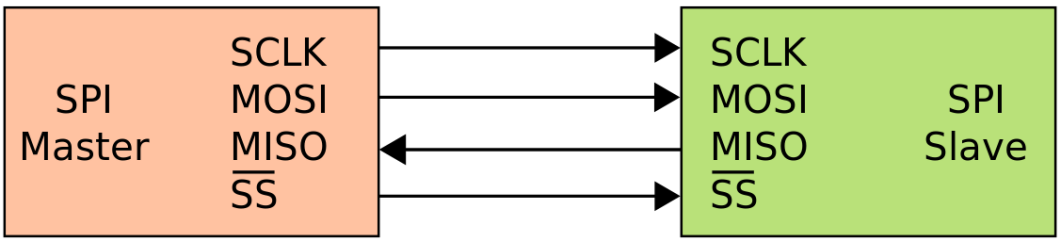


Figura IV.6.- Interfaz SPI

El bus SPI establece una comunicación síncrona mediante la señal de reloj (SCLK), los datos se transmiten a través de las líneas MOSI/MISO (Master Output/Input Slave Input/Output), la línea SS (Slave Select) sirve para seleccionar el dispositivo con el que se quiere comunicar en caso de que se encuentren varios en el bus.

El transceptor utiliza un protocolo de comunicación propio (Enhanced ShockBurst™) (Figura IV.7) que se basa en la comunicación por paquetes, los cuales pueden tener hasta un máximo de 32 bytes de datos.

Figura IV.7.- Formato de un paquete de Enhanced Shockburst ™

El protocolo gestiona automáticamente el envío y recepción de paquetes. Tiene un funcionamiento half-duplex, de manera que no puede estar enviando y recibiendo datos a la vez.

Primero un transceptor se pone en estado transmisor y envía un paquete, al terminar se pone en modo receptor.

El otro transceptor que se encuentra en modo receptor está constantemente buscando una dirección valida en las señales que recibe, una vez llega la dirección correcta comprueba que el paquete no tenga fallos mediante el código detector de errores CRC (*Cyclic Redundancy*

Check), si todo está bien desecha las cabeceras del paquete y almacena los datos (*payload*) en una memoria FIFO (*First In First Out*).

Una vez que recibe el paquete, el receptor pasa durante un breve periodo de tiempo a modo transmisor para enviar un paquete ACK (*Acknowledgment*) al transmisor y que este sepa que el receptor recibió el paquete.

Si el transmisor no recibe el paquete ACK en un tiempo determinado vuelve a reenviarlo y se repite el proceso.

IV.4.- Ethernet. Publicación en página Web

IV.4.1- Introducción

Uno de los aspectos más interesantes del presente proyecto, es la posibilidad de poder realizar un control y seguimiento del sistema de forma remota, de modo que no sea necesaria la presencia de un operador, para realizar la supervisión. Esto se consigue mediante el envío de información, mediante Ethernet, a un servidor Web creado para tal fin.

Para ello se ha hecho uso de una placa comúnmente denominada *Ethernet shield* (Figura IV.7), que se acopla simplemente insertando sus pines tipo hembra en el microcontrolador utilizado (Arduino Mega), permitiendo una rápida y cómoda conexión a Internet.

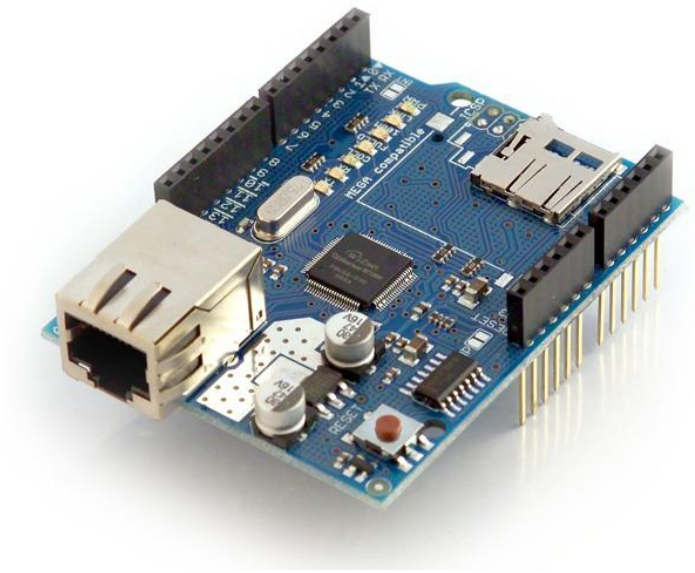


Figura IV.7.- Tarjeta Ethernet Shield

Dicha placa utiliza el chip iEthernet W5100 [8], del fabricante WIZnet Co., optimizado para dispositivos portátiles y sistemas empotrados. El uso de este modelo facilita las labores de creación *sockets de Internet*, evitando que el usuario tenga que manejar un complicado controlador de Ethernet (tiene capacidad para hasta cuatro conexiones simultáneas). Añade un procesamiento hardware para prevenir ataques de red del tipo: suplantación de identidad e inyección[3]. Además, es capaz de soportar los subestándares de comunicación 10BaseT y 100BaseTX (estándar de FastEthernet 100Mbit/s), y múltiples protocolos TCP/IP (TCP, UDP, ICMP, IPv4, ARP, IGMP, PPPoE[4]).

A continuación, se representa su diagrama de bloques interno (Figura IV.8):

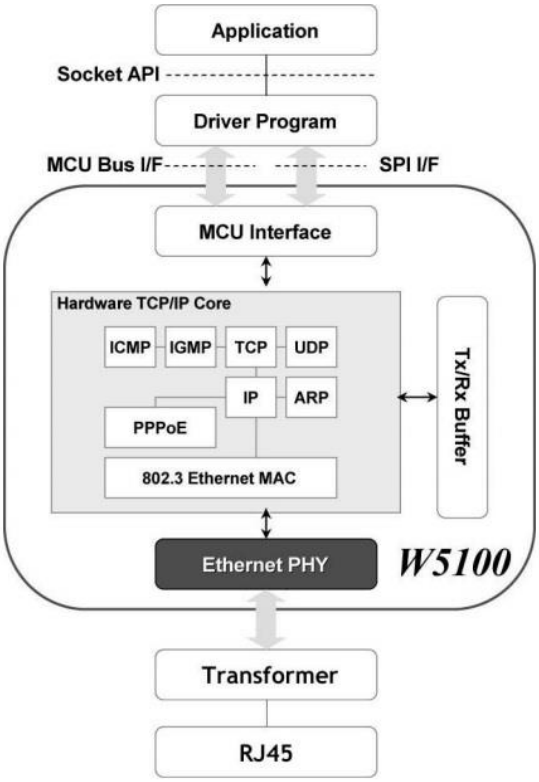


Figura IV.8.- Diagrama de bloques

Se ha utilizado la versión genérica de este dispositivo, en lugar de la oficial *Arduino Ethernet Shield*, por ofrecer igualdad de prestaciones a un precio significativamente menor. Hace uso del protocolo de conexión SPI y tiene la capacidad de obtener la alimentación eléctrica necesaria (para el shield y el microcontrolador) del propio cable de conexión de red, a lo que se conoce como «PoE» (*Power over Ethernet*).

[3] Método de infiltración de código intruso que se vale de una vulnerabilidad informática

[4] «Protocolo de Punto a Punto sobre Ethernet», implementa una capa IP sobre dos puertos Ethernet, dando la posibilidad de transferir paquetes de datos entre los dispositivos que estén conectados [14].

Existe una ranura para tarjetas micro-SD en la placa, que puede es utilizada para almacenar archivos para servir a través de la red.

El microcontrolador Arduino se comunica tanto con el W5100 como con la tarjeta SD usando el bus SPI (a través del conector ICSP). Esto son, los pines digitales 10, 50, 51, y 52 (Arduino Mega). El pin 10 se utiliza para seleccionar el W5100 y el 4 para la tarjeta SD, que no se pueden utilizar como E/S estándar. En los Mega, además, el pin 53 debe mantenerse como una salida por requerimientos de la interfaz SPI.

IV.4.2- Conceptos previos y funcionamiento

Para operar con este *shield* se han empleado las librerías estándar de Arduino: *Ethernet* y *SPI*. Una vez que se ha realizado la conexión con el router son necesarios unos pasos previos de configuración de la tarjeta de Ethernet.

Seguidamente se detallarán algunos conceptos fundamentales sobre redes que serán de utilidad más adelante:

- Dirección IP

Es un elemento esencial que identifica a la tarjeta de red de un dispositivo en concreto dentro de una red TCP/IP. Este «código» consta de cuatro grupos de dígitos separados por un punto, con unos posibles valores entre 0 y 255.

Una dirección IP puede ser asignada manualmente, de modo que se obtendría una dirección fija (denominada «IP estática»), o puede ser impuesta por un dispositivo existente en la red denominado servidor DHCP, cuya labor es conceder dichas direcciones a los otros elementos miembros de ésta, en cuyo caso se trataría de una «IP dinámica».

En concreto, para la placa de Ethernet empleada se puede utilizar cualquiera de los tipos comentados.

Aparte de la clasificación anterior, se puede realizar otra distinción: Direcciones «privadas» y «públicas». Cada dispositivo que se encuentre directamente conectado a Internet, dispone de una IP de tipo «pública», con la que son capaces de comunicarse con el resto del mundo. Estas direcciones no son ilimitadas y existe un organismo internacional denominado IANA (*Internet Assigned Numbers Authority*) encargado de

repartirlas a los organismos y entidades autorizadas como los operadores telefónicos, entre otros. Por lo tanto, un usuario doméstico no podrá disponer de una IP pública, además de que supondría un desaprovechamiento de las direcciones existentes, dada la ingente cantidad de dispositivos conectados a Internet que existen actualmente.

Como solución a este problema se emplean las «IP privadas», que sólo pueden ser utilizadas en el interior de una red local. Por lo que ahora se hace necesario un dispositivo que actúe como enlace con la red exterior (Internet). Este dispositivo se denomina «router», y es el único que dispone de una dirección IP pública.

- Máscara de red

Este parámetro indica a que red pertenece una dirección IP determinada. Un dispositivo en concreto no puede pertenecer a más de una red a la vez y solo serán capaces de comunicarse entre sí los que pertenezcan a la misma.

Al igual que en el caso de la dirección IP, la máscara de red está formada por cuatro valores separados por un punto con valores comprendidos entre 0 y 255.

- Dirección MAC

La dirección MAC se puede asimilar a un código serie, compuesto por 48 bits (en notación hexadecimal), que identifica unívocamente a la tarjeta de red de un dispositivo. Es un valor normalmente fijo y asignado por el fabricante.

Para el caso de la placa de Ethernet que nos ocupa, se puede tener un valor predefinido para la dirección MAC, pero en caso contrario se deberá asignar manualmente de modo que no coincida con la dirección de otro dispositivo existente en la red.

- Servidor de DNS

Uno de los mayores inconvenientes de las direcciones IP radica en la dificultad de memorizar dichas direcciones. Para dar solución a este problema, existen los servidores DNS («Sistema de Nombres de Dominio»), que asignan a cada IP un nombre descriptivo con el que poder identificar y conectarse a los equipos presentes en la red. Suelen ser recursos de uso público que se encuentran disponibles a través de Internet.

Cuando se realiza una búsqueda en un navegador de Internet con un nombre DNS, se envía una consulta al servidor dónde se encuentra alojada dicha información, solicitando

la dirección IP real correspondiente a ese nombre. Una vez que se obtiene se puede establecer la conexión con el sitio mediante dicha dirección.

Un ejemplo muy ilustrativo de lo comentado anteriormente se muestra en la tabla IV.1

Dirección IP	Nombre DNS
173.194.67.103	google.es
193.145.118.52	ull.es

Tabla IV.1 Ejemplo de relación entre direcciones IP y nombre DNS

Se puede comprobar la dificultad que supondría utilizar directamente las direcciones IP en el día a día para acceder a los sitios web deseados.

Después de este resumen de conceptos, se puede comenzar con la configuración del Arduino y la placa de Ethernet.

Lo primero que se debe realizar, es asignar los valores de la dirección MAC e IP. Para ello se utiliza el método incluido en la librería para tal fin:

```
Ethernet.begin(mac, ip);
```

En este caso la dirección MAC no se encontraba predefinida de fábrica, así que fue necesario asignarle una teniendo en cuenta que no coincidiera con la dirección de otro dispositivo de la red. Para almenarla se crea un «array» de 6 elementos de tipo byte.

```
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
```

Se procede de forma similar con la dirección IP. Se define una dirección estática dentro del rango permitido para la red.

```
IPAddress ip(192, 168, 0, 200);
```

Por último, se debe especificar el puerto que se va a utilizar, y por el que Arduino recibirá las peticiones entrantes de los clientes del servidor. El puerto 80 es el establecido por defecto para HTTP.

```
EthernetServer server(80);
```

A partir de este instante ya se pueden realizar tareas como crear un cliente o un servidor Web. El inconveniente es que dicho servidor sólo sería accesible desde los dispositivos que se encuentren en la misma red local, mientras que uno de los objetivos principales de este proyecto, es lograr una accesibilidad a los datos y ajustes de forma remota, desde cualquier lugar. Para ello se debe vincular la IP pública del router a la placa Arduino, de modo que los mensajes entrantes desde el exterior sean reenviados hacia ella.

Este método tiene otro problema, ya que las direcciones ofrecidas normalmente por los operadores de telefonía, son de tipo dinámicas, es decir, que cambian cada cierto tiempo y sin previo aviso, por lo que no resultaría rentable pues se debería de conocer la IP en todo momento para poder acceder al servidor.

Una forma de evitar este problema es utilizando un servicio de DNS dinámico (DDNS), como *no-ip.com* o *dyndns.com*, los cuales deben ser configurados para vincular la dirección pública del router con un nombre DNS elegido por el usuario, de forma que para acceder al servidor no sea necesario conocer la IP, y cuando ésta cambie, el servicio DDNS actualizará la vinculación automáticamente.

El último paso para lograr la accesibilidad desde el exterior es la redirección de los mensajes que llegan al router hacia el dispositivo correcto dentro de la red LAN, en este caso el Arduino. Esto se logra con la redirección de puertos («Port Forwarding»).

Después de estos ajustes, ya es posible hacer uso de la nueva funcionalidad de Ethernet que ha adquirido el Arduino Mega, aspectos en los que se entrará en detalle en un capítulo posterior.

III.5.- Periféricos

El usuario puede modificar la configuración del sistema de riego mediante un teclado y una pantalla unidos al Arduino Maestro, además estos periféricos sirven para comprobar los valores de las variables que están midiendo los Esclavos.

La pantalla es de tipo LCD (Liquid Crystal Display) monocromo que permite visualizar caracteres simples en una disposición de 4 filas por 20 columnas, estas pantallas suelen necesitar muchos pines del microcontrolador para funcionar (normalmente 16), pero la empleada en el proyecto dispone de un chip conversor paralelo-serie que hace uso del

protocolo I2C, lo que permite que sólo se tengan que usar 4 pines del Arduino (SDA, SCL, GND, VCC)

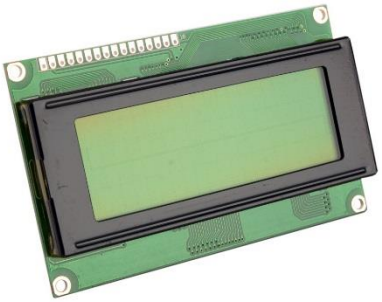


Figura IV.9.- Display LCD

Se ha usado un teclado tipo matriz de 4 filas por 4 columnas para introducir los datos. Dichas teclas son controladas mediante la librería *keypad.h*.

Figura IV.10.- Teclado

Mediante software se ha conseguido que la tecla "*" (asterisco) se pueda utilizar para escribir "." (punto), para así poder introducir decimales que serán necesarios para ajustar algunos parámetros del sistema.

CAPÍTULO V.-

ALIMENTACIÓN ELÉCTRICA, EVAPOTRANSPIRACIÓN Y RIEGO

CAPÍTULO V.- Alimentación eléctrica, evapotranspiración y riego

V.1.- Introducción

A lo largo del presente capítulo, se expondrán los aspectos más importantes referentes al método de alimentación eléctrica utilizada, un completo desarrollo del parámetro denominado evapotranspiración y, por último, la elección del tipo de electroválvula seleccionada para el sistema de riego.

V.2.- Alimentación

El sistema necesita energía eléctrica para funcionar, por lo que se analizaron distintas alternativas para alimentar las placas Arduino.

Al Arduino Maestro, puesto que se encuentra en el interior de la casa, lo más sencillo es conectarlo a la instalación eléctrica domestica mediante un simple convertidor AC/DC (Figura V.1), los cuales son bastante económicos y existe una gran variedad debido al auge de dispositivos electrónicos portátiles. Se conectaría al Arduino mediante su entrada para conectores tipo Jack de 2,1mm.

Sus características son mostradas en la tabla siguiente.

Input	110-220 V	AC: 50/60 Hz	
Output	9 V	DC	1000 mA

Tabla V.1.- Características adaptador de corriente

Figura V.1.- Adaptador de corriente

Para el caso de los Arduino Nano, que se encuentran repartidos por varios puntos del terreno, se han barajado diversas posibilidades:

Primero se consideró usar pilas, pero éstas tienen inconvenientes como su poca duración. Además, el usuario deberá estar pendiente de cuando se agotan para cambiarlas, lo cual le quita al sistema su principal beneficio, que es su automatización y el hecho de que ahorra tiempo y esfuerzo a las personas.

También existe la posibilidad de usar baterías de larga duración, pero son caras y persiste el problema de su recambio. Asimismo, sería necesaria una pareja de éstas por cada Esclavo.

Al final se llegó a la conclusión de que una buena solución estriba en alimentar los Esclavos mediante energía solar, que aunque requiera una mayor inversión inicial, se amortiza con el tiempo y mantiene las ventajas de un sistema autónomo.

Para incorporar este tipo de energía se usa una placa solar, una pequeña batería de litio y un controlador de carga (Figura V.2). La batería permite que el dispositivo alimentado continúe con su funcionamiento normal cuando la luz solar recibida por la placa no sea la óptima.

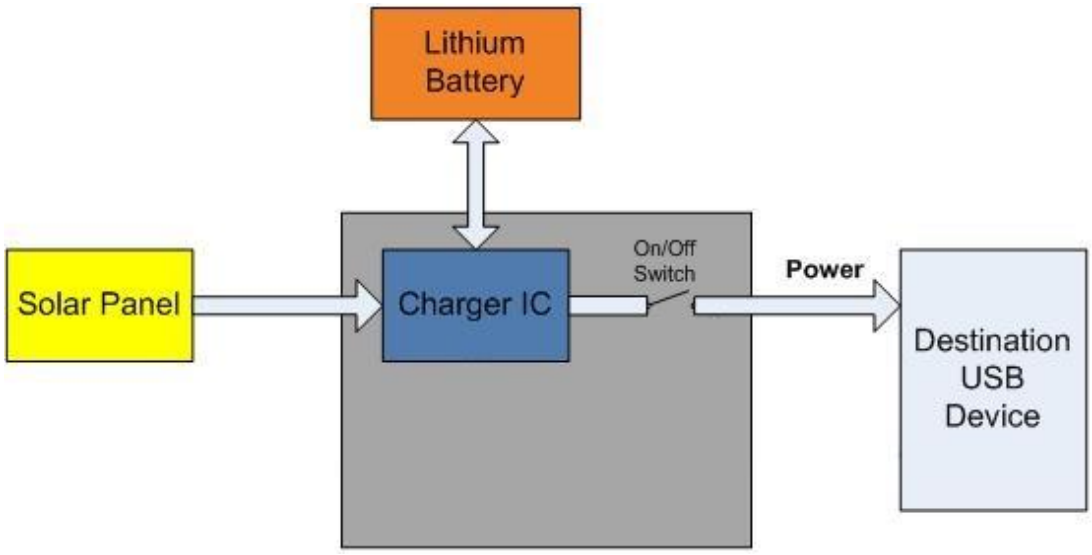


Figura V.2.- Esquema de dispositivo alimentado mediante energía solar

Se ha empleado el controlador de carga Lipo Rider [9] (Figura V.3), representado por el cuadrado gris en la figura anterior. Proporciona una salida constante de 5V gracias a su componente principal, el circuito integrado CN3065, que está especializado en ajustar la carga en sistemas de energía solar.

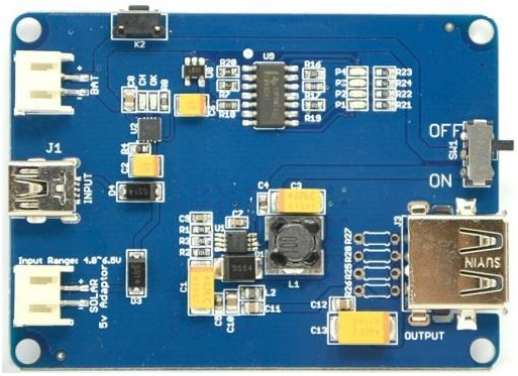


Figura V.3.- Controlador de carga

Sus características principales son las siguientes:

Parámetro	**Min**	**Typ**	**Max**
I_{in} Solar	4.8V	5.0V	6.5V
I_{charge} (R_{Iset}=3.9kΩ)	400mA	500mA	600mA
I_{supply}	0mA		1000mA
V_{batt}(R_x=0Ω)	4.2V		
$V_{destination\ USB}$	5.0V		

Tabla V.2.- Principales características del controlador de carga Lipo Rider

Las baterías utilizadas son del tipo LiPo de 3,7 V y 2000 mAh (Figura V.4), con capacidad suficiente para mantener la alimentación al Arduino Nano, en los periodos en los que no se pueda hacer uso de la energía solar.

Figura V.4.- Batería LiPo

También ha sido necesario un regulador de tensión de 3,3 V, requerido por el sensor de presión y por los transceptores. Para ello se ha empleado el regulador ajustable LM317 [10] de Texas Instruments.

Circuito implementado:

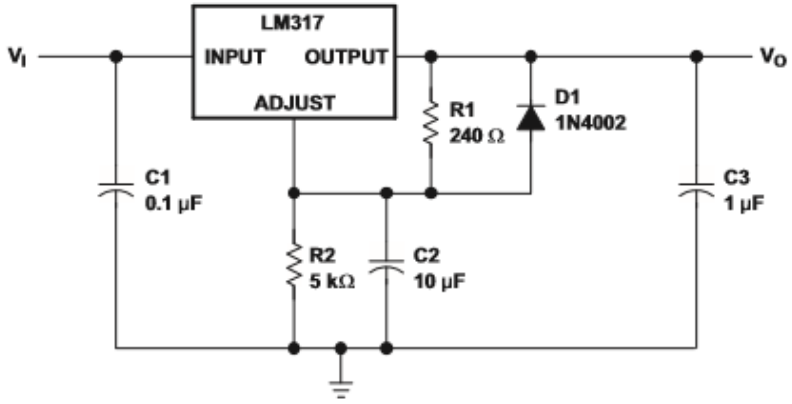


Figura V.5.- Circuito regulador de tensión

Como fuente de energía del sistema, se emplea un panel solar de silicio policristalino de 6V, 1.1 W y 200mA y de dimensiones 112 x 84 mm (Figura V.6).

Se debe utilizar una pareja de estas placas conectadas en paralelo para poder obtener un valor de corriente adecuado para el controlador de carga.

Figura V.6.- Panel solar

V.3.- Evapotranspiración

V.3.1.- Introducción y conceptos previos

En este apartado se detallará el método utilizado para determinar las necesidades hídricas del cultivo, y con ello poder implementar el algoritmo para el control del riego.

Dicho método se basa en la estimación del parámetro denominado *Evapotranspiración*, que da cuenta de la cantidad de agua que se pierde a través de la superficie del suelo por evaporación y por transpiración de las plantas.

Evaporación

La evaporación es el proceso por el cual el agua líquida se convierte en vapor de agua y se retira de la superficie evaporante. Para que ocurra este cambio de fase es necesario un aporte de energía externo, que principalmente es la radiación solar directa, y en menor grado la temperatura del aire. La diferencia entre la presión de vapor del suelo y de la atmosfera, definen la velocidad a la que ocurre este proceso. A medida que ocurre la evaporación, el aire circundante se va saturando y el proceso se ralentiza. Para que continúe es importante que exista una renovación de este aire húmedo por otro más seco, para lo que influye enormemente la velocidad del viento.

En definitiva, los principales factores que están involucrados en el proceso de la evaporación son: la radiación, la temperatura del aire, la humedad atmosférica y la velocidad del viento.

Transpiración

La transpiración se fundamenta en la pérdida mediante vaporización, del agua contenida en los tejidos vegetales. Ésta ocurre principalmente en las hojas de las plantas, y de mismo modo que en la evaporación, depende de la radiación, la temperatura ambiente, la humedad atmosférica y el viento. Otros factores involucrados son el estado de desarrollo de la planta, así como las características del suelo y del agua de riego, que determinan la facilidad con la que ésta será absorbida por las raíces.

Como se ha comprobado, estos dos fenómenos descritos anteriormente ocurren de forma simultánea y se encuentran estrechamente relacionados. En las primeras fases del desarrollo

del cultivo, el factor predominante en la pérdida de agua es la evaporación, pero con la evolución de la planta, y el desarrollo de las hojas, la transpiración cobra mayor importancia hasta convertirse en el proceso principal.

En la figura V.7 se detalla está relación entre los dos fenómenos a lo largo del crecimiento del cultivo.

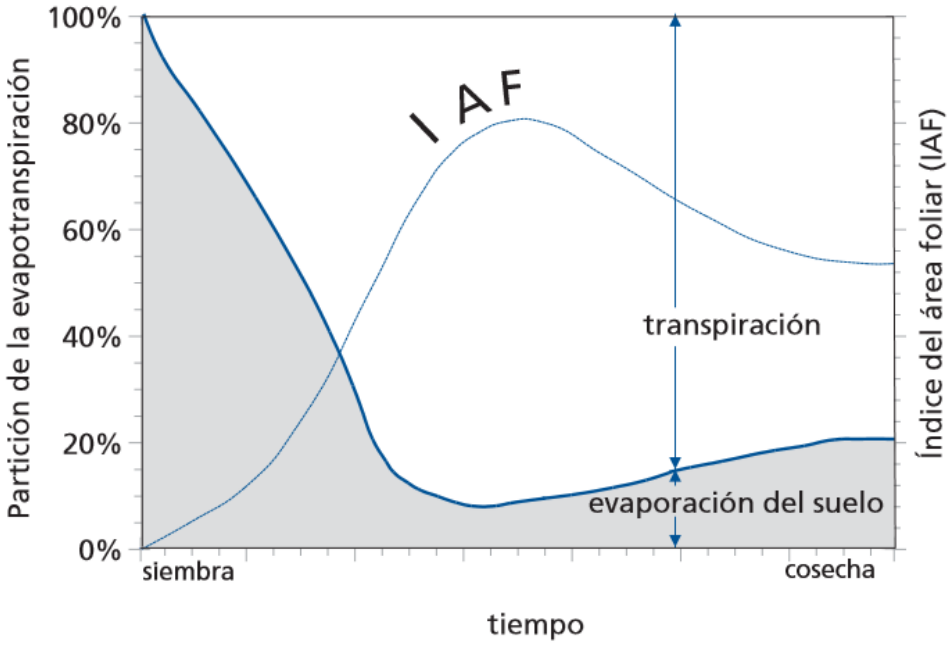


Tabla V.7.- Evolución transpiración-evaporación

Las unidades más utilizadas para expresar la evapotranspiración, son los milímetros (mm) por unidad de tiempo. Representan la altura de agua perdida en un periodo, que puede ser desde una hora hasta varios días. Otra forma de expresarlo es en m^3 por hectárea, de forma que una pérdida de 1 mm de agua equivale a 10 m^3 por hectárea y día ($\frac{m^3}{ha\ día}$).

La evapotranspiración no depende únicamente de las variables climáticas, también hay que tener en consideración otros aspectos como los factores de cultivo, entre los que se encuentra el tipo de planta, su altura, etapa de desarrollo y el manejo-condiciones ambientales como la salinidad o fertilidad de la tierra, presencia de enfermedades y parásitos, la cubierta del suelo y las prácticas de cultivo y método de riego.

Así pues, en base a los factores comentados, se tienen tres tipos de evapotranspiración: evapotranspiración del cultivo de referencia (ET_O), evapotranspiración del cultivo bajo

condiciones estándar (ET_C) y evapotranspiración del cultivo bajo condiciones no estándar ($ET_{C\ aj}$), como se puede apreciar en la figura V.8.

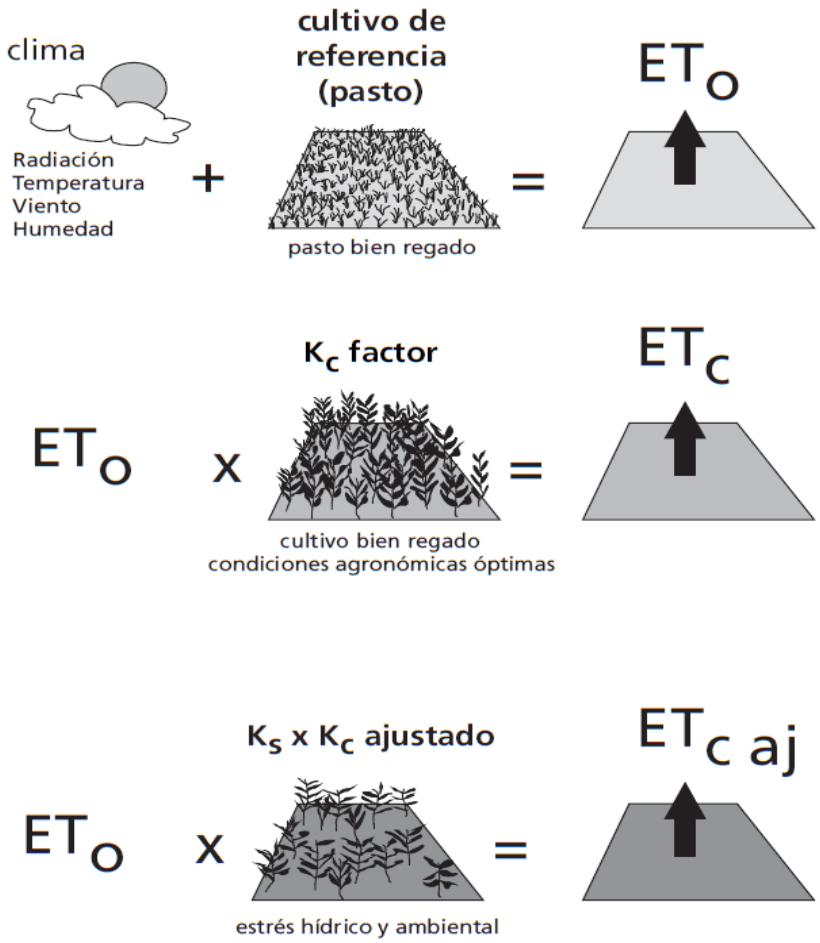


Figura V.8.- Ajustes de la evapotranspiración de referencia

Evapotranspiración del cultivo de referencia (ET_O)

Este concepto se emplea para determinar la demanda de evapotranspiración de la atmosfera de forma independiente al tipo de cultivo y condiciones de manejo. Para ello se utiliza como referencia para el cálculo una «superficie extensa de pasto verde, bien regada, de altura uniforme, creciendo activamente y dando sombra totalmente a un suelo moderadamente seco que recibe riego con una frecuencia semanal aproximadamente» [11]. De este modo se consigue que este parámetro sólo dependa de las variables climáticas, por lo que puede ser obtenido a partir de datos meteorológicos.

Evapotranspiración del cultivo bajo condiciones estándar (ET_C)

Este parámetro refleja la evapotranspiración de un cultivo en óptimas condiciones, libre de enfermedades, bien fertilizado y regado.

Las necesidades de riego de un cultivo es la diferencia entre la cantidad de agua requerida para compensar la evapotranspiración y la precipitación efectiva.

La relación existente entre ET_C y ET_O puede ser determinada experimentalmente y es conocida como Coeficiente del Cultivo (K_C), que es utilizado para relacionar ET_C con ET_O de manera que:

$$ET_C = K_C \cdot ET_O \quad (1)$$

Evapotranspiración del cultivo bajo condiciones no estándar ($ET_{C\ aj}$)

La evapotranspiración del cultivo bajo condiciones no estándar, da cuenta de la evapotranspiración en cultivos que no se encuentran bajo las condiciones óptimas, como exceso o falta de agua, presencia de plagas, o baja fertilidad del suelo, factores que pueden contribuir en un desarrollo deficiente de las plantas, y así en un distinto valor de ET_C. Por ello, para obtener el valor de la evapotranspiración ajustado se utiliza un coeficiente K_S (estrés hídrico), y el K_C modificado, de modo que reflejen las condiciones reales del cultivo.

V.3.2.- Cálculo de la evapotranspiración de referencia (ET_O)

Como se ha comentado, la ET_O se puede calcular utilizando datos meteorológicos. Para ello según la FAO (Organización de las Naciones Unidas para la Alimentación y la Agricultura), se recomienda utilizar únicamente el método de *FAO Penman-Monteith* [11], que requiere datos de radiación, temperatura del aire, humedad atmosférica y velocidad del viento.

El método utilizado está basado en una formulación anterior de la ecuación de *Penman-Monteith* desarrollada a partir de la combinación del balance energético con el método de

transferencia de masa, y ampliada posteriormente con unos factores de resistencia que describen la dificultad del flujo de vapor de la planta y suelo hacia la atmosfera.

Dicha ecuación fue revisada por un conjunto de expertos en mayo de 1990, dando lugar a una versión modificada de ésta, que aumenta la precisión respecto al método anterior, dando lugar a la nueva ecuación estándar *FAO Penman-Monteith* (2). La ventaja de esta nueva formulación, es que es capaz de proporcionar valores aceptables de la ET_O, incluso cuando no se dispone de todos los datos necesarios, ya que se puede obtener una estimación de éstos a partir de otros parámetros.

$$ET_O = \frac{0{,}408\,\Delta(R_n - \mathrm{G}) + \gamma\,\frac{900}{T+273}\,u_2\,(e_s - e_a)}{\Delta + \gamma\,(1 + 0{,}34\,u_2)} \qquad (2)$$

Dónde:

ET_O	evapotranspiración de referencia (mm dia^{-1})
R_n	radiación neta en la superficie del cultivo (MJ m^{-2} dia^{-1})
R_a	radiación extraterrestre (mm dia^{-1})
G	flujo del calor de suelo (MJ m^{-2} dia-1)
T	temperatura media del aire a 2 m de altura (°C)
U_2	velocidad del viento a 2 m de altura (m s^{-1})
e_s	presión de vapor de saturación (kPa)
e_a	presión real de vapor (kPa)
$e_s - e_a$	déficit de presión de vapor (kPa)
Δ	pendiente de la curva de presión de vapor (kPa $°C^{-1}$)
γ	constante psicométrica (kPa $°C^{-1}$)

Datos meteorológicos y atmosféricos necesarios para el cálculo:

- **Radiación solar**

La evapotranspiración depende de la cantidad de energía disponible para evaporar el agua. La radiación solar es la mayor fuente de energía que recibe el planeta y cambia según la localización y la época del año.

- **Temperatura del aire**

La radiación solar es absorbida por la atmosfera y la tierra, aumentando la temperatura del aire, que a su vez transfiere energía al cultivo.

- **Humedad del aire**

La humedad del aire tiene un papel muy relevante en el proceso de la evapotranspiración, puesto que la diferencia entre la presión de vapor de la superficie y del aire circundante es el factor determinante para la transferencia del vapor del cultivo.

- **Velocidad del viento**

El proceso de remoción de vapor depende en gran medida de la velocidad del viento. La evaporación del agua conlleva la saturación del aire circundante con lo que este proceso se va ralentizando. La existencia de un viento constante permite la renovación de ese aire por uno más seco, con capacidad de absorber más vapor.

No se tienen datos de la radiación solar y de la velocidad del viento, pero se pueden estimar utilizando los métodos propuestos por la FAO, en su publicación Nº56, de la serie Riego y Drenaje [11].

- **Estimación de la velocidad del viento**

Se parte del hecho de que «el flujo de aire sobre una región puede tener unas variaciones relativamente grandes en el transcurso de un día, pero éstas son pequeñas cuando se trata de periodos más largos» [11]. A falta de datos de la velocidad del viento, se puede realizar una estimación temporal asumiendo un valor de 2 m/s.

- **Estimación de los datos de radiación**

Según la metodología propuesta, se puede realizar una aceptable aproximación del valor de la radiación neta R_n, utilizando principalmente los datos de la temperatura mínima y máxima. Para ello se hace uso de la siguiente ecuación:

$$R_n = R_{ns} - R_{nl} \qquad (3)$$

Dónde:

R_n radiación neta [MJ m^{-2} d^{-1}]
R_{ns} radiación de onda corta [MJ m^{-2} d^{-1}]
R_{nl} radiación neta de onda larga [MJ m^{-2} d^{-1}]

Antes de poder aplicar la ecuación anterior, son necesarios una serie de pasos previos:

1. Obtener la latitud en radianes del lugar bajo estudio

En este caso, La Hidalga (Arafo), 28º 20' 02'' N, este valor se debe pasar a grados decimales y posteriormente a radianes.

2. Cálculo de la radiación extraterrestre (R_a)

Para ello se utiliza la siguiente expresión:

$$R_a = \frac{24 * 60}{\pi} G_{SC}\, d_r [\omega_S \sin\varphi \sin\delta + \cos\varphi \cos\delta \sin\omega] \qquad (4)$$

Dónde:

R_a radiación extraterrestre [MJ m-2 dia-1]
G_{sc} constante solar = 0,082 MJ m-2 min-1
d_r distancia relativa inversa Tierra-Sol (Ecuación 5)
ω_s ángulo de radiación a la puesta del sol (Ecuación 7) [rad]
φ latitud [rad]
δ declinación solar (Ecuación 6) [rad]

Como se puede observar, primero se deben obtener unos factores previos:

2.1.Distancia relativa inversa Tierra-Sol y declinación solar

Estos parámetros vienen dados por:

$$d_r = 1 + 0{,}033 * \cos\left(\frac{2\pi}{365} J\right) \qquad (5)$$

$$\delta = 0{,}409 * \sin\left(\frac{2\pi}{365} J - 1{,}39\right) \qquad (6)$$

Donde *J* representa el número del día en el total de días del año, es decir, su valor varía entre 1 y 365.

2.2. Ángulo de radiación a la puesta del sol (ω_s)

Se emplea la siguiente ecuación:

$$\omega_s = \cos^{-1}[-\tan(\varphi)\tan(\delta)] \qquad (7)$$

Una vez que se tiene R_a, se puede obtener el valor de R_s mediante la ecuación de Radiación de Hargreaves [12] (8).

$$R_S = k_{Rs}\sqrt{(T_{max} - T_{min})}\, R_a \qquad (8)$$

Dónde:

R_a	radiación extraterrestre [MJ m^{-2} d^{-1}]
T_{max}	temperatura máxima del aire [°C]
T_{min}	temperatura mínima del aire [°C]
k_{Rs}	coeficiente de ajuste (0,16 - 0,19) [$°C^{-0,5}$]

Todavía se requieren unos pasos más para realizar el cálculo de la radiación neta R_n.

3. Obtención de la Radiación solar en día despejado (R_{SO})

Para ello se utiliza la siguiente fórmula:

$$R_{SO} = (0{,}75 + 2 \cdot 10^{-5} z) R_a \qquad (9)$$

Donde *z* hace referencia a la altura sobre el nivel del mar [m].

4. Obtención de la Radiación de onda corta (R_{ns})

Viene dada por:

$$R_{ns} = (1 - \alpha) R_S \qquad (10)$$

5. Cálculo de la Radiación neta de onda larga (R_{nl})

Finalmente, el último parámetro necesario para la obtención de la radiación neta R_n, se halla mediante la siguiente expresión:

$$R_{nl} = \sigma \left[\frac{T_{max,K}^4 + T_{min,K}^4}{2}\right] \left(0{,}34 - 0{,}14\sqrt{e_a}\right) \left(1{,}35\frac{R_S}{R_{SO}} - 0{,}35\right) \qquad (11)$$

Dónde:

R_{nl}	radiación neta de onda larga [MJ m^{-2} dia^{-1}],
σ	constante de Stefan-Boltzmann [4,903 x 10^{-9} MJ K^{-4} m^{-2} dia^{-1}]
$T_{max,K}$	temperatura máxima absoluta durante un periodo de 24 horas[5]
$T_{min,K}$	temperatura mínima absoluta durante un periodo de 24 horas
e_a	presión de vapor real (Ecuación 12) [kPa],
R_s/R_{so}	radiación relativa de onda corta (valores ≤ 1,0),

[5] Temperatura expresada en grados Kelvin. K = °C + 273,16

R_s radiación solar (Ecuación 8) [MJ m^{-2} dia^{-1}]
R_{so} radiación en un día despejado (Ecuación 9) [MJ m^{-2} dia^{-1}]

5.1. Presión de vapor real (e_a)

El único valor desconocido en la ecuación anterior (11) es la Presión de Vapor Real (e_a) que se calcula a partir de la fórmula que se presenta seguidamente (12), en la que se hace uso de los valores de la humedad relativa.

$$e_a = \frac{e^o(T_{min})\frac{HR_{max}}{100} + e^o(T_{max})\frac{HR_{min}}{100}}{2} \quad (12)$$

Dónde:

e_a presión real de vapor [kPa]
$e°(T_{min})$ presión de saturación de vapor a la temperatura mínima diaria (Ecuación 13) [kPa]
$e°(T_{max})$ presión de saturación de vapor a la temperatura máxima diaria (Ecuación 13) [kPa]
HR_{max} humedad relativa máxima [%]
HR_{min} humedad relativa mínima [%]

El parámetro de la presión de saturación (e^o) se obtiene de:

$$e^o(T) = 0{,}6108 * e^{\left[\frac{17{,}27*T}{T+237{,}3}\right]} \quad (13)$$

Finalmente, después de haber obtenido todos los factores necesarios, se puede aplicar la ecuación 3 para el cálculo de la radiación neta.

El fin último de todo este proceso, es la obtención de los valores de la evapotranspiración de referencia ET_O, para ello es preciso hallar unos parámetros adicionales.

- Flujo de calor del suelo (G)

Para estimaciones diarias del ET_O, este valor se puede aproximar a cero.

- Presión de vapor de saturación (e_s)

Se obtiene como la media entre la presión de saturación de vapor a la temperatura máxima y a la temperatura mínima (13).

$$e_s = \frac{e^o(T_{\max}) + e^o(T_{min})}{2} \quad (14)$$

- Pendiente de la curva de presión de vapor (Δ)

Se halla mediante la siguiente expresión:

$$\Delta = \frac{4098 * \left[0{,}6108 * e^{\left(\frac{17{,}27*T}{T+237{,}3}\right)}\right]}{(T+237{,}3)^2} \qquad (15)$$

Para el cálculo se utiliza la temperatura media.

- Constante psicométrica (γ)

Para este parámetro existen valores de γ tabulados en función de la altitud. Para el caso en concreto que nos ocupa, la altitud sobre el nivel del mar es de 215 m.

Por lo que, realizando una simple interpolación con los valores de la tabla V.3, se puede obtener γ.

z (m)	γ kPa °C⁻¹
0	0,067
100	0,067
200	0,066
300	0,065

Tabla V.3.-Valores de la constante psicométrica para distintas altitudes

Por último, ya se está en condiciones de aplicar la fórmula de la FAO Penman-Monteith (2), para el cálculo de la ET_O.

$$ET_O = \frac{0{,}408\,\Delta(R_n - G) + \gamma\,\frac{900}{T+273}\,u_2\,(e_s - e_a)}{\Delta + \gamma\,(1 + 0{,}34\,u_2)} \qquad (2)$$

Como se ha comentado en el apartado V.3.1, la evapotranspiración del cultivo de referencia sirve de marco de comparación para diversos periodos del año y distintos lugares, proporcionando un valor estándar e independiente al tipo de cultivo y a las condiciones de manejo. Por ello se debe ajustar mediante un coeficiente K_c, que da cuenta del tipo de planta y su estado de desarrollo, para finalmente obtener el valor de la evapotranspiración bajo condiciones estándar (ET_C), utilizando para ello la expresión (1) vista con anterioridad.

A continuación, en la tabla V.4 se presentan algunos de los valores del coeficiente de cultivo K_C utilizados en el presente proyecto. Para una lista más exhaustiva acudir al Anexo 2.

Cultivo	$K_{C\ ini}$	$K_{C\ med}$	$K_{C\ fin}$
Manzanas, Cerezas, Peras	0,6	0,95	0,75
Albaricoque, Melocotón, Nectarinas	0,55	0,90	0,65
Aguacate	0,6	0,85	0,75
Cítricos	0,85	0,85	0,85
Hortalizas pequeñas (Lechugas, zanahorias, etc.)	0,7	1,05	0,95
Raíces y tubérculos	0,5	1,10	0,95
Leguminosas (judías, guisantes, etc.)	0,4	1,15	0,55

Tabla V.4.- Valores comunes de Kc

La cantidad de agua requerida para compensar la pérdida por evapotranspiración del cultivo se define como la necesidad de agua del cultivo, por lo que la necesidad hídrica neta (NH_n) básicamente representa la diferencia entre el requerimiento de agua del cultivo (ET_C) y la precipitación efectiva (PE).

$$NH_n = ET_C - PE \qquad (16)$$

Este es finalmente el parámetro que será utilizado en el sistema de control implementado.

Dicho valor viene dado en milímetros por día (mm/día), aunque también se puede expresar en litros por metro cuadrado (l/m^2), de modo que teniendo en cuenta el marco de plantación (densidad de plantas por metro cuadrado), se puede conocer la cantidad necesaria para cada una.

V.4.- Riego

Para activar las tomas de riego se usarán electroválvulas, en concreto, el modelo Rainbird LFV de bajo caudal (9V) (Figura V.9), y un relé para cada una, de modo que se pueda cambiar su estado de manera más cómoda.

Figura V.9.- Electroválvula Rainbird

Se escogió por su adaptación al sistema de riego por goteo y por el bajo nivel de tensión que se necesita para activarla. Dicha tensión se obtendrá de la alimentación solar que usan los Arduino Esclavos mediante el convertidor elevador DC-DC *U3V12F9* [13] (Figura V.10), que a partir de tensiones continuas de entrada de un mínimo de 2,5 V, es capaz de obtener 9 V de salida.

Figura V.10.- Convertidor dc-dc U3V12F9

CAPÍTULO VI.- SOFTWARE

CAPÍTULO VI.- Software

VI.1. Introducción

En este capítulo se describe el funcionamiento del software del sistema de riego, lo que incluye los programas del Arduino Maestro, los Arduinos Esclavos y la página web.

El sistema se gestiona automáticamente, de modo que el usuario sólo tiene que introducir tres parámetros: El caudal de riego que proporcionan las tomas de agua, la superficie del terreno en metros cuadrados y el parámetro Kc, que indica el tipo de planta que se va a cultivar.

Opcionalmente se pueden llevar a cabo dos tareas de forma manual: Anular la acción de riego programada para el día actual o añadir un riego extra el mismo día, seleccionando hora y duración. Estas acciones se pueden realizar, tanto desde el teclado del Arduino Maestro, como a través de la página web desde cualquier dispositivo con conexión a Internet.

Otra característica de la que dispone, es que las variables ambientales medidas son almacenadas en un archivo que puede ser descargado desde la página web junto con la hora y día en que se obtuvieron con el fin de poder analizar los datos y realizar gráficas.

VI.2.- Funcionamiento

El Arduino Maestro es el que inicia la comunicación con los Esclavos, mientras que éstos solo registran las variables ambientales cuando éste se las pide, y no toman ninguna decisión por sí mismos.

Una vez puesto en marcha, el sistema va registrando cada 30 minutos las variables de temperatura y humedad, éstas junto con la constante Kc, permiten hallar la evapotranspiración, que refleja la cantidad de agua que ha perdido el conjunto tierra-planta. Conociendo dicha cantidad de agua necesaria, las dimensiones del terreno y el caudal de la instalación se puede deducir el tiempo que tienen que estar abiertas las electroválvulas ese día para satisfacer las necesidades de las plantas.

Todos los días a las 00:15 se procederá a realizar el riego de las parcelas durante el tiempo calculado. En caso de que los sensores de lluvia se hayan activado, al menos, tres veces ese día se cancelará el riego programado. La temperatura y humedad que se registran cada 30 minutos se podrán visualizar en la página web. A continuación, se presentan los diagramas de flujo del dispositivo Maestro y los Esclavos.

VI.3.- Diagrama de flujo principal del programa

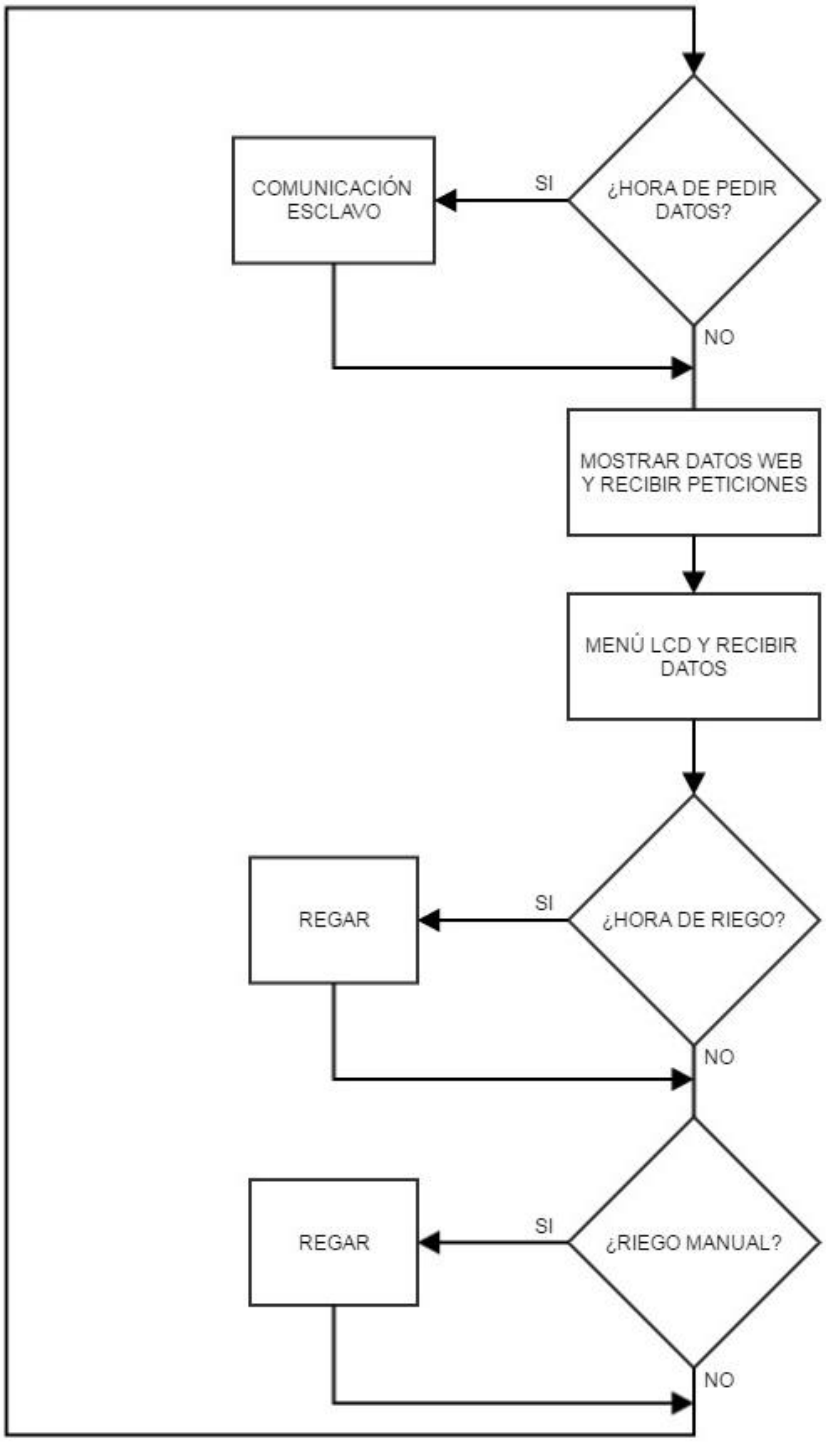


Figura VI.1.- Diagrama de flujo del controlador Maestro

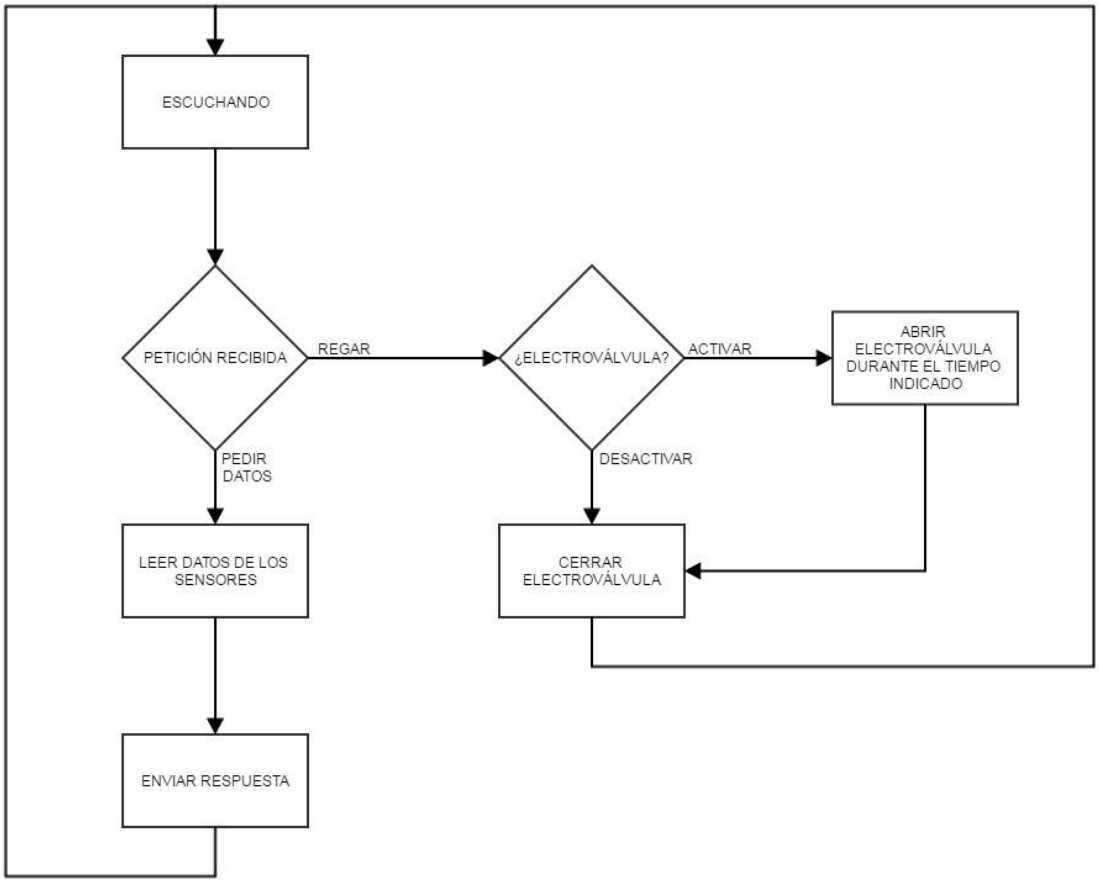


Figura VI.2.- Diagrama de flujo de los dispositivos Esclavo

VI.4. Maestro

El Arduino Maestro hace uso de un Reloj en Tiempo Real (RTC) que activa una interrupción cada 30 minutos. Cuando esto ocurre, el programa salta a una subrutina que sirve para pedirle la información de variables ambientales a los Esclavos. En ese instante el display mostrará un texto indicando que se está estableciendo una comunicación para impedir que el usuario intente interactuar con el programa en ese momento. Una vez obtenidos se almacenan y se muestran en la web.

El resto del tiempo el display mostrará la pantalla de inicio, que consta de las cuatro líneas que se aprecian en la Figura VI.3. La forma de interactuar con el dispositivo es la siguiente: Cuando una línea comienza con una letra mayúscula seguida por un punto (por

ejemplo: A. Introducir Datos) se requiere pulsar la tecla correspondiente para acceder a ese menú. Otro aspecto a tener en cuenta es el botón "*" (asterisco) del teclado, empleado para introducir el punto decimal y la tecla # (almohadilla) equivalente al «Enter», que se pulsará cuando se haya terminado de introducir los datos o cuando se desea volver al menú anterior.

Figura VI.3.- Pantalla principal

Estructura del display:

- La primera línea indica la fecha y hora.
- La segunda línea (A. Introducir Datos). Se activará cuando se pulse la letra A y en el display se visualizará lo indicado en la Figura VI.4. En esta pantalla se podrán introducir los datos Kc, caudal y superficie.

Figura VI.4.- Pantalla introducir datos

- La tercera línea (B. Riego extra). Como se puede observar en la figura siguiente, en esta opción se podrá indicar la hora y duración de una acción de riego extra que se llevará acabo el mismo día.

Figura VI.5.- Pantalla riego extra

- La cuarta línea (C. Anular riego) suprime el riego programada para ese día. Figura VI.6

Figura VI.6.- Pantalla anular riego

Por último, cuando se detecte que son las 00:00 se activará una alarma que llevará el programa a otra subrutina donde se calculará si hay que regar o no y la duración de éste, llevando acabo el riego si es oportuno.

VI.5. Esclavo

Los Esclavos están permanentemente a la escucha, y cuando reciben una orden del Maestro se detiene la radio y ejecutan sus órdenes. Estas pueden ser de dos tipos, una petición de los valores de las variables ambientales o una orden de riego.

Cuando es una petición de datos, se comienza a registrar los valores obtenidos de los sensores de temperatura, humedad y lluvia, que son devueltos al Maestro. Cuando éste los reciba de forma satisfactoria, iniciará la petición al siguiente Esclavo y así sucesivamente.

También se puede recibir una orden de riego, que dará lugar a que se active/desactive la electroválvula durante el tiempo indicado.

Es importante aclarar que todos los dispositivos reciben las peticiones del controlador Maestro, pero cada uno tiene un código de identificación único, de modo que se garantiza que sólo se obtendrá la respuesta del Esclavo solicitado.

VI.6.- Página Web

Para la realización de la página Web se han empleado diversos lenguajes de programación con el fin de lograr el aspecto gráfico y comportamiento deseados, que incluyen los siguientes:

- Lenguaje HTML para conformar la estructura básica de la página y la colocación de la información.
- Con el CSS se emplean hojas de estilo para definir la parte estética y de presentación de la página Web.
- La programación en Javascript permite crear sitios Web dinámicos, con la posibilidad de realizar una mejor interacción con el usuario. En este proyecto en concreto, se ha utilizado para el control de acceso mediante contraseña, como se puede observar en la Figura VI.7.

Se puede acceder a la página mediante la introducción del número IP del servidor en el navegador (sólo en red local), o empleando la dirección Web proporcionada por el servicio de DDNS

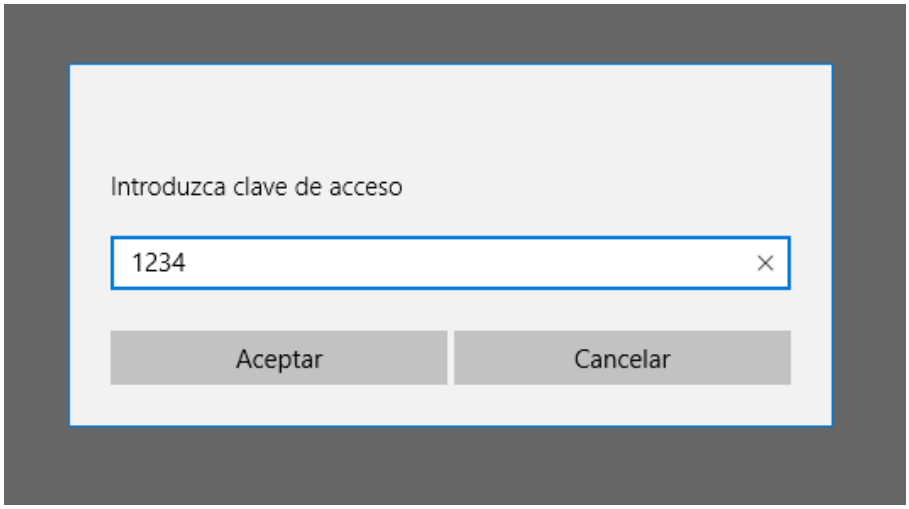


Figura VI.7.- Pantalla de acceso. Contraseña

Como se puede apreciar en la imagen anterior, se obtiene un mensaje advirtiendo de la necesidad de introducir la contraseña para acceder al servidor. Una vez que se ha especificado una clave válida, aparece un segundo mensaje informando que se procederá a redireccionar al sitio requerido.

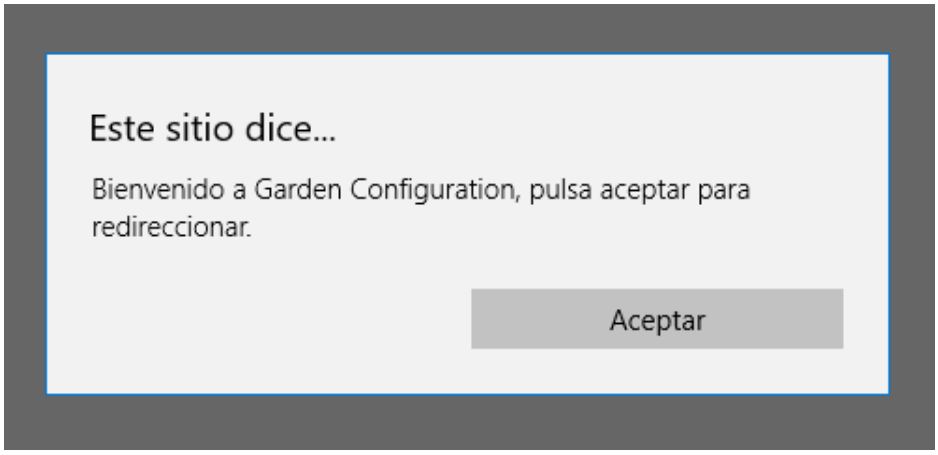


Figura VI.8.- Pantalla de acceso. Confirmación

Finalmente, en la Figura VI.9 se muestra el aspecto de la página Web implementada. Se encuentra estructurada en varias secciones, en las que se podrán visualizar los datos actuales obtenidos por los sensores, cambiar el estado del sistema, activando o desactivando el control del riego automático global, o realizar la activación manual de determinados sectores. También es posible variar el valor de la constante del cultivo (Kc), dependiente del tipo de planta y de su estado de crecimiento, y que es necesaria para el ajuste del valor de ETo (se tiene un enlace en el que es posible consultar el valor de dicha contante).

Por último, como ya se ha comentado, se dispone de la posibilidad de visualizar los datos históricos que se encuentran almacenados en la tarjeta de memoria SD de la placa de Ethernet.

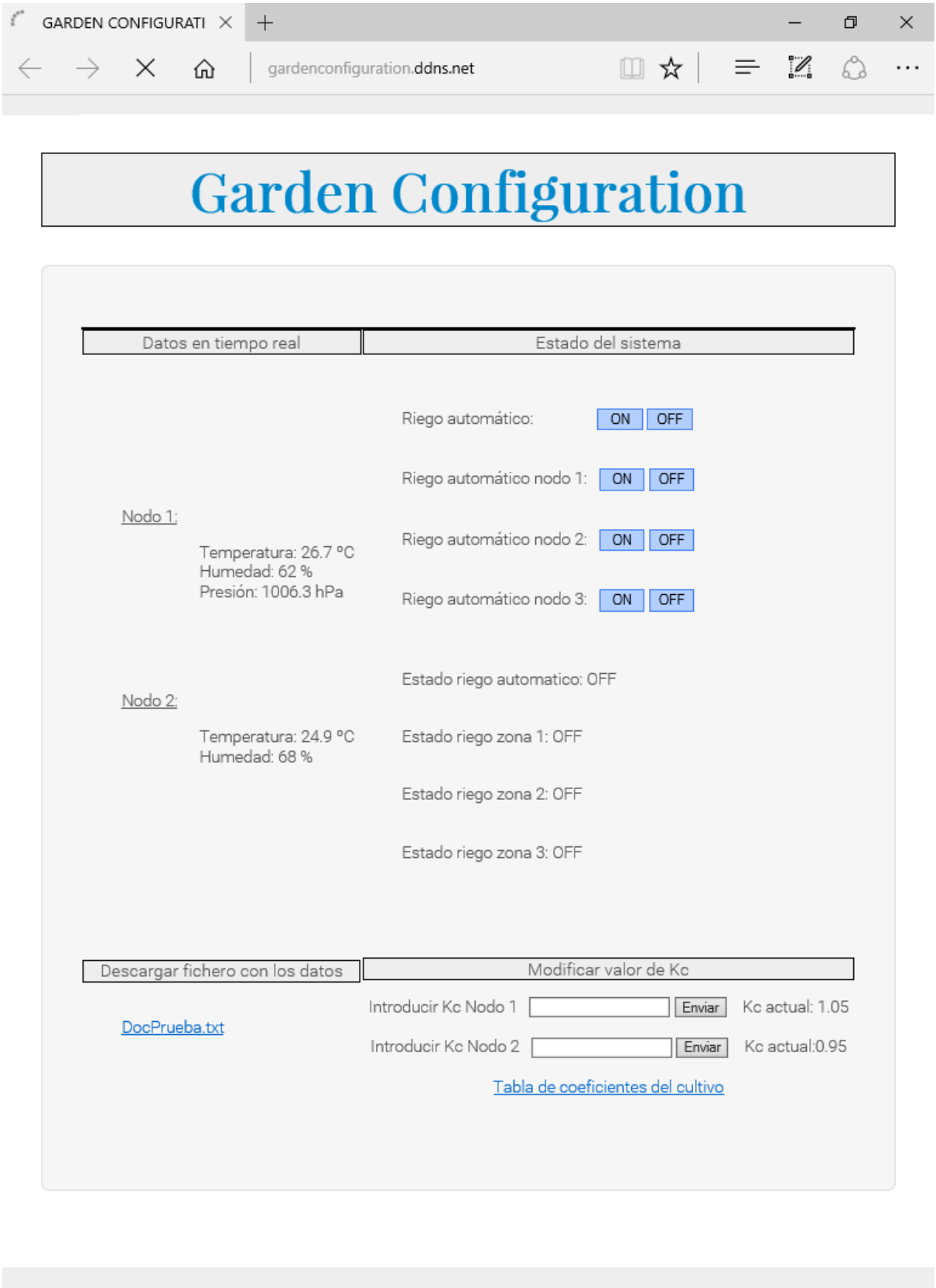


Figura VI.9.- Aspecto general de la página Web

En la siguiente figura se muestra el diagrama de flujo del programa de la página Web.

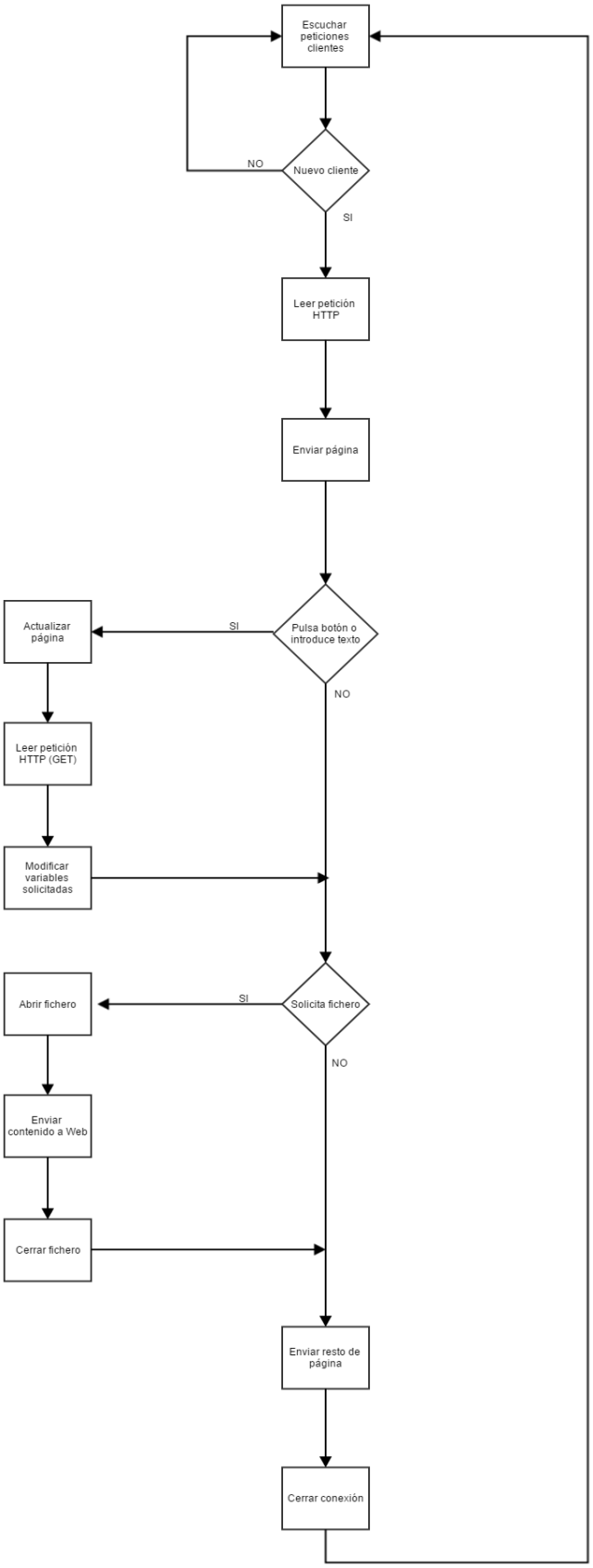


Figura 1VI.10.- Diagrama de flujo de la página Web

CAPÍTULO VII.-

RESULTADOS EXPERIMENTALES

CAPÍTULO VII.- Resultados experimentales

VII.1.- Introducción

A lo largo del presente capitulo, se expondrán los resultados obtenidos en el cálculo y estimación de la evapotranspiración, comparándolo con los valores de control extraídos de una estación agrometeorológica propiedad del MAGRAMA (Ministerio de Agricultura Alimentación y Medio Ambiente), perteneciente a la red SIAR (Sistema de Información Agroclimática para el Regadío), de modo que permitirá ofrecer una idea de la validez del modelo de predicción y de los datos obtenidos por los sensores utilizados.

Dicha estación se encuentra situada en el municipio de Güimar, a una altitud de 120 m sobre el nivel del mar y existe una separación entre ambas de aproximadamente 2 km (Figura VII.1), por lo que la información de ella obtenida puede ser utilizada para la comparación con los datos propios, dada la cercanía entre los dos puntos de medida.

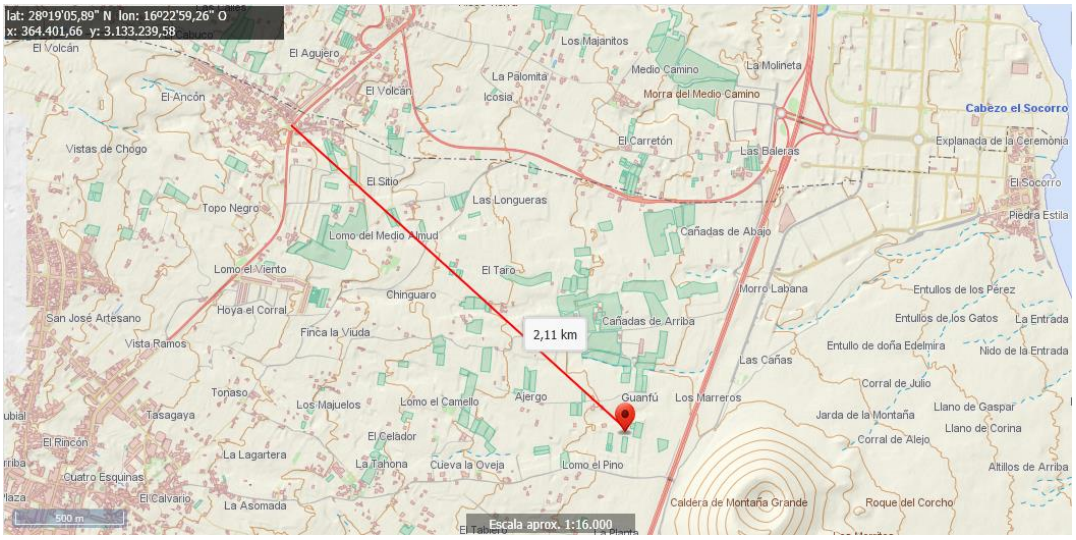


Figura VII.1.- Distancia entre las estaciones de medida

Como se puede observar en la siguiente figura, cuenta con todos los sensores necesarios para realizar un correcto cálculo de la ETo.

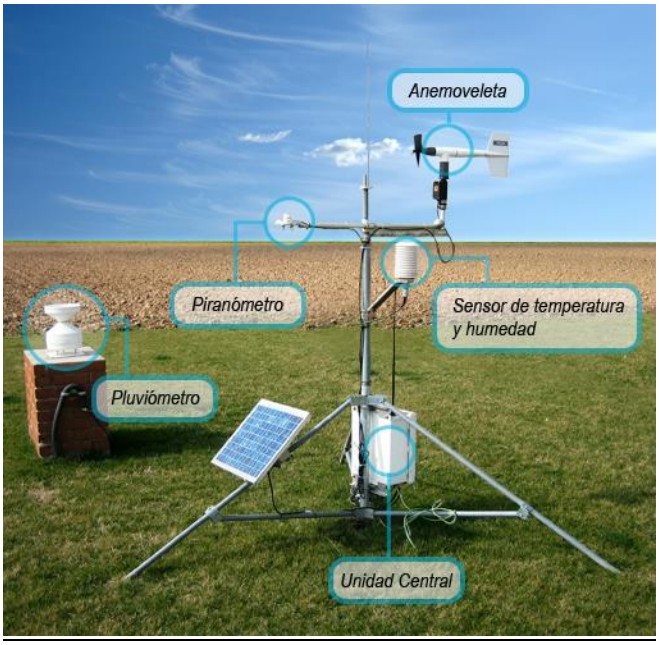


Figura VII.2.- Estación agrometeorológica del SIAR

En nuestro caso, como se ha comentado en el Capítulo VI, no se dispone de sensores con los que obtener datos de radiación y velocidad del viento, por lo que se procedió a realizar la estimación de dichos parámetros. Para la radiación neta se ha empleado el procedimiento explicado en el citado capitulo, mientras que, para la velocidad del viento, se ha seguido la recomendación de la FAO [1], y se ha tomado un valor de 2m/s. [6]

Después de realizar la toma de datos en el emplazamiento definitivo del sistema durante varios días, se han logrado valores suficientes para obtener el parámetro ETo, y realizar comparaciones con los datos oficiales.

[6] Este parámetro puede sufrir oscilaciones a lo largo del día, pero la media suele ser estable en torno a este valor.

VII.2.- Gráficas de parámetros meteorológicos

- **Comparación del valor de temperatura**

A continuación, se puede observar la evolución de la temperatura de forma horaria, entre los días 28/06/2016 y 29/06/2016 (Figura VII.3). Se ha realizado una comparación con los datos de los dos sensores propios disponibles[7] y los de la estación de la red SIAR.

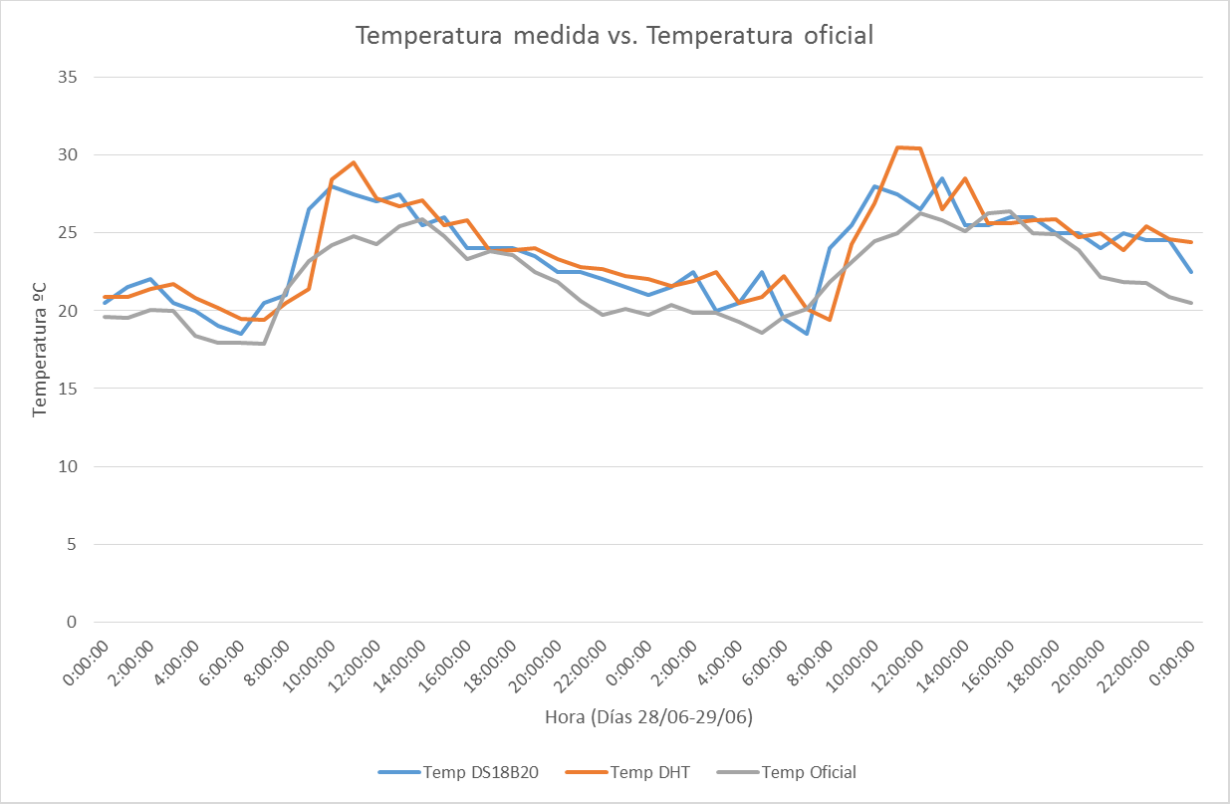


Figura VII.3.- Comparación de la temperatura

[7] Sensor de humedad y temperatura DHT22, y sensor de temperatura DS18B20

- **Comparación de la humedad relativa**

Al igual que con la temperatura, se han tomado datos horarios de la humedad relativa y se han comparado con los obtenidos de la red SIAR (Figura VII.4).

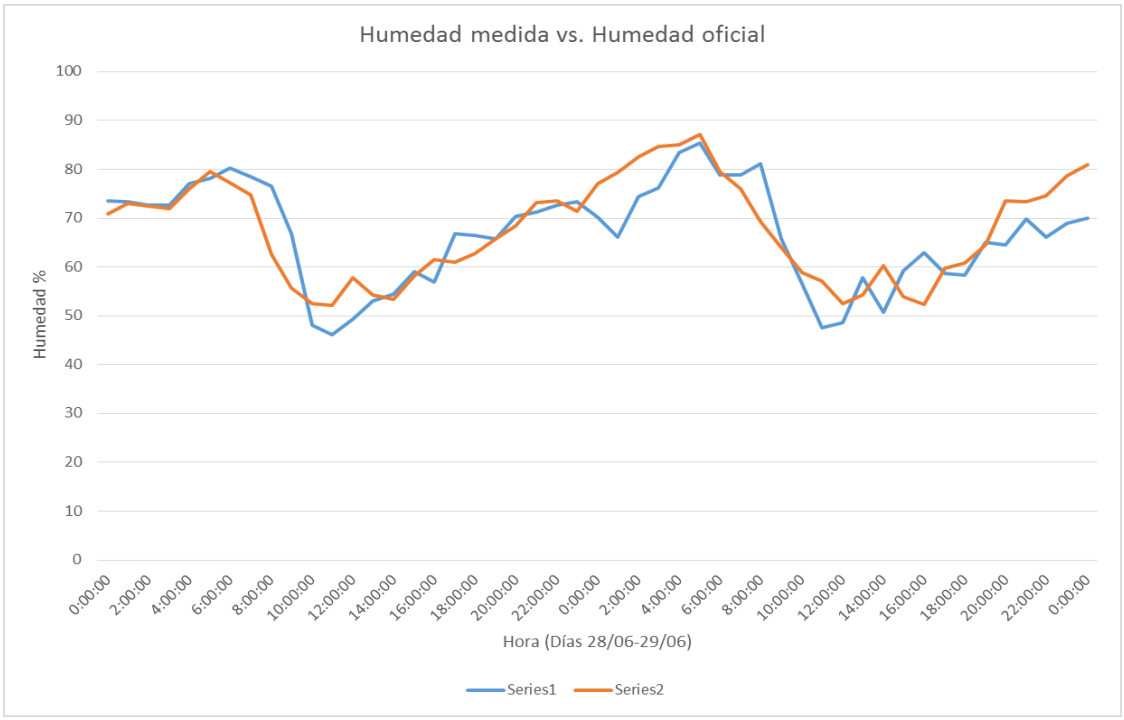


Figura VII.4.- Comparación de la humedad relativa

En las figuras anteriores se puede comprobar como los datos de los sensores propios y los del Ministerio de Medio Ambiente, siguen la misma tendencia, salvo pequeñas diferencias, que pueden ser provocadas por múltiples factores como la colocación y orientación de los sensores, y la propia separación que existe entre las dos estaciones de medida.

- **Variación Temperatura-Humedad**

En la siguiente gráfica (Figura VII.5), se puede observar la dinámica de la temperatura y la humedad relativa a lo largo de 72 horas obtenidas mediante el sistema implementado.

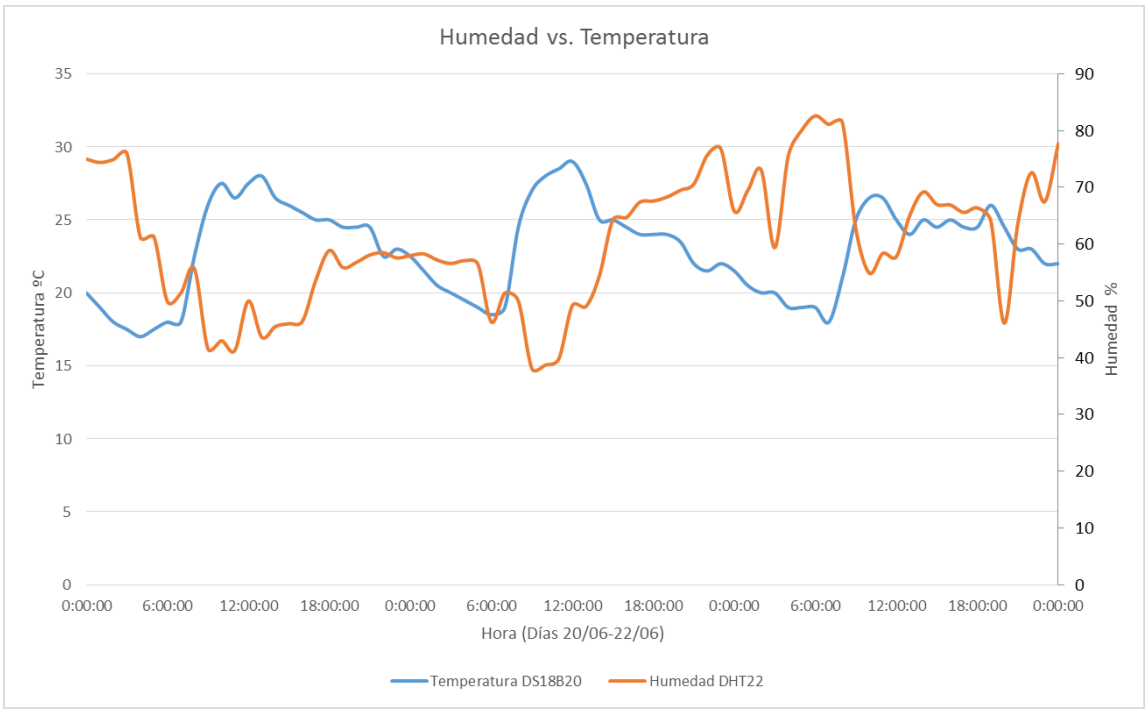


Figura VII.5.- Evolución Temperatura - Humedad relativa

VII.3.- Datos de la evapotranspiración

Se ha realizado la estimación de la radiación neta para diversos días, según la metodología propuesta por la FAO en su publicación Nº 56 de la serie «Riego y Drenaje» [11], a partir de datos como la temperatura, el día del año, o la latitud, entre otros, gracias a lo cual ha sido posible calcular el valor de la ETo.

Seguidamente, en la Tabla VII.1 se muestra el valor de la ETo medido comparado con el proporcionado por la estación agrometeorológica oficial, así como, la diferencia entre ambas medidas.

Comparación del valor de ETo oficial (MAGRAMA[1]) y medido			
ETo Oficial	Fecha	ETo Medido 2m/s	Diferencia %
5,77	20 Jun	5,613	-2,804
5,3	21 Jun	5,757	7,943
5,07	22 Jun	5,0746	0,091
4,87	28 Jun	5,445	10,568
5,73	29 Jun	5,331	-7,477

[1]Ministerio de Agricultura Alimentación y Medio Ambiente

Tabla VII.1.- Comparación de la evapotranspiración

En la Figura VII.6 se aprecia la evolución de la evapotranspiración diaria a lo largo de varios días.

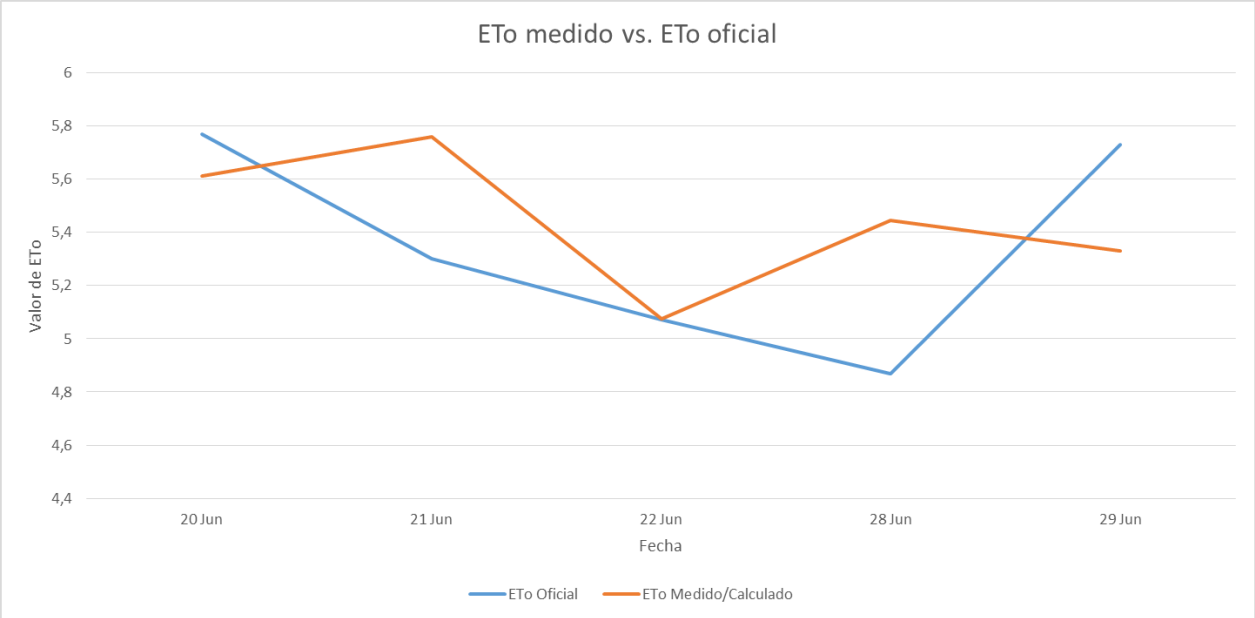


Figura VII.6.- Comparación ETo medido - ETo oficial

Como se ha comentado anteriormente, se ha tomado un valor medio estándar para la velocidad del aire (2m/s). A continuación (Tabla VII.2), se comprueba el efecto que supondría sobre el cálculo de la ETo ligeras variaciones en dicho parámetro, de modo que se pueda comprobar la importancia de éste en el proceso de la evapotranspiración.

Influencia de pequeñas variaciones en la velocidad del viento, en el valor de ETo. Comparación con el valor estandar (u = 2m/s)				
ETo estándar 2m/s	Fecha	ETo Medido 3m/s		Diferencia %
5,613	20 Jun	5,870		4,380
5,757	21 Jun	5,707		-0,882
5,0746	22 Jun	6,093		16,718
5,445	28 Jun	6,300		13,561
5,331	29 Jun	5,460		2,360
	Fecha	ETo Medido 1m/s		Diferencia %
	20 Jun	4,949		-13,408
	21 Jun	4,892		-17,696
	22 Jun	5,047		-0,545
	28 Jun	5,123		-6,290
	29 Jun	4,621		-15,374

Tabla VII.2.- Efecto de la velocidad del viento sobre el cálculo de ETo

Se observa que, dependiendo del día, estas diferencias empiezan a ser relevantes, por lo que, en futuras mejoras del presente proyecto, se encuentra incluir un anemómetro con el fin de obtener predicciones más precisas de la evapotranspiración.

CAPÍTULO VIII.- PRESUPUESTO

CAPÍTULO VIII.- Presupuesto

Descripción	Cantidad	Coste Unitario (€/u)	Coste Total (€)
Arduino Nano	2	4,9	9,8
Arduino Mega	1	13,4	13,4
Módulo Ethernet	1	16,65	16,65
Transceptores	3	2,35	7,05
Teclado	1	8	8
Pantalla LCD	1	7	7
Conversor paralelo-serie	1	2,1	2,1
Reloj tiempo real	1	2,93	2,93
Resistencias	17	0,036	0,612
Condensadores	11	0,25	2,75
Sensor de temperatura	2	3,71	7,42
Sensor de humedad	2	5,63	11,26
Sensor de presión	2	5,1	10,2
Sensor de lluvia	2	2,45	4,9
Relé	2	4,5	9
LM317 T	1	0,17	0,17
LM393	1	1,59	1,59
Cables	1	2,76	2,76
Tarjeta SD	1	4,45	4,45
Placa Solar	2	11,9	23,8
Controlador de carga	2	16,6	33,2
Batería	2	17,6	35,2
Total Coste Materiales			214,23

Concepto	Cantidad (h)	Coste Unitario(€/u)	Coste Total (€)
Tiempo de Análisis	120	65	7800
Tiempo de Codificación	300	70	21000
Tiempo de Implementación	40	35	1400
Tiempo de Documentación	60	20	1200
Total Costes Mano de Obra			31400

Costes Total (€)	
Total Costes Materiales (MT)	214,24
Total Costes Mano de Obra (MO)	31400
Gastos Generales: 6% (MT+MO)	1896,85
Beneficio Industrial: 13% (MT+MO)	4109,85
Coste Total del Proyecto	37620,94

APORTACIONES Y CONCLUSIONES

Aportaciones y conclusiones.

El presente trabajo ha tenido como objetivo el diseño e implementación de un sistema electrónico, que haciendo uso de microcontroladores y enlace hertziano, ha posibilitado:

1.- Generar una red de puntos de medida (denominados Esclavos), cada uno de los cuales está gestionado por un microcontrolador Arduino Nano con procesador ATmega 328, que posibilitan la medida de parámetros como la Temperatura, Humedad relativa y Presión atmosférica.

2.- Los Esclavos se comunican mediante enlace hertziano con el llamado Maestro, que haciendo uso de un microcontrolador Arduino Mega (ATmega 2560), es el encargado de controlar a los anteriores y servir de interfaz con el usuario con el que se comunica a través de un display/teclado e Internet.

3.- Para esto último, el Maestro hace uso de una tarjeta Ethernet lo que posibilita acceder a los datos adquiridos o realizar labores de control desde cualquier punto.

4.- Los datos de T, H y P permiten, junto con Kc, determinar la evapotranspiración y, con ello, estimar los tiempos de riego.

5.- Se ha dotado al sistema de alimentación mediante energía fotovoltaica, lo que contribuye a que los puntos de medida sean fácilmente reubicados en otras posiciones.

BIBLIOGRAFÍA

Bibliografía

[1] Dallas Semiconductor (Maxim), [En línea]. Available: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

[2] S. Laserna, «AgroES,» [En línea]. Available: http://www.agroes.es/agricultura/el-suelo/143-temperatura-del-suelo-agricultura. [Último acceso: 2016].

[3] Aosong Electronics , [En línea]. Available: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf.

[4] Wikipedia, «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Presi%C3%B3n_atmosf%C3%A9rica. [Último acceso: Junio 2016].

[5] Bosch Sensortec , [En línea]. Available: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BMP180-DS000-121.pdf.

[6] Dallas Semiconductor, [En línea]. Available: https://datasheets.maximintegrated.com/en/ds/DS3231.pdf.

[7] Nordic Semiconductor, [En línea]. Available: https://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01P.

[8] WIZnet, [En línea]. Available: http://www.wiznet.co.kr/wp-content/uploads/wiznethome/Chip/W5100/Document/W5100_Datasheet_v1.2.6.pdf.

[9] CONSONANCE, [En línea]. Available: http://lib.chipdip.ru/164/DOC001164905.pdf.

[10] Texas Instruments, [En línea]. Available: http://www.ti.com/lit/ds/symlink/lm317.pdf.

[11] FAO, [En línea]. Available: http://www.fao.org/3/a-x0490s.pdf. [Último acceso: Junio 2016].

[12] G. Hargreaves y Z. Samani, «Estimation of potential evapotranspiration,» *Journal of Irrigation and Drainage Division,* 1982.

[13] Pololu, [En línea]. Available: https://www.pololu.com/product/2116.

[14] C. Valero, «ADSLZone,» Junio 2016. [En línea]. Available: http://www.adslzone.net/adsl_pppoe.html.

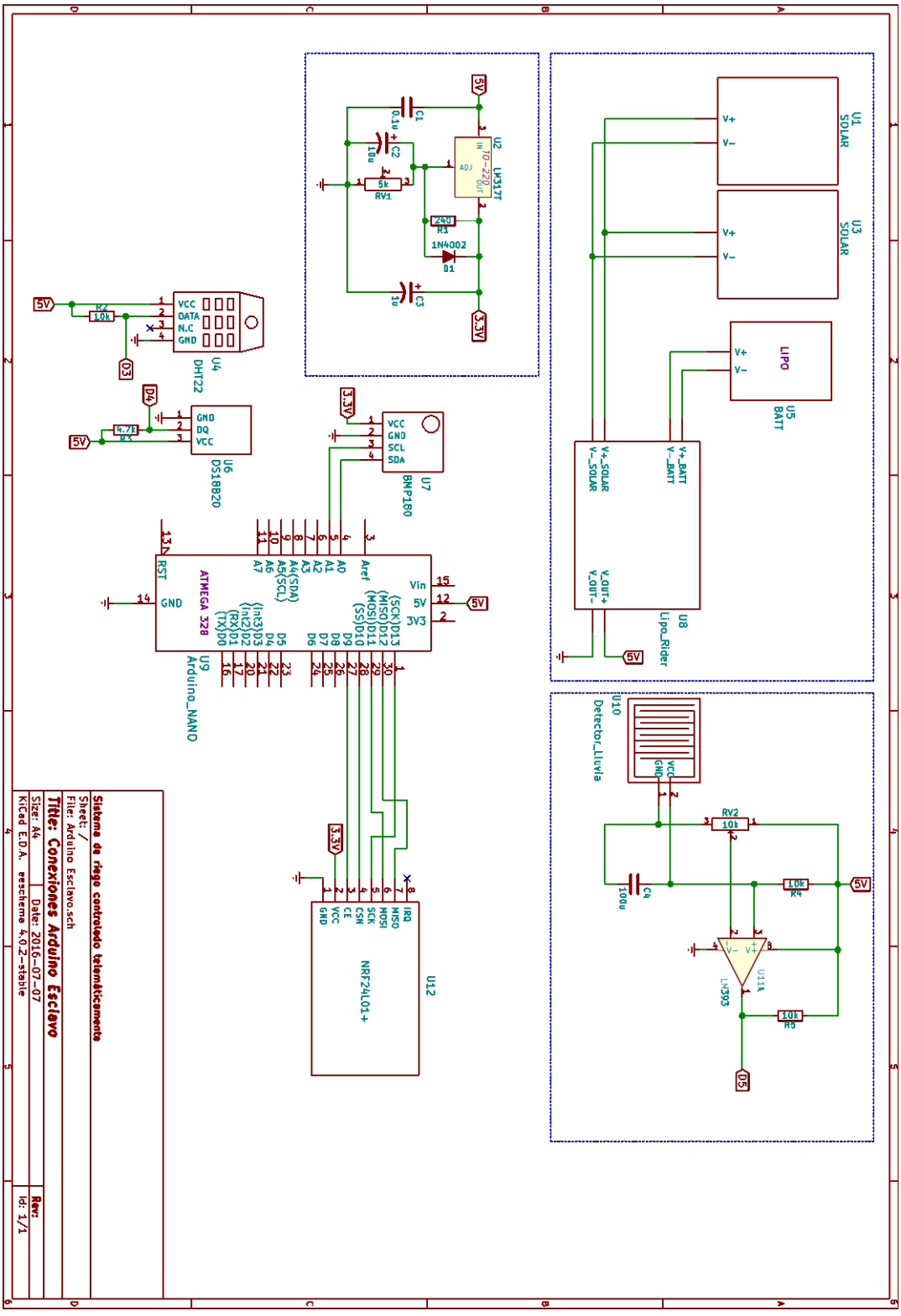
[15] L. M. S. Pérez, «Agro Cabildo,» Cabildo de Tenerife, [En línea]. Available: http://www.agrocabildo.org/publica/analisisclimatico/evapotrans2008.pdf. [Último acceso: Junio 2016].

ANEXOS

Anexos

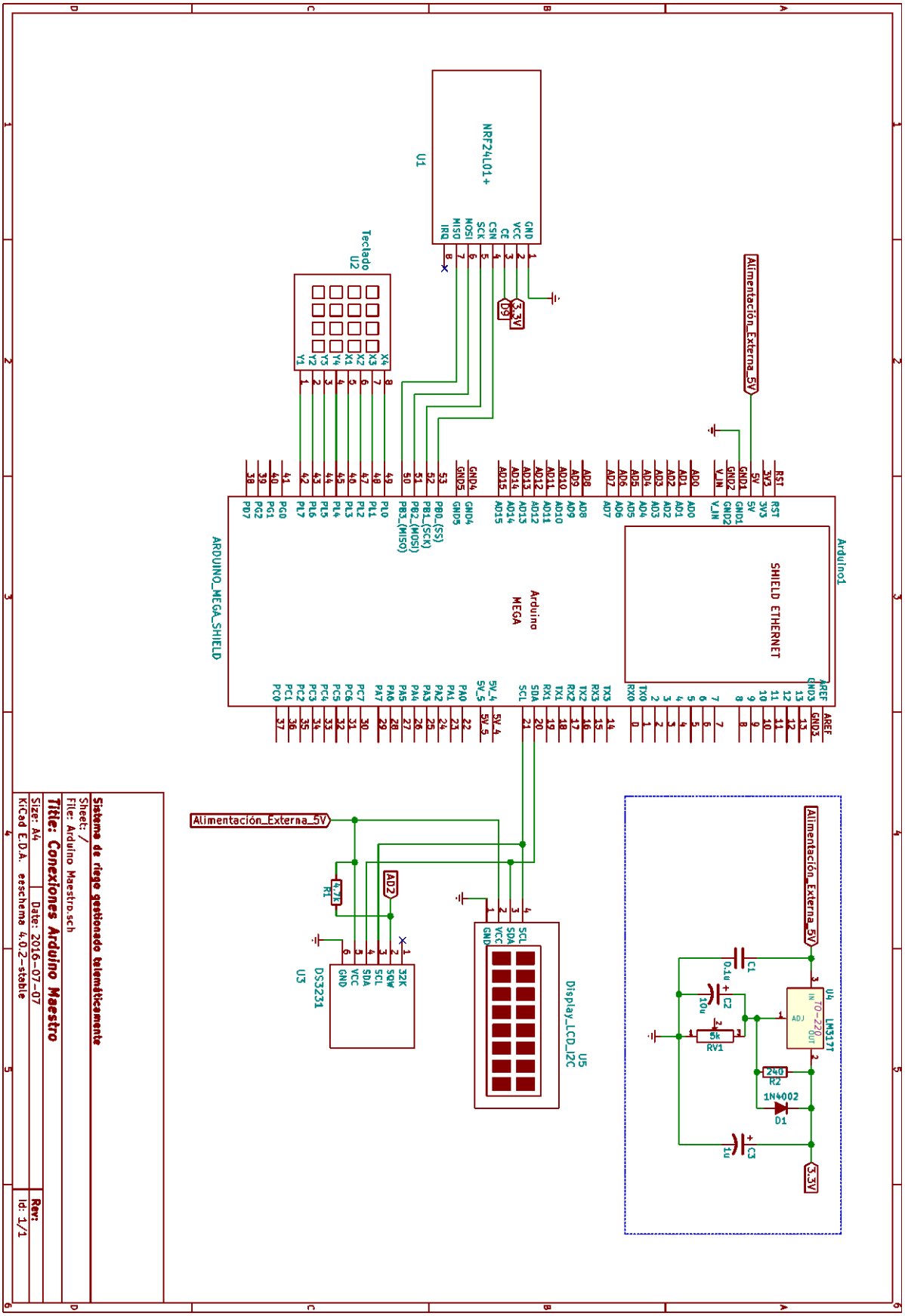
Anexo I.- Esquema y conexiones

Dispositivos Esclavos



Esquema de conexiones de los dispositivos Esclavos

Dispositivo Maestro



Esquema de conexiones del dispositivo Maestro

Anexo II.- Datasheets

II.1 Datasheet DS18B20

II.2 Datasheet DS3231

II.3 Datasheet DHT22

II.4 Datasheet BMP180

II.5 Datasheet nRF24L01+

II.6 Datasheet CN3065

II.7 Datasheet ATmega328

II.8 Datasheet ATmega2560

II.9 Datasheet WIZnet W5100

II.10 Datasheet LM393

II.11 Datasheet LM317

II.1 Datasheet DS18B20

DS18B20

Programmable Resolution 1-Wire Digital Thermometer

General Description

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

Applications

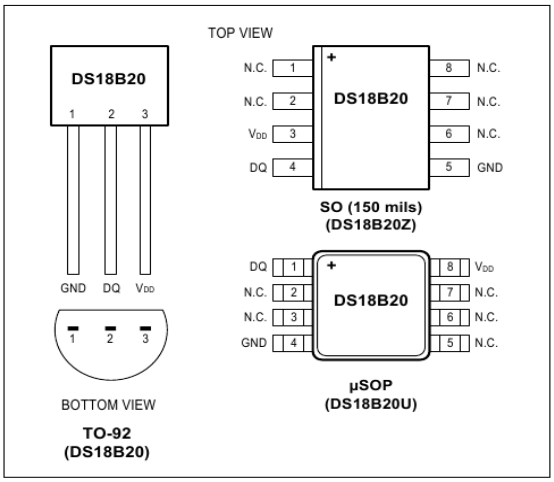
- Thermostatic Controls
- Industrial Systems
- Consumer Products
- Thermometers
- Thermally Sensitive Systems

Ordering Information ***appears at end of data sheet.***

Benefits and Features

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
  - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
  - ±0.5°C Accuracy from -10°C to +85°C
  - Programmable Resolution from 9 Bits to 12 Bits
  - No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
  - Each Device Has a Unique 64-Bit Serial Code Stored in On-Board ROM
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages

Pin Configurations



19-7487; Rev 4; 1/15

maxim integrated™

Absolute Maximum Ratings

Voltage Range on Any Pin Relative to Ground ....-0.5V to +6.0V
Operating Temperature Range..... -55°C to +125°C
Storage Temperature Range ..... -55°C to +125°C
Solder Temperature .....Refer to the IPC/JEDEC J-STD-020 Specification.

These are stress ratings only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

DC Electrical Characteristics

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Supply Voltage	V_{DD}	Local power (Note 1)		+3.0		+5.5	V
Pullup Supply Voltage	V_{PU}	Parasite power	(Notes 1, 2)	+3.0		+5.5	V
		Local power		+3.0		V_{DD}	
Thermometer Error	t_{ERR}	-10°C to +85°C	(Note 3)			±0.5	°C
		-55°C to +125°C				±2	
Input Logic-Low	V_{IL}	(Notes 1, 4, 5)		-0.3		+0.8	V
Input Logic-High	V_{IH}	Local power	(Notes 1,6)	+2.2		The lower of 5.5 or V_{DD} + 0.3	V
		Parasite power		+3.0			
Sink Current	I_L	$V_{I/O}$ = 0.4V		4.0			mA
Standby Current	I_{DDS}	(Notes 7, 8)			750	1000	nA
Active Current	I_{DD}	V_{DD} = 5V (Note 9)			1	1.5	mA
DQ Input Current	I_{DQ}	(Note 10)			5		µA
Drift		(Note 11)			±0.2		°C

Note 1: All voltages are referenced to ground.
Note 2: The Pullup Supply Voltage specification assumes that the pullup device is ideal, and therefore the high level of the pullup is equal to V_{PU}. In order to meet the V_{IH} spec of the DS18B20, the actual supply rail for the strong pullup transistor must include margin for the voltage drop across the transistor when it is turned on; thus: $V_{PU\_ACTUAL} = V_{PU\_IDEAL} + V_{TRANSISTOR}$.
Note 3: See typical performance curve in Figure 1.
Note 4: Logic-low voltages are specified at a sink current of 4mA.
Note 5: To guarantee a presence pulse under low voltage parasite power conditions, V_{ILMAX} may have to be reduced to as low as 0.5V.
Note 6: Logic-high voltages are specified at a source current of 1mA.
Note 7: Standby current specified up to +70°C. Standby current typically is 3µA at +125°C.
Note 8: To minimize I_{DDS}, DQ should be within the following ranges: GND ≤ DQ ≤ GND + 0.3V or V_{DD} – 0.3V ≤ DQ ≤ V_{DD}.
Note 9: Active current refers to supply current during active temperature conversions or EEPROM writes.
Note 10: DQ line is high (“high-Z” state).
Note 11: Drift data is based on a 1000-hour stress test at +125°C with V_{DD} = 5.5V.

II.2 Datasheet DS3231

DS3231

Extremely Accurate I²C-Integrated RTC/TCXO/Crystal

General Description

The DS3231 is a low-cost, extremely accurate I²C real-time clock (RTC) with an integrated temperature-compensated crystal oscillator (TCXO) and crystal. The device incorporates a battery input, and maintains accurate timekeeping when main power to the device is interrupted. The integration of the crystal resonator enhances the long-term accuracy of the device as well as reduces the piece-part count in a manufacturing line. The DS3231 is available in commercial and industrial temperature ranges, and is offered in a 16-pin, 300-mil SO package.

The RTC maintains seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. Two programmable time-of-day alarms and a programmable square-wave output are provided. Address and data are transferred serially through an I²C bidirectional bus.

A precision temperature-compensated voltage reference and comparator circuit monitors the status of V_{CC} to detect power failures, to provide a reset output, and to automatically switch to the backup supply when necessary. Additionally, the $\overline{RST}$ pin is monitored as a pushbutton input for generating a µP reset.

Benefits and Features

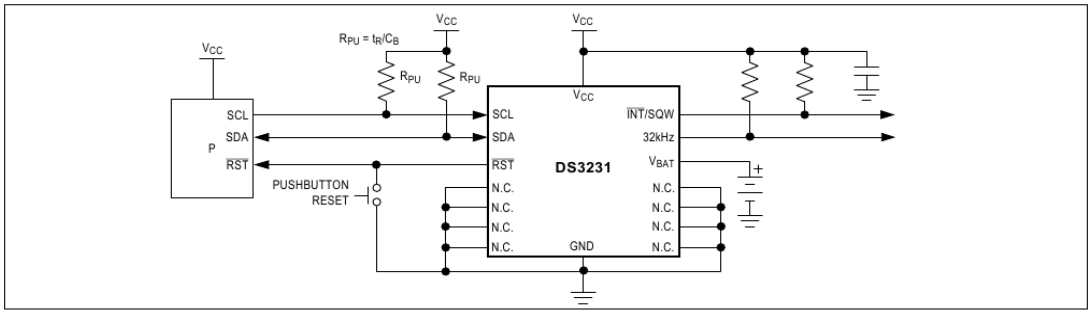
- Highly Accurate RTC Completely Manages All Timekeeping Functions
  - Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year, with Leap-Year Compensation Valid Up to 2100
  - Accuracy ±2ppm from 0°C to +40°C
  - Accuracy ±3.5ppm from -40°C to +85°C
  - Digital Temp Sensor Output: ±3°C Accuracy
  - Register for Aging Trim
  - $\overline{RST}$ Output/Pushbutton Reset Debounce Input
  - Two Time-of-Day Alarms
  - Programmable Square-Wave Output Signal
- Simple Serial Interface Connects to Most Microcontrollers
  - Fast (400kHz) I²C Interface
- Battery-Backup Input for Continuous Timekeeping
  - Low Power Operation Extends Battery-Backup Run Time
  - 3.3V Operation
- Operating Temperature Ranges: Commercial (0°C to +70°C) and Industrial (-40°C to +85°C)
- Underwriters Laboratories® (UL) Recognized

Applications

- Servers
- Telematics
- Utility Power Meters
- GPS

Ordering Information and Pin Configuration appear at end of data sheet.

Typical Operating Circuit



Underwriters Laboratories is a registered certification mark of Underwriters Laboratories Inc.

19-5170; Rev 10; 3/15

maxim integrated™

Absolute Maximum Ratings

Voltage Range on Any Pin Relative to Ground ....-0.3V to +6.0V
Junction-to-Ambient Thermal Resistance (θ_{JA}) (Note 1)73°C/W
Junction-to-Case Thermal Resistance (θ_{JC}) (Note 1) ....23°C/W
Operating Temperature Range
DS3231S .....0°C to +70°C
DS3231SN..... -40°C to +85°C
Junction Temperature.....+125°C
Storage Temperature Range..... -40°C to +85°C
Lead Temperature (soldering, 10s) .....+260°C
Soldering Temperature (reflow, 2 times max) .....+260°C
(see the *Handling, PCB Layout, and Assembly* section)

Note 1: Package thermal resistances were obtained using the method described in JEDEC specification JESD51-7, using a four-layer board. For detailed information on package thermal considerations, refer to **www.maximintegrated.com/thermal-tutorial**.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Recommended Operating Conditions

(T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 2, 3)

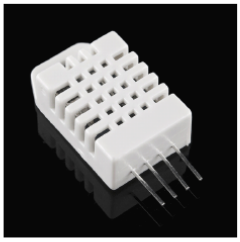
PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V_{CC}		2.3	3.3	5.5	V
	V_{BAT}		2.3	3.0	5.5	V
Logic 1 Input SDA, SCL	V_{IH}		0.7 x V_{CC}		V_{CC} + 0.3	V
Logic 0 Input SDA, SCL	V_{IL}		-0.3		0.3 x V_{CC}	V

Electrical Characteristics

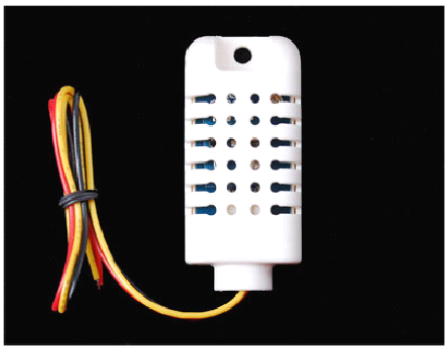
(V_{CC} = 2.3V to 5.5V, V_{CC} = Active Supply (see Table 1), T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Typical values are at V_{CC} = 3.3V, V_{BAT} = 3.0V, and T_A = +25°C, unless otherwise noted.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Active Supply Current	I_{CCA}	(Notes 4, 5)	V_{CC} = 3.63V			200	µA
			V_{CC} = 5.5V			300	
Standby Supply Current	I_{CCS}	I²C bus inactive, 32kHz output on, SQW output off (Note 5)	V_{CC} = 3.63V			110	µA
			V_{CC} = 5.5V			170	
Temperature Conversion Current	$I_{CCSCONV}$	I²C bus inactive, 32kHz output on, SQW output off	V_{CC} = 3.63V			575	µA
			V_{CC} = 5.5V			650	
Power-Fail Voltage	V_{PF}			2.45	2.575	2.70	V
Logic 0 Output, 32kHz, $\overline{INT}$/SQW, SDA	V_{OL}	I_{OL} = 3mA				0.4	V
Logic 0 Output, $\overline{RST}$	V_{OL}	I_{OL} = 1mA				0.4	V
Output Leakage Current 32kHz, $\overline{INT}$/SQW, SDA	I_{LO}	Output high impedance		-1	0	+1	µA
Input Leakage SCL	I_{LI}			-1		+1	µA
$\overline{RST}$ Pin I/O Leakage	I_{OL}	$\overline{RST}$ high impedance (Note 6)		-200		+10	µA
V_{BAT} Leakage Current (V_{CC} Active)	I_{BATLKG}				25	100	nA

II.3 Datasheet DHT22



Standard AM2302/DHT22



AM2302/DHT22 with big case and wires

Digital relative humidity & temperature sensor AM2302/DHT22

1. Feature & Application:

*High precision
*Capacitive type
*Full range temperature compensated
*Relative humidity and temperature measurement
*Calibrated digital signal
*Outstanding long-term stability
*Extra components not needed
*Long transmission distance, up to 100 meters
*Low power consumption
*4 pins packaged and fully interchangeable

2. Description:

AM2302 output calibrated digital signal. It applys exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability. Its sensing elements is connected with 8-bit single-chip computer.

Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

Small size & low consumption & long transmission distance(100m) enable AM2302 to be suited in all kinds of harsh application occasions. Single-row packaged with four pins, making the connection very convenient.

3. Technical Specification:

Model	AM2302	
Power supply	3.3-5.5V DC	
Output signal	digital signal via 1-wire bus	
Sensing element	Polymer humidity capacitor	
Operating range	humidity 0-100%RH;	temperature -40~80Celsius
Accuracy	**humidity +-2%RH**(Max +-5%RH);	temperature +-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH;	temperature 0.1Celsius
Repeatability	humidity +-1%RH;	temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH	
Long-term Stability	+-0.5%RH/year	
Interchangeability	fully interchangeable	

Thomas Liu (Sales Manager)

Email: thomasliu198518@aliyun.com

II.4 Datasheet BMP180

BMP180
Digital, barometric pressure sensor

Bosch Sensortec

BOSCH
Invented for life

General description

The BMP180 is the new digital barometric pressure sensor of Bosch Sensortec, with a very high performance, which enables applications in advanced mobile devices, such as smart phones, tablet PCs and sports devices. It follows the BMP085 and brings many improvements, like the smaller size and the expansion of digital interfaces.

The ultra-low power consumption down to 3 µA makes the BMP180 the leader in power saving for your mobile devices. BMP180 is also distinguished by its very stable behavior (performance) with regard to the independency of the supply voltage.

Technology and specification

Through its high relative accuracy of ±0.12 hPa (±1m) the BMP180 has become the most reliable sensor for precise applications, like indoor-navigation. The small size of 3.6 x 3.8 mm² and the height of only 0.93 mm makes it very suitable for the implementation in small mobile devices. The high absolute accuracy (please see parameter sheet beside) and a noise level down to 0.02 hPa (altitude of 17 cm) open new perspectives for applications in the sport devices.

The BMP180 is a sensor based on piezo-resistive MEMS technology for EMC robustness and high quality standards. The dies of the BMP180 are protected by a stable and thin LGA package with a metal lid. The package has seven optimized pins. The BMP180 can communicate directly with a microcontroller in the device through I²C or SPI as a variant.

BMP180 target applications

- Indoor navigation
- GPS-enhancement for dead-reckoning, slope detection, etc.
- Sport devices, e.g. altitude profile
- Weather forecast
- Vertical velocity indication (rise/sink speed)

Technical Data	BMP180
Pressure range	300 ... 1100 hPa
RMS noise expressed in pressure	0.06 hPa, typ. (ultra low power mode) 0.02 hPa, typ (ultra high resolution mode)
RMS noise expressed in altitude	0.06 hPa, typ. (ultra low power mode)
	0.02 hPa, typ. (ultra high resolution mode)
Relative accuaracy pressure V_{DD} = 3.3 V	950 ... 1050 hPa @ 25 °C ±0.12 hPa ±1.0 m
	700 ... 900 hPa 25 ... 40 °C ±0.12 hPa ±1.0 m
Absolute accuracy p=300 ... 1100hPa (T=0 ... +65°C,	Pressure: -4.0 ... +2.0 hPa Temperature: -1 hPa (+/- 1 hPa), typ.
Average current consumption (1Hz data refresh rate)	3 µA, typ. (ultra-low power mode)
	32 µA, typ. (advanced mode)
Peak current	650 µA, typ.
Stand-by current	0.1 µA, typ.
Supply voltage V_{DDIO} Supply voltage V_{DD}	1.62 ... 3.6 V 1.8 ... 3.6 V
Operation temp. range full accuracy	-40 ... +85 °C 0 ... +65 °C
Pressure conv. time	-5 msec, typ. (std. mode)
I²C date transfer rate	3.4 MHz, max.
Package type / pin no.	LGA / 7
Package dimensions	3.6 x 3.8 x 0.93 mm³

II.5 Datasheet nRF24L01+

nRF24L01+

Single Chip 2.4GHz Transceiver

Preliminary Product Specification v1.0

Key Features

- Worldwide 2.4GHz ISM band operation
- 250kbps, 1Mbps and 2Mbps on air data rates
- Ultra low power operation
- 11.3mA TX at 0dBm output power
- 13.5mA RX at 2Mbps air data rate
- 900nA in power down
- 26µA in standby-I
- On chip voltage regulator
- 1.9 to 3.6V supply range
- Enhanced ShockBurst™
- Automatic packet handling
- Auto packet transaction handling
- 6 data pipe MultiCeiver™
- Drop-in compatibility with nRF24L01
- On-air compatible in 250kbps and 1Mbps with nRF2401A, nRF2402, nRF24E1 and nRF24E2
- Low cost BOM
- ±60ppm 16MHz crystal
- 5V tolerant inputs
- Compact 20-pin 4x4mm QFN package

Applications

- Wireless PC Peripherals
- Mouse, keyboards and remotes
- 3-in-1 desktop bundles
- Advanced Media center remote controls
- VoIP headsets
- Game controllers
- Sports watches and sensors
- RF remote controls for consumer electronics
- Home and commercial automation
- Ultra low power sensor networks
- Active RFID
- Asset tracking systems
- Toys

March 2008

II.6 Datasheet CN3065

CONSONANCE

Lithium Ion Battery Charger for Solar-Powered Systems

CN3065

General Description:

The CN3065 is a complete constant-current /constant voltage linear charger for single cell Li-ion and Li Polymer rechargeable batteries. The device contains an on-chip power MOSFET and eliminates the need for the external sense resistor and blocking diode. An on-chip 8-bit ADC can adjust charging current automatically based on the output capability of input power supply, so CN3065 is ideally suited for solar powered system. Furthermore, the CN3065 is specifically designed to work within USB power specifications. Thermal feedback regulates the charge current to limit the die temperature during high power operation or high ambient temperature. The regulation voltage is internally fixed at 4.2V with 1% accuracy, it can also be adjusted with an external resistor. The charge current can be programmed externally with a single resistor. When the input supply is removed, the CN3065 automatically enters a low power sleep mode , dropping the battery drain current to less than 3uA. Other features include undervoltage lockout, automatic recharge, battery temperature sensing and charging/termination indicator.
The CN3065 is available in 8-pin DFN package.

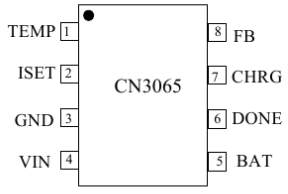
Applications:

- Solar Powered System
- Digital Still Cameras
- MP3 Players
- Bluetooth Applications
- Portable Devices
- Chargers

Features:

- On-chip 8-bit ADC can adjust charging current automatically based on the output capability of input power supply
- Suitable for Solar-Powered System
- On-chip Power MOSFET
- No external Blocking Diode or Current Sense Resistors Required
- Preset 4.2V Regulation Voltage with 1% Accuracy, adjustable with an external resistor
- Precharge Conditioning for Reviving Deeply Discharged Cells and Minimizing Heat Dissipation During Initial Stage of Charge
- Continuous Programmable Charge Current Up to 1000mA
- Constant-Current/Constant-Voltage Operation with Thermal Regulation to Maximize Charge Rate Without Risk of Overheating
- Automatic Low-Power Sleep Mode When Input Supply Voltage is Removed
- Status Indication for LEDs or uP Interface
- C/10 Charge Termination
- Automatic Recharge
- Battery Temperature Sensing
- Available in DFN-8 Package
- Pb-free Available

Pin Assignment



II.7 Datasheet ATmega328

Atmel

ATmega48A/PA/88A/PA/168A/PA/328/P

ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH

Features

- High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
  - 256/512/512/1KBytes EEPROM
  - 512/1K/1K/2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix® acquisition
  - Up to 64 sense channels
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change

1. Pin Configurations

Figure 1-1. Pinout ATmega48A/PA/88A/PA/168A/PA/328/P

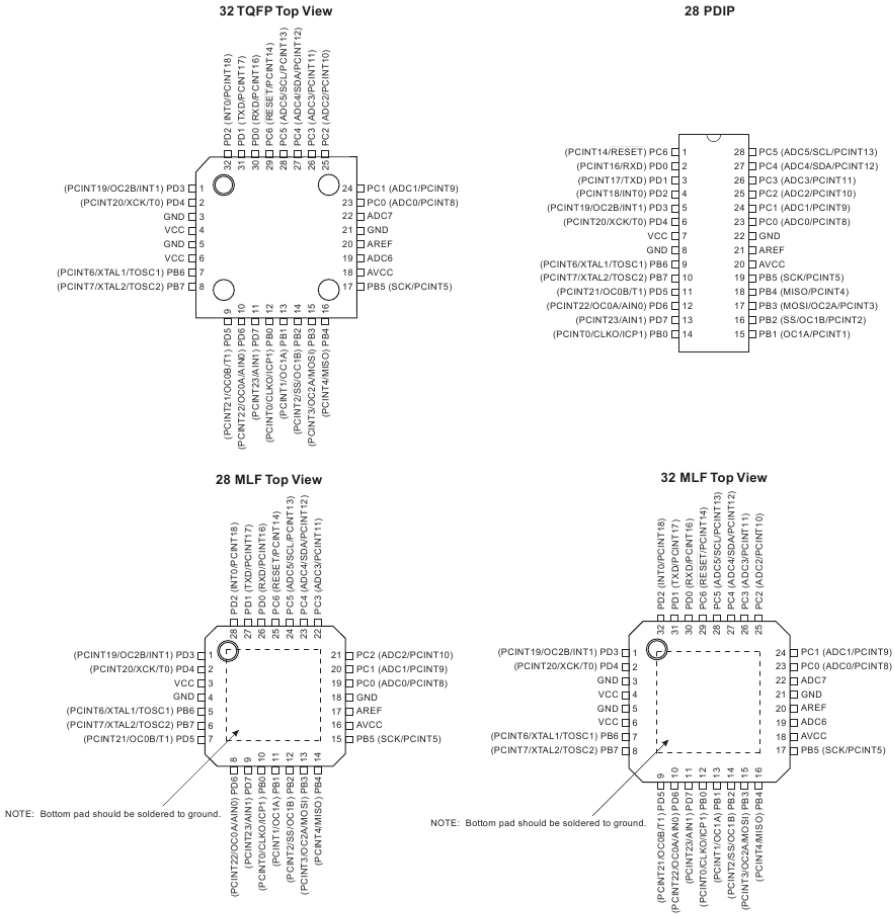


Table 1-1. 32UFBGA - Pinout ATmega48A/48PA/88A/88PA/168A/168PA

	1	2	3	4	5	6
A	PD2	PD1	PC6	PC4	PC2	PC1
B	PD3	PD4	PD0	PC5	PC3	PC0
C	GND	GND			ADC7	GND
D	VDD	VDD			AREF	ADC6
E	PB6	PD6	PB0	PB2	AVDD	PB5
F	PB7	PD5	PD7	PB1	PB3	PB4

II.8 Datasheet ATmega2560

Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V

8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash

SUMMARY

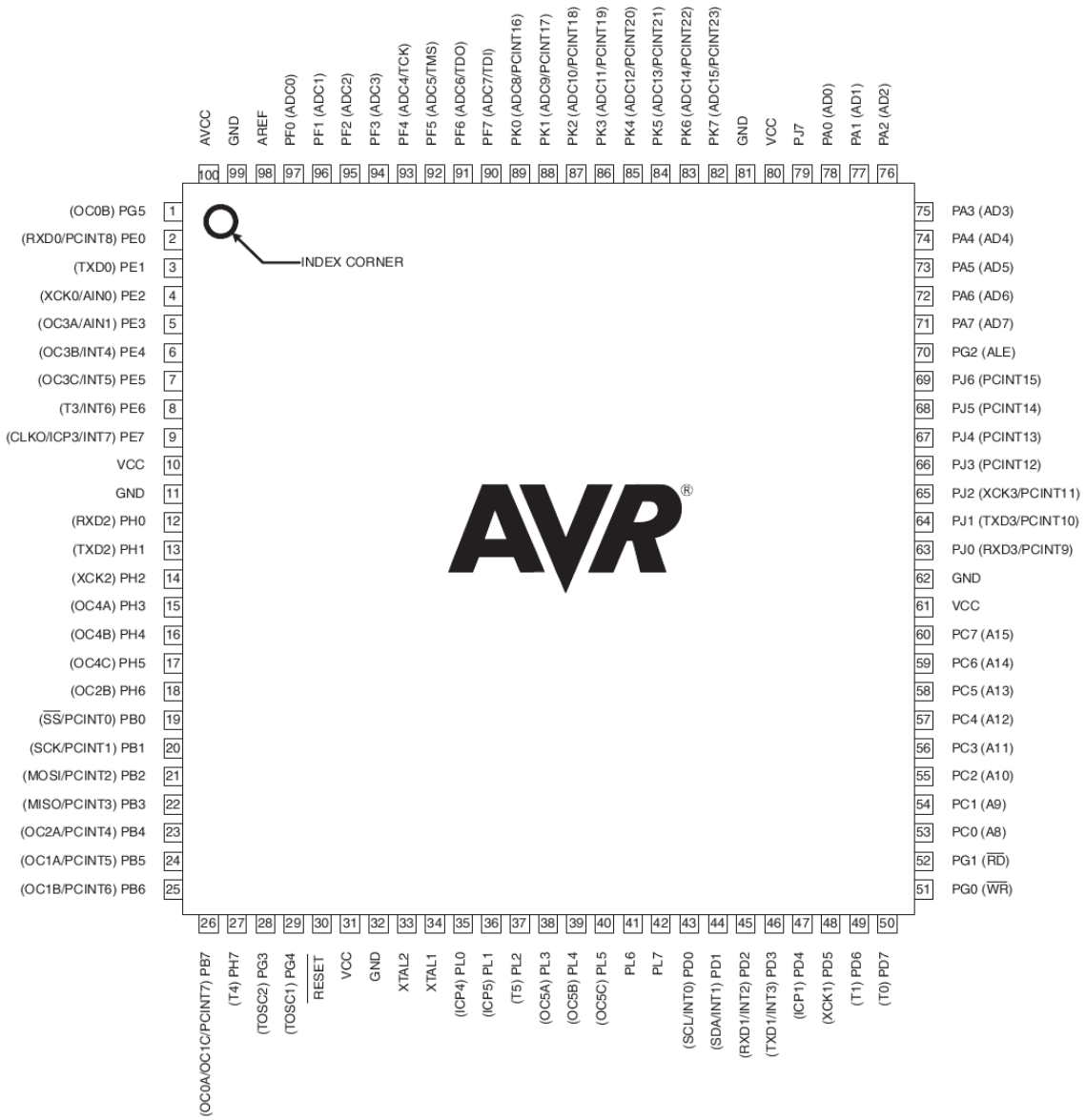
Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256KBytes of In-System Self-Programmable Flash
  - 4Kbytes EEPROM
  - 8Kbytes Internal SRAM
  - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
    - Endurance: Up to 64Kbytes Optional External Memory Space
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix acquisition
  - Up to 64 sense channels
- JTAG (IEEE® std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green
- Temperature Range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode: 1MHz, 1.8V: 500µA
  - Power-down Mode: 0.1µA at 1.8V
- Speed Grade:
  - ATmega640V/ATmega1280V/ATmega1281V:
    - 0 - 4MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega2560V/ATmega2561V:
    - 0 - 2MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega640/ATmega1280/ATmega1281:
    - 0 - 8MHz @ 2.7V - 5.5V, 0 - 16MHz @ 4.5V - 5.5V
  - ATmega2560/ATmega2561:
    - 0 - 16MHz @ 4.5V - 5.5V

2549QS–AVR–02/2014

1. Pin Configurations

Figure 1-1. TQFP-pinout ATmega640/1280/2560



II.9 Datasheet WIZnet W5100

W5100 Datasheet

The W5100 is a full-featured, single-chip Internet-enabled 10/100 Ethernet controller designed for embedded applications where ease of integration, stability, performance, area and system cost control are required. The W5100 has been designed to facilitate easy implementation of Internet connectivity without OS. The W5100 is IEEE 802.3 10BASE-T and 802.3u 100BASE-TX compliant.

The W5100 includes fully hardwired, market-proven TCP/IP stack and integrated Ethernet MAC & PHY. Hardwired TCP/IP stack supports TCP, UDP, IPv4, ICMP, ARP, IGMP and PPPoE which has been proven in various applications for several years. 16Kbytes internal buffer is included for data transmission. No need of consideration for handling Ethernet Controller, but simple socket programming is required.

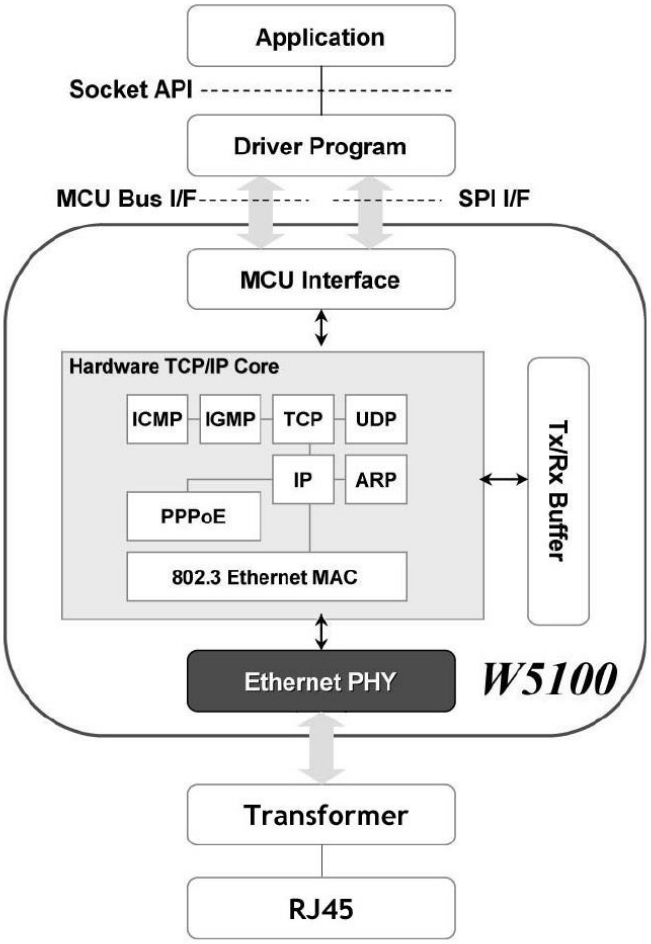
For easy integration, three different interfaces like memory access way, called direct, indirect bus and SPI, are supported on the MCU side.

Target Applications

The W5100 is well suited for many embedded applications, including:

- Home Network Devices: Set-Top Boxes, PVRs, Digital Media Adapters
- Serial-to-Ethernet: Access Controls, LED displays, Wireless AP relays, etc.
- Parallel-to-Ethernet: POS / Mini Printers, Copiers
- USB-to-Ethernet: Storage Devices, Network Printers
- GPIO-to-Ethernet: Home Network Sensors
- Security Systems: DVRs, Network Cameras, Kiosks
- Factory and Building Automations
- Medical Monitoring Equipments
- Embedded Servers

Block Diagram



II.10 Datasheet LM393

TEXAS INSTRUMENTS

LM193-N, LM2903-N, LM293-N, LM393-N

SNOSBJ6F – OCTOBER 1999 – REVISED DECEMBER 2014

LMx93-N, LM2903-N Low-Power, Low-Offset Voltage, Dual Comparators

1 Features

- Wide Supply
  - Voltage Range: 2.0 V to 36 V
  - Single or Dual Supplies: ±1.0 V to ±18 V
- Very Low Supply Current Drain (0.4 mA) — Independent of Supply Voltage
- Low Input Biasing Current: 25 nA
- Low Input Offset Current: ±5 nA
- Maximum Offset voltage: ±3 mV
- Input Common-Mode Voltage Range Includes Ground
- Differential Input Voltage Range Equal to the Power Supply Voltage
- Low Output Saturation Voltage: 250 mV at 4 mA
- Output Voltage Compatible with TTL, DTL, ECL, MOS and CMOS logic systems
- Available in the 8-Bump (12 mil) DSBGA Package
- See AN-1112 (SNVA009) for DSBGA Considerations
- Advantages
  - High Precision Comparators
  - Reduced V_{OS} Drift Over Temperature
  - Eliminates Need for Dual Supplies
  - Allows Sensing Near Ground
  - Compatible with All Forms of Logic
  - Power Drain Suitable for Battery Operation

2 Applications

- Battery powered applications
- Industrial applications

3 Description

The LM193-N series consists of two independent precision voltage comparators with an offset voltage specification as low as 2.0 mV max for two comparators which were designed specifically to operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage. These comparators also have a unique characteristic in that the input common-mode voltage range includes ground, even though operated from a single power supply voltage.

Application areas include limit comparators, simple analog to digital converters; pulse, squarewave and time delay generators; wide range VCO; MOS clock timers; multivibrators and high voltage digital logic gates. The LM193-N series was designed to directly interface with TTL and CMOS. When operated from both plus and minus power supplies, the LM19-N series will directly interface with MOS logic where their low power drain is a distinct advantage over standard comparators.

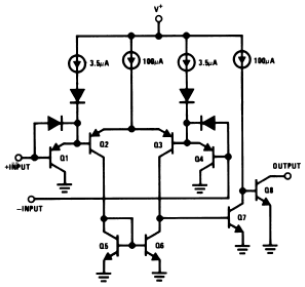
The LM393 and LM2903 parts are available in TI's innovative thin DSBGA package with 8 (12 mil) large bumps.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM193-N	TO-99 (8)	9.08 mm x 9.08 mm
LM293-N		
LM393-N	SOIC (8)	4.90 mm x 3.91 mm
LM2903-N		

(1) For all available packages, see the orderable addendum at the end of the datasheet.

4 Simplified Schematic



II.11 Datasheet LM317

National Semiconductor

July 2004

LM117/LM317A/LM317
3-Terminal Adjustable Regulator

General Description

The LM117 series of adjustable 3-terminal positive voltage regulators is capable of supplying in excess of 1.5A over a 1.2V to 37V output range. They are exceptionally easy to use and require only two external resistors to set the output voltage. Further, both line and load regulation are better than standard fixed regulators. Also, the LM117 is packaged in standard transistor packages which are easily mounted and handled.

In addition to higher performance than fixed regulators, the LM117 series offers full overload protection available only in IC's. Included on the chip are current limit, thermal overload protection and safe area protection. All overload protection circuitry remains fully functional even if the adjustment terminal is disconnected.

Normally, no capacitors are needed unless the device is situated more than 6 inches from the input filter capacitors in which case an input bypass is needed. An optional output capacitor can be added to improve transient response. The adjustment terminal can be bypassed to achieve very high ripple rejection ratios which are difficult to achieve with standard 3-terminal regulators.

Besides replacing fixed regulators, the LM117 is useful in a wide variety of other applications. Since the regulator is "floating" and sees only the input-to-output differential voltage, supplies of several hundred volts can be regulated as long as the maximum input to output differential is not exceeded, i.e., avoid short-circuiting the output.

Also, it makes an especially simple adjustable switching regulator, a programmable output regulator, or by connecting a fixed resistor between the adjustment pin and output, the LM117 can be used as a precision current regulator. Supplies with electronic shutdown can be achieved by clamping the adjustment terminal to ground which programs the output to 1.2V where most loads draw little current.

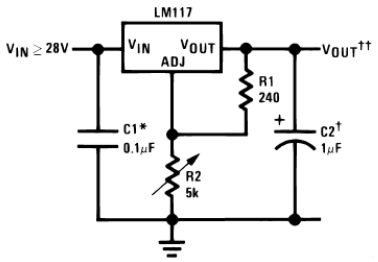
For applications requiring greater output current, see LM150 series (3A) and LM138 series (5A) data sheets. For the negative complement, see LM137 series data sheet.

Features

- Guaranteed 1% output voltage tolerance (LM317A)
- Guaranteed max. 0.01%/V line regulation (LM317A)
- Guaranteed max. 0.3% load regulation (LM117)
- Guaranteed 1.5A output current
- Adjustable output down to 1.2V
- Current limit constant with temperature
- P+ Product Enhancement tested
- 80 dB ripple rejection
- Output is short-circuit protected

Typical Applications

1.2V–25V Adjustable Regulator



00906301

Full output current not available at high input-output voltages

*Needed if device is more than 6 inches from filter capacitors.

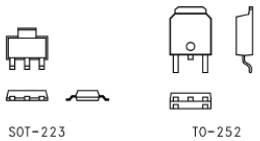
†Optional — improves transient response. Output capacitors in the range of 1μF to 1000μF of aluminum or tantalum electrolytic are commonly used to provide improved output impedance and rejection of transients.

$$\dagger\dagger V_{OUT} = 1.25V\left(1 + \frac{R2}{R1}\right) + I_{ADJ}(R_2)$$

LM117 Series Packages

Part Number Suffix	Package	Design Load Current
K	TO-3	1.5A
H	**TO-39**	**0.5A**
T	TO-220	1.5A
E	**LCC**	**0.5A**
S	TO-263	1.5A
EMP	SOT-223	1A
MDT	**TO-252**	**0.5A**

SOT-223 vs. D-Pak (TO-252) Packages



00906354

Scale 1:1

Anexo III.- Tabla de valores de Kc

CUADRO 12

Valores del coeficiente único (promedio temporal) del cultivo, K_c y alturas medias máximas de las plantas para cultivos no estresados y bien manejados en dimas sub-húmedos ($HR_{min} \approx 45\%$, $u_2 \approx 2$ m s^{-1}) para usar en la formula de la FAO Penman-Monteith ET_o.

Cultivo	K_{cini} [1]	$K_{c\,med}$	$K_{c\,fin}$	Altura Máx. Cultivo (h) (m)
a. Hortalizas Pequeñas	**0,7**	**1,05**	**0,95**	
Brécol (Brócoli)		1,05	0,95	0,3
Col de Bruselas		1,05	0,95	0,4
Repollo		1,05	0,95	0,4
Zanahoria		1,05	0,95	0,3
Coliflor		1,05	0,95	0,4
Apio (Céleri)		1,05	1,00	0,6
Ajo		1,00	0,70	0,3
Lechuga		1,00	0,95	0,3
Cebolla – seca		1,05	0,75	0,4
– verde		1,00	1,00	0,3
– semilla		1,05	0,80	0,5
Espinaca		1,00	0,95	0,3
Rábano		0,90	0,85	0,3
b. Hortalizas– Familia de la Solanáceas	**0,6**	**1,15**	**0,80**	
Berenjena		1,05	0,90	0,8
Pimiento Dulce (campana)		1,05 [2]	0,90	0,7
Tomate		1,15 [2]	0,70–0,90	0,6
c. Hortalizas– Familia de las Cucurbitáceas	**0,5**	**1,00**	**0,80**	
Melón	0,5	0,85	0,60	0,3
Pepino – Cosechado Fresco	0,6	1,00 [2]	0,75	0,3
– Cosechado a Máquina	0,5	1,00	0,90	0,3
Calabaza de Invierno		1,00	0,80	0,4
Calabacín (zucchini)		0,95	0,75	0,3
Melón dulce		1,05	0,75	0,4
Sandía	0,4	1,00	0,75	0,4
d. Raíces y Tubérculos	**0,5**	**1,10**	**0,95**	
Remolacha, mesa		1,05	0,95	0,4
Yuca o Mandioca – año 1	0,3	0,80 [3]	0,30	1,0
– año 2	0,3	1,10	0,50	1,5
Chirivía	0,5	1,05	0,95	0,4
Patata o Papa		1,15	0,75 [4]	0,6
Camote o Batata		1,15	0,65	0,4
Nabos (Rutabaga)		1,10	0,95	0,6
Remolacha Azucarera	0,35	1,20	0,70 [5]	0,5

CUADRO 12 (continuación)

Cultivo	$K_{c\,ini}$[1]	$K_{c\,med}$	$K_{c\,fin}$	Altura Máx. Cultivo (h) (m)
e. Leguminosas (Leguminosae)	**0,4**	**1,15**	**0,55**	
Frijoles o judías, verdes	0,5	1,05[2]	0,90	0,4
Frijoles o judías, secos y frescos	0,4	1,15[2]	0,35	0,4
Garbanzo (chick pea)		1,00	0,35	0,4
Habas – Fresco – Seco/Semilla	0,5 0,5	1,15[2] 1,15[2]	1,10 0,30	0,8 0,8
Garbanzo hindú	0,4	1,15	0,35	0,8
Caupís (cowpeas)		1,05	0,60-0,35[6]	0,4
Maní		1,15	0,60	0,4
Lentejas		1,10	0,30	0,5
Guisantes o arveja – Frescos – Secos/Semilla	0,5	1,15[2] 1,15	1,10 0,30	0,5 0,5
Soya		1,15	0,50	0,5-1,0
f. Hortalizas perennes (con letargo invernal y suelo inicialmente desnudo o con mantillo)	**0,5**	**1,00**	**0,80**	
Alcachofa	0,5	1,00	0,95	0,7
Espárragos	0,5	0,95[7]	0,30	0,2-0,8
Menta	0,60	1,15	1,10	0,6-0,8
Fresas	0,40	0,85	0,75	0,2
g. Cultivos Textiles	**0,35**			
Algodón		1,15-1,20	0,70-0,50	1,2-1,5
Lino		1,10	0,25	1,2
Sisal[8]		0,4-0,7	0,4-0,7	1,5
h. Cultivos Oleaginosos	**0,35**	**1,15**	**0,35**	
Ricino		1,15	0,55	0,3
Canola (colza)		1,0-1,15[9]	0,35	0,6
Cártamo		1,0-1,15[9]	0,25	0,8
Sésamo (ajonjolí)		1,10	0,25	1,0
Girasol		1,0-1,15[9]	0,35	2,0
i. Cereales	**0,3**	**1,15**	**0,4**	
Cebada		1,15	0,25	1
Avena		1,15	0,25	1
Trigo de Primavera		1,15	0,25-0,4[10]	1
Trigo de Invierno – con suelos congelados – con suelos no-congelados	0,4 0,7	1,15 1,15	0,25-0,4[10] 0,25-0,4[10]	1
Maíz, (grano)		1,20	0,60,0,35[11]	2
Maíz, (dulce)		1,15	1,05[12]	1,5
Mijo		1,00	0,30	1,5
Sorgo – grano – dulce		1,00-1,10 1,20	0,55 1,05	1-2 2-4
Arroz	1,05	1,20	0,90-0,60	1

CUADRO 12 (continuación)

Cultivo	K_{cini}[1]	K_{cmed}	K_{cfin}	Altura Máx. Cultivo (h) (m)
j. Forrajes				
Alfalfa (heno) – efecto promedio de los cortes	0,40	0,95[13]	0,90	0,7
– períodos individuales de corte	0,40[14]	1,20[14]	1,15[14]	0,7
– para semilla	0,40	0,50	0,50	0,7
Bermuda (heno) – efecto promedio de los cortes	0,55	1,00[13]	0,85	0,35
– cultivo para semilla (primavera)	0,35	0,90	0,65	0,4
Trébol heno, Bersim – efecto promedio de los cortes	0,40	0,90[13]	0,85	0,6
– períodos individuales de corte	0,40[14]	1,15[14]	1,10[14]	0,6
Rye Grass (heno) – efecto promedio de los cortes	0,95	1,05	1,00	0,3
Pasto del Sudán (anual) – efecto promedio de los cortes	0,50	0,90[14]	0,85	1,2
– período individual de corte	0,50[14]	1,15[14]	1,10[14]	1,2
Pastos de Pastoreo – pastos de rotación	0,40	0,85-1,05	0,85	0,15-0,30
– pastoreo extensivo	0,30	0,75	0,75	0,10
Pastos (césped, turf grass) – época fría[15]	0,90	0,95	0,95	0,10
– época caliente[15]	0,80	0,85	0,85	0,10
k. Caña de azúcar	0,40	1,25	0,75	3
l. Frutas Tropicales y Árboles				
Banana – 1er año	0,50	1,10	1,00	3
– 2do año	1,00	1,20	1,10	4
Cacao	1,00	1,05	1,05	3
Café – suelo sin cobertura	0,90	0,95	0,95	2-3
– con malezas	1,05	1,10	1,10	2-3
Palma Datilera	0,90	0,95	0,95	8
Palmas	0,95	1,00	1,00	8
Piña[16] – suelo sin cobertura	0,50	0,30	0,30	0,6-1,2
– con cobertura de gramíneas	0,50	0,50	0,50	0,6-1,2
Árbol del Caucho	0,95	1,00	1,00	10
Té – no sombreado	0,95	1,00	1,00	1,5
– sombreado[17]	1,10	1,15	1,15	2
m. Uvas y Moras				
Moras (arbusto)	0,30	1,05	0,50	1,5
Uvas – Mesa o secas (pasas)	0,30	0,85	0,45	2
– Vino	0,30	0,70	0,45	1,5-2
Lúpulo	0,3	1,05	0,85	5

CUADRO 12 (continuación)

Cultivo	K_{cini}[1]	K_{cmed}	K_{cfin}	Altura Máx. Cultivo (h) (m)
n. Árboles Frutales				
Almendras, sin cobertura del suelo	0,40	0,90	0,65[18]	5
Manzanas, Cerezas, Peras[19]				
– sin cobertura del suelo, con fuertes heladas	0,45	0,95	0,70[18]	4
– sin cobertura del suelo, sin heladas	0,60	0,95	0,75[18]	4
– cobertura activa del suelo, con fuertes heladas	0,50	1,20	0,95[18]	4
– cobertura activa del suelo, sin heladas	0,80	1,20	0,85[18]	4
Albaricoque, Melocotón o Durazno, Drupas[19,20]				
– sin cobertura del suelo, con fuertes heladas	0,45	0,90	0,65[18]	3
– sin cobertura del suelo, sin heladas	0,55	0,90	0,65[18]	3
– cobertura activa del suelo, con fuertes heladas	0,50	1,15	0,90[18]	3
– cobertura activa del suelo, sin heladas	0,80	1,15	0,85[18]	3
Aguacate, sin cobertura del suelo	0,60	0,85	0,75	3
Cítricos, sin cobertura del suelo[21]				
– 70% cubierta vegetativa	0,70	0,65	0,70	4
– 50% cubierta vegetativa	0,65	0,60	0,65	3
– 20% cubierta vegetativa	0,50	0,45	0,55	2
Cítricos, con cobertura activa del suelo o malezas[22]				
– 70% cubierta vegetativa	0,75	0,70	0,70	4
– 50% cubierta vegetativa	0,80	0,80	0,80	3
– 20% cubierta vegetativa	0,85	0,85	0,85	2
Coníferas[23]	1,00	1,00	1,00	10
Kiwi	0,40	1,05	1,05	3
Olivos (40 a 60% de cobertura del suelo por el dosel)[24]	0,65	0,70	0,70	3-5
Pistachos, sin cobertura del suelo	0,40	1,10	0,45	3-5
Huerto de Nogal[19]	0,50	1,10	0,65[18]	4-5

Obtenido de: FAO, Estudio FAO Riego y Drenaje Nº 56 [11].

Anexo IV.- Código implementado

Código de los dispositivos Esclavos

```
/*****************************************************

  Esclavo

 *****************************************************/

#include <SPI.h>
#include "RF24.h"
#include <OneWire.h>
#include <DallasTemperature.h>
#include <DHT.h>
#include <Wire.h>
#include <Adafruit_BMP085.h>

#define DHTPIN 6
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

#define ONE_WIRE_BUS 4
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
```

```
Adafruit_BMP085 bmp;

RF24 radio(9, 10);

byte addresses[][6] = {"1Node", "2Node"};
int valvCerrada = 0;

#define pinLed 5

struct dataStruct {
  float temperatura = 0;
  float humedad = 0;
  float presion = 0;
} info;

struct dataStruct2 {
  int idNodo = 9;
  int riegoNodo0 = 0;
  int riegoNodo1 = 0;
  //int responder = 0;
  //float duracion = 0;
} peticiones;
/*************************************************************/
void setup() {
```

```
Serial.begin(115200);
Serial.println(F("empezando"));

pinMode(pinLed, OUTPUT);

/***************SETUP TEMPERATURA*****************************/
Serial.print("Locating devices...");
sensors.begin();
Serial.print("Found ");
Serial.print(sensors.getDeviceCount(), DEC);
Serial.println(" devices.");

// report parasite power requirements
Serial.print("Parasite power is: ");
if (sensors.isParasitePowerMode()) Serial.println("ON");
else Serial.println("OFF");
/************************************************************/

dht.begin();

radio.begin();


radio.setPALevel(RF24_PA_LOW);
```

```
  radio.openWritingPipe(addresses[0]);
  radio.openReadingPipe(1, addresses[1]);


  radio.startListening();
}

/********************loop ****************************/
void loop() {
  if ( radio.available()) {

    while (radio.available()) {                              // While there is data ready
      radio.read( &peticiones, sizeof(peticiones) );             // Get the payload
    }


    if ((peticiones.idNodo == 1) /*&& (peticiones.responder == 1)*/ ) {
      Serial.println("peticion al arduino1 ");
      Serial.println(peticiones.idNodo);
      Serial.println(peticiones.riegoNodo0);
      Serial.println(peticiones.riegoNodo1);
```

```
  //info.presion = info.presion + 1;

  sensors.requestTemperatures();
  info.temperatura = sensors.getTempCByIndex(0);
  info.humedad = dht.readHumidity();
  info.presion = bmp.readPressure();

  radio.stopListening();
  radio.write( &info, sizeof(info) );
  radio.startListening();
  Serial.println(F("Respuesta del arduino 1 "));
  Serial.println(info.temperatura);
  Serial.println(info.humedad);
}

if ((peticiones.idNodo == 1) && (peticiones.riegoNodo1 == 1) ) {

  Serial.println(F("Comienza riego... "));
  digitalWrite(pinLed, HIGH);
  valvCerrada = 0;

}
```

```
    if ((peticiones.idNodo == 1) && (peticiones.riegoNodo1 == 0) ) {


      if (valvCerrada == 0) {
        Serial.println(F("Fin de riego... "));
        digitalWrite(pinLed, LOW);
        valvCerrada = 1;
      }


    }


  }


} // Loop
```

Código del dispositivo Maestro

```
/**************************************
   Sistema de riegos. Maestro

**************************************/
#include <avr/sleep.h>
```

```
#include <Sodaq_DS3231.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <Keypad.h>
#include <SD.h>

#include "RF24.h"

volatile byte flag = 0;
RF24 radio(7, 8);

byte addresses[][6] = {"1Node", "2Node"};


struct dataStruct {
  float Temp = 0;
  float Hum = 0;
} info;

struct dataStruct3 {
  float Temp = 0;
  float Hum = 0;
} buf;
```

```
struct dataStruct4 {
  float Temp = 0;
  float Hum = 0;
  float TempMin = 100;
  float TempMax = 0;
  float HumMin = 100;
  float HumMax = 0;
} esclavo0;

struct dataStruct5 {
  float Temp = 0;
  float Hum = 0;
  float TempMin = 100;
  float TempMax = 0;
  float HumMin = 100;
  float HumMax = 0;
} esclavo1;

struct dataStruct6 {
  float kc = 0;
  float caudal = 0;
  float superficie = 0;
} datos0;
```

```
struct dataStruct7 {
  float kc = 0;
  float caudal = 0;
  float superficie = 0;
} datos1;

struct dataStruct8 {
  float kc = 0;
  float caudal = 0;
  float superficie = 0;
} datos;

int k = 0;

int NumNodos = 2;

struct dataStruct2 {
  int idNodo = 1;
  int riegoNodo0 = 0;
  int riegoNodo1 = 0;
  //  int responder = 0;
} enviar;
```

```
float horaRiego;


boolean riegoAuto = true;


int j = 1;
float tiempoFinal0 = 0;
float tiempoFinal1 = 0;
const byte ROWS = 4;
const byte COLS = 4;
char hexaKeys[ROWS][COLS] = {
  {'0', '1', '2', '3'},
  {'4', '5', '6', '7'},
  {'8', '9', '.', '#'},
  {'A', 'B', 'C', 'D'}
};
byte rowPins[ROWS] = {16, 3, 14, 5};
byte colPins[COLS] = {6, 15, 17, 9};
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);



#define I2C_ADDR    0x3F
LiquidCrystal_I2C           lcd(I2C_ADDR, 2, 1, 0, 4, 5, 6, 7);
float prueba = 0;
```

```
/*****************************/
void setup() {

  Serial.begin(115200);
  Serial.println(F("prueba 14:21"));
  lcd.begin (20, 4);
  lcd.setBacklightPin(3, POSITIVE);
  lcd.setBacklight(HIGH);

  radio.begin();
  PORTD |= 0x04;
  DDRD &= ~ 0x04;
  pinMode(2, INPUT);
  rtc.begin();
  attachInterrupt(0, INT0_ISR, FALLING); // Se aceptan interrupciones en el primer pin (INT0) cuando pasa de High a Low, y se llama a la funcion INT0_ISR
  rtc.enableInterrupts(EveryHour);

  //radio.setPALevel(RF24_PA_LOW);

  radio.openWritingPipe(addresses[1]);
  radio.openReadingPipe(1, addresses[0]);

  radio.startListening();
```

```
  Comprobar si la tarjeta está presente y puede ser inicializada
  if (!SD.begin(4)) {
    Serial.println("Fallo en la tarjeta o no colocada");
    // No realizar nada mas
    return;
  }
  Serial.println("Tarjeta inicializada");

}

/*****LOOP*********/

void loop() {
  DateTime now = rtc.now();
  if (1) {

    pantallaPrincipal();
  }

  if ((now.hour() == 0) && (now.minute() == 20) && (riegoAuto == false) && (now.second() == 0)) {
    riegoAuto = true;
  }
  if ((now.hour() == 0) && (now.minute() == 20) && (now.second() == 0)) {
```

```
        esclavo0.TempMin = 100;
        esclavo0.TempMax = 0;
        esclavo0.HumMin = 100;
        esclavo0.HumMax = 0;
        esclavo1.TempMin = 100;
        esclavo1.TempMax = 0;
        esclavo1.HumMin = 100;
        esclavo1.HumMax = 0;
      }
      if ((now.hour() == 19) && (now.minute() == 10) && (riegoAuto == true) && (now.second() == 30)) {
        float tiempo0 = tiempoRiego (datos0.kc, datos0.caudal, datos0.superficie, esclavo0.TempMin, esclavo0.TempMax, esclavo0.HumMin, esclavo0.HumMax);
        float tiempo1 = tiempoRiego (datos1.kc, datos1.caudal, datos1.superficie, esclavo0.TempMin, esclavo0.TempMax, esclavo0.HumMin, esclavo0.HumMax);
        tiempoFinal0 = 55 + tiempo0;
        tiempoFinal1 = 55 + tiempo1;
        activarRiego (1, 1);

      }

      if ((now.hour() == 19) && (now.minute() == tiempoFinal0) && (now.second() == 0))
{
        anularRiego(0, enviar.riegoNodo1);
```

```
      }
      if ((now.hour() == 19) && (now.minute() == tiempoFinal1) && (now.second() == 0))
{
        anularRiego(enviar.riegoNodo0, 0);
      }




      if (flag == 1) {
        flag = 0;
        rtc.clearINTStatus();
        Serial.println("Interrupcion!");
        funcionRadio();
      }




    } // Loop

    /*****************************************
    *******Declaracion de Funciones*********
    *****************************************/
```

```
void funcionRadio() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Estableciendo ");
  lcd.setCursor(0, 1);
  lcd.print("comunicacion con los");
  lcd.setCursor(0, 2);
  lcd.print("Esclavos...");
  lcd.setCursor(0, 3);
  lcd.print("                    ");
  delay(1000);

  j = j + 1;
  Serial.println("**********************");
  Serial.print(F("peticion numero: "));
  Serial.println(j);


  for (int i = 0; i < NumNodos; i++) {
    while (k == 0) {


      radio.stopListening();
      enviar.idNodo = i;
```

```
unsigned long time = micros();

if (!radio.write( &enviar, sizeof(enviar) )) {
  //        Serial.println(F("fallo al enviar"));
}

radio.startListening();
unsigned long started_waiting_at = micros();
boolean timeout = false;

while ( ! radio.available() ) {
  if (micros() - started_waiting_at > 200000 ) {
    timeout = true;
    break;
  }
}

if ( timeout ) {
  Serial.println(F("Fallo, time out"));
} else {
  unsigned long got_time;
  buf.Temp = 0;
  buf.Hum = 0;
  radio.read( &buf, sizeof(buf) );
```

```
    if ( (buf.Temp != 0) && (buf.Hum != 0) ) {
      k = 1;
      info.Temp =  buf.Temp;
      info.Hum =  buf.Hum;
    }

    unsigned long time = micros();

  }
} //fin del bucle while
k = 0;

if (i == 0) {
  esclavo0.Temp =  info.Temp;
  esclavo0.Hum = info.Hum;
  Serial.println(F("esclavo 0: "));
  Serial.println(esclavo0.Temp);
  Serial.println(esclavo0.Hum);

}
if (i == 1) {
  esclavo1.Temp =  info.Temp;
  esclavo1.Hum =  info.Hum;
  Serial.println(F("esclavo 1: "));
```

```
    Serial.println(esclavo1.Temp);
    Serial.println(esclavo1.Hum);
  }



}// fin del bucle for
if (esclavo0.TempMin > esclavo0.Temp) {
  esclavo0.TempMin = esclavo0.Temp;
}
if (esclavo0.TempMax < esclavo0.Temp) {
  esclavo0.TempMax = esclavo0.Temp;
}
if (esclavo0.HumMin > esclavo0.Hum) {
  esclavo0.HumMin = esclavo0.Hum;
}
if (esclavo0.HumMax < esclavo0.Hum) {
  esclavo0.HumMax = esclavo0.Hum;
}
if (esclavo1.TempMin > esclavo1.Temp) {
  esclavo1.TempMin = esclavo1.Temp;
}
Serial.println(esclavo1.TempMax);
if (esclavo1.TempMax < esclavo1.Temp) {
```

```
  esclavo1.TempMax = esclavo1.Temp;
  Serial.println(esclavo1.TempMax);
}
if (esclavo1.HumMin > esclavo1.Hum) {
  esclavo1.HumMin = esclavo1.Hum;
}
if (esclavo1.HumMax < esclavo1.Hum) {
  esclavo1.HumMax = esclavo1.Hum;
}



File Datos = SD.open("datalog.txt", FILE_WRITE);

// Si el archivo esta disponible, se guardan los datos
if (Datos) {
  Serial.println("Guardando datos");
  Datos.print(now.date(), DEC); //Fecha y hora
  Datos.print(",");
  Datos.print(now.month(), DEC);
  Datos.print(",");
  Datos.print(now.year(), DEC);
  Datos.print(",");
  Datos.print(now.hour(), DEC);
  Datos.print(",");
```

```
  Datos.print(now.minute(), DEC);
  Datos.print(",");
  Datos.print(now.second(), DEC);
  Datos.print(",");
  Datos.print(esclavo0.Temp); //Esclavo 0, variable 1
  Datos.print(",");
  Datos.print(esclavo0.Hum); //Esclavo 0, variable 2
  Datos.print(",");
  Datos.print(esclavo1.Temp); //Esclavo 2, variable 1
  Datos.print(",");
  Datos.println(esclavo1.Hum); //Esclavo 2, variable 2
  Datos.close();
  //digitalWrite(4, HIGH); Deshabilitar SD
}
En caso contrario da error
else {
  Serial.println("error abriendo datalog.txt");
}



lcd.clear();
lcd.setCursor(0, 0);
lcd.print("fin comunicacion ");
```

```
  lcd.setCursor(0, 1);
  lcd.print("pulse # para volver al menu");
  delay(1000);
} //fin funcion radio




float introducirDato() {
  float numero = 0;
  String inString = "";
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.blink();
  char letra = '0';
  while (letra != '#') {
    letra = customKeypad.getKey();
    if (letra && (letra != '#') ) {
      lcd.print(letra);
      inString = inString + letra;
    }
  }
  numero = inString.toFloat();
  lcd.noBlink();
```

```
  lcd.clear();
  return numero;
}// fin funcion introducirDato


void pantallaPrincipal() {
  DateTime now = rtc.now();
  lcd.setCursor(0, 0);
  lcd.print(now.hour(), DEC);
  lcd.print(':');
  lcd.print(now.minute());
  //  lcd.print(':');
  //  lcd.print(now.second());
  lcd.print(' ');

  lcd.print(now.date(), DEC);
  lcd.print('/');
  lcd.print(now.month(), DEC);
  lcd.print('/');
  lcd.print(now.year(), DEC);
  lcd.print("      ");

  lcd.setCursor(0, 1);
  lcd.print("A.Introducir Datos  ");
```

```
lcd.setCursor(0, 2);
lcd.print("B.Riego manual      ");
lcd.setCursor(0, 3);
lcd.print("C.Parar riego       ");
char opcion = customKeypad.getKey();
if (opcion) {
  switch (opcion) {
    case 'A':
      pantallaPonerDatos1();
      break;
    case 'B':
      riegoExtra();
      break;
    case 'C':
      anularRiego();
      break;
    default:
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("opcion no valida");
      lcd.clear();
      break;
  }
}
```

```
}// fin funcion pantalla principal



void pantallaPonerDatos() {
  lcd.setCursor(0, 0);
  lcd.print("A.Kc: ");
  lcd.setCursor(0, 1);
  lcd.print("B.Caudal: ");
  lcd.setCursor(0, 2);
  lcd.print("C.Superficie: ");
  int i = 0;
  char opcion = '0';
  while (opcion != '#') {
    opcion = customKeypad.getKey();
    if (opcion ) {
      switch (opcion) {
        case 'A':
          datos.kc = introducirDato();
          pantallaPonerDatos();
          break;
        case 'B':
          datos.caudal = introducirDato();
          pantallaPonerDatos();
          break;
```

```
        case 'C':
          datos.superficie = introducirDato();
          pantallaPonerDatos();
          break;
        case '#':
          opcion = '#';
          i = i + 1;
          break;
        default:
          Serial.println("default");
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("OPCION no valida");
          pantallaPonerDatos();
          break;
      }
    }
  }
}//fin funcion pantallaPonerDatos

void pantallaPonerDatos1() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("A.Nodo 0: ");
```

```
lcd.setCursor(0, 1);
lcd.print("B.Nodo 1: ");
lcd.setCursor(0, 2);
lcd.print("           ");

char opcion = '0';
int i = 0;
while (opcion != '#') {
  opcion = customKeypad.getKey();
  if (opcion ) {
    switch (opcion) {
      case 'A':
        pantallaPonerDatos();
        datos0.kc = datos.kc;
        datos0.caudal = datos.caudal;
        datos0.superficie = datos.superficie;
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Esclavo 0");
        lcd.setCursor(0, 1);
        lcd.print("kc:");
        lcd.print(datos0.kc);
        lcd.setCursor(0, 2);
        lcd.print("Caudal:");
```

```
  lcd.print(datos0.caudal);
  lcd.setCursor(0, 3);
  lcd.print("Superficie:");
  lcd.print(datos0.superficie);
  delay(2000);
  break;
case 'B':
  pantallaPonerDatos();
  datos1.kc = datos.kc;
  datos1.caudal = datos.caudal;
  datos1.superficie = datos.superficie;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Esclavo 1");
  lcd.setCursor(0, 1);
  lcd.print("kc:");
  lcd.print(datos1.kc);
  lcd.setCursor(0, 2);
  lcd.print("Caudal:");
  lcd.print(datos1.caudal);
  lcd.setCursor(0, 3);
  lcd.print("Superficie:");
  lcd.print(datos1.superficie);
  delay(2000);
```

```
          break;
        case '#':
          opcion = '#';
          break;
        default:
          Serial.println("default");
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("OPCION no valida");
          pantallaPonerDatos();
          break;
      }
    }

  }
}// fin funcion ponerDatos1



void riegoExtra() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("A.Riego manual Nodo0");
  lcd.setCursor(0, 1);
```

```
lcd.print("B.Riego manual nodo1");
lcd.setCursor(0, 2);
lcd.print("                    ");
lcd.setCursor(0, 3);
lcd.print("                    ");



char opcion = '0';
int i = 0;
while (opcion != '#') {
  opcion = customKeypad.getKey();
  if (opcion ) {
    switch (opcion) {
      case 'A':
        activarRiego (1, enviar.riegoNodo1);
        break;
      case 'B':
        activarRiego (enviar.riegoNodo0, 1);
        break;
      case '#':
        opcion = '#';
        i = i + 1;
        break;
      default:
```

```
        Serial.println("default");
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("OPCION no valida");
        pantallaPonerDatos();
        break;
      }
    }
  }
  lcd.clear();
}// Fin funcion riegoExtra


void anularRiego() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("A.Anular riego Nodo0");
  lcd.setCursor(0, 1);
  lcd.print("B.Anular riego Nodo1");
  lcd.setCursor(0, 2);
  lcd.print("C.Anular riego      ");
  lcd.setCursor(0, 3);
  lcd.print("automatico del dia  ");
```

```
char opcion = '0';
while (opcion != '#') {
  opcion = customKeypad.getKey();
  if (opcion ) {
    switch (opcion) {
      case 'A':
        anularRiego (0, enviar.riegoNodo1);
        break;
      case 'B':
        anularRiego (enviar.riegoNodo0, 0);
        break;
      case 'C':
        riegoAuto = false;
        break;
      case '#':
        opcion = '#';
        break;
      default:
        Serial.println("default");
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("OPCION no valida");
        pantallaPonerDatos();
```

```
        break;
      }
    }
  }
}/* Fin funcion anularRiego


  /*Función activa electrovalvula*/
void activarRiego (int variable0, int variable1) {
  //  enviar.responder = 0;
  if (variable0 == 1) {
    enviar.riegoNodo0 = variable0;
  }
  if (variable1 == 1) {
    enviar.riegoNodo1 = variable1;
  }
  funcionRadio();

}//Funcion activar electrovalvula

/*Función para electrovalvula*/
void anularRiego (int variable0, int variable1) {
  //  enviar.responder = 0;
  if (variable0 == 0) {
```

```
    enviar.riegoNodo0 = variable0;
  }
  if (variable1 == 0) {
    enviar.riegoNodo1 = variable1;
  }
  funcionRadio();
}//Funcion parar electrovalvula



/*******Funcion que utiliza una interrupcion del reloj de tiempo real RTC****/
void INT0_ISR()
{
  //Función ISR (Lo que se realiza cuando ocurre la interrupción)

  flag = 1;
}// fin de la interrupcion



//funcion tiempo riego
float tiempoRiego (float Kc, float caudal, float superficie, float Tmin, float Tmax, float Hmin, float Hmax) {
  /******Parmetros******/
```

```
DateTime now = rtc.now();

float Tmed = (Tmax + Tmin) / 2;

int z = 215;

int u2 = 2;

float gamma = 0.06585;

int G = 0;

float alpha = 0.23;

float kRs = 0.19;

float Gsc = 0.082;

float J = ((now.month() * 275 / 9) - 30 + now.date()) - 2;

//float J = ((7 * 275 / 9) - 30 + 8) - 2;

float phi = 0.49452;

float sigma = 0.000000004903;



float delta = 0.409 * (sin((2 * 3.14 * J / 365) - (1.39)));

float dr = 1 + (0.033 * cos(2 * 3.14 * J / 365));

float ws = acos(-tan(phi) * tan(delta));


float Ra = (24 * 60 / 3.14) * Gsc * dr * ((ws * sin(phi) * sin(delta)) + (cos(phi) * cos(delta) * sin(ws)));


float Rs = kRs * Ra * sqrt((Tmax - Tmin));

float Rso = Ra * (0.75 + (0.00002) * z);
```

```
float eoTmax = 0.6108 * (pow (2.718 , ((17.27 * Tmax) / (Tmax + 237.3))));

float eoTmin = 0.6108 * (pow (2.718 , ((17.27 * Tmin) / (Tmin + 237.3))));

float ea = (((eoTmin * Hmax / 100) + (eoTmax * Hmin / 100)) / 2);

float es = (eoTmax + eoTmin) / 2;


float Rns = (1 - alpha) * Rs;

float Rnl = sigma * ((pow ((Tmax + 273.15) , 4) + pow ((Tmin + 273.15) , 4)) / 2) * (0.34 - (0.14 * sqrt(ea))) * ((1.35 * Rs / Rso) - 0.35);

float Rn = Rns - Rnl;

float D = (4098 * (0.6108 * (pow (2.718 , ((17.27 * Tmed) / (Tmed + 237.3)))))) / (pow ((Tmed + 237.3) , 2));

float ETo = ((0.408 * D * (Rn - G)) + (gamma * 900 * u2 * (es - ea) / (Tmed + 273))) / (D + (gamma * (1 + 0.34 * u2)));

float ETc = Kc * ETo;


float NHn = ETc * superficie;

float tiempo = NHn / caudal;


Serial.print("ETo: ");

Serial.println(ETo);

Serial.print("ETc: ");

Serial.println(ETc);

Serial.print("NHn: ");

Serial.println(NHn);

Serial.print("tiempo: ");
```

```
  Serial.println(tiempo);

  return tiempo;

}// fin funcion tiempoRiego



/*************************************************
 * WEB
*************************************************/



#include <SPI.h>
#include <SD.h>
#include <Ethernet.h>
#include <avr/pgmspace.h>


byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192, 168, 0, 200);
```

```
EthernetServer server(80);

boolean Riego1 = false;
boolean Riego2 = false;
boolean Riego3 = false;
boolean RiegoAll = false;
float Kc1 = 0.0;
float Kc2 = 0.0;
char caracter;
String peticion = "";
String ValorKc1 = "";
String ValorKc2 = "";


void setup() {

  Serial.begin(9600);

  pinMode(53, OUTPUT);
  digitalWrite(53, HIGH);
  Serial.print("Starting SD..");
  if (!SD.begin(4)) Serial.println("failed");
  else Serial.println("ok");
```

```
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());


}


void loop() {
  float T1 = 21.1;
  float T2 = 22.2;
  float T3 = 23.3;
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
```

```
//Coger solo los 30 primeros caracteres de la respuesta y unirlos en un string
if (peticion.length() < 50 ) {
  peticion.concat(c);
}

if (c == '\n' && currentLineIsBlank) {


  if (peticion.indexOf("DATALOG.TXT") > 0) {

    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/txt");

    client.println();

    File myFile = SD.open("DATALOG.TXT");

    if (myFile) {

      byte clientBuf[64];
      int clientCount = 0;

      while (myFile.available())
```

```
    {
      clientBuf[clientCount] = myFile.read();
      clientCount++;

      if (clientCount > 63)
      {

        client.write(clientBuf, 64);
        clientCount = 0;
      }
    }

    if (clientCount > 0) client.write(clientBuf, clientCount);

    myFile.close();
  }
  delay(1);
}

else {

  // send a standard http response header
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
```

```
client.println("Connection: close");  // the connection will be closed after completion of the response
//client.println("Refresh: 5");  // refresh the page automatically every 5 sec
client.println();
client.println("<!DOCTYPE HTML>");
client.println("<html>");

/**************************Script javascript para la contraseña*****************************************/
client.println(F("<script type=\"text/javascript\">"));
client.println(F("function passWord() {"));
client.println(F("  var pass1 = prompt('Introduzca la clave de acceso');"));
client.println(F("while(1) {"));
client.println("if(!pass1) history.go(-1)");
client.println("if(pass1.toLowerCase() == \"1234\") {");
client.println(F("alert('Bienvenido a Garden Configuration, pulsa aceptar para redireccionar.');"));
client.println(F("break;"));
client.println(F("}"));
client.println(F("var pass1 = prompt('Error, clave incorrecta.');"));
client.println(F("}"));
client.println(F("}"));
```

```
client.println(F("document.write(passWord());"));

client.println(F("</script>"));


/********************** Ahora viene la pagina ***********************************/


//////////////////TITULO//////////////////////

client.println(F("<head>"));

client.println(F("<title>GARDEN CONFIGURATION</title>"));

client.println(F("<link href=\"http://fonts.googleapis.com/css?family=Roboto:300|Playfair+Display:400\" rel=\"stylesheet\" type=\"text/css\"/>"));

client.println(F("<link rel=\"stylesheet\" href=\"http://static.tumblr.com/pjglohe/2qinf00ga/estilos.min.css\">"));

client.println(F("</head>"));




client.println(F("<body>"));


client.println(F("<div class=\"page-wrap\">"));

///Aqui el script para el cambio de color

client.println(F("<script type=\"text/javascript\">"));

client.println(F("function myFunction(elmnt,clr) {"));

client.println(F("elmnt.style.color = clr;"));
```

```
client.println(F("</script>"));


///Aqui el script para el cambio de color
client.println(F("<script type=\"text/javascript\">"));
client.println(F("function myFunction2(elmnt,clr) {"));
client.println(F("elmnt.style.color = clr;"));
client.println(F("</script>"));


//////////////////Titulo
client.println(F("<div align=\"center\" style=\"background: #eeeeee; border: 1px solid black;\">"));
client.println(F("<h1>Garden Configuration</h1>"));
client.println(F("</div>"));
client.println(F("<div class=\"device\">"));
//////////////////Tabla
client.println(F("<p> </p>"));
client.println(F("<div align=\"center\" style=\"background: #EEEEEE; border: 1px solid black;\">"));
client.println(F("</div>"));
client.println(F("<table border=\"0\" cellpadding=\"1\" cellspacing=\"1\" style=\"width: 100%; height: 708px;\">"));
client.println(F("<tbody>"));
//client.println(F("<div align=\"center\" style=\"background: #eeeeee; border: 1px solid black;\">"));
client.println(F("<tr>"));
```

```
client.println(F("<td align=\"center\" style=\"text-align: center; background: #EEEEEE; border: 1px solid black;\"><span style=\"font-size: 18px;\">Datos en tiempo real.</span></td>"));

client.println(F("<td align=\"center\" style=\"text-align: center; background: #EEEEEE; border: 1px solid black;\"><span style=\"font-size: 18px;\">Estado del sistema</span></td>"));

client.println(F("</tr>"));

//client.println(F("</div>"));

client.println(F("<tr>"));

client.println(F("<td>"));

client.println(F("<p style=\"margin-left: 40px;\"><u>Nodo 1:</u></p>"));


client.println(F("<ul>"));

client.println("<li style=\"margin-left: 80px;\">Temperatura: " + String(T1) );

client.println(F("&ordm;C</li>"));

client.println("<li style=\"margin-left: 80px;\">Humedad: " + String(T1) );

client.println(F("%</li>"));

client.println("<li style=\"margin-left: 80px;\">Presi&oacute;n: " + String(T1) );

client.println(F("bar</li>"));

client.println(F("</ul>"));


client.println(F("<p> </p>"));


client.println(F("<p style=\"margin-left: 40px;\"><u>Nodo 2:</u></p>"));
```

```
client.println(F("<ul>"));
client.println("<li style=\"margin-left: 80px;\">Temperatura: " + String(T2) );
client.println(F("&ordm;C</li>"));
client.println("<li style=\"margin-left: 80px;\">Humedad: " + String(T2) );
client.println(F("%</li>"));
client.println(F("</ul>"));

client.println(F("</td>"));
client.println(F("<td>"));
client.println(F("<p> </p>"));

////////////////////////////Los botones
client.println(F("<form action=\"\" method=\"get\">"));
client.println(F("<ul>"));
client.println(F("<li>Riego autom&aacute;tico:               <button size=\"33\" style=\"width: 47px; height: 23px; background-color: #B2CFFF; border: 1px solid #3F7CFF\" type='submit' name='ONman' value='01' onclick=\"myFunction(this,'red')& myFunction2(OFFman,'black')\"/>ON </button> <button size=\"33\" style=\"width: 47px; height: 23px; background-color: #B2CFFF; border: 1px solid #3F7CFF\" type='submit' name='OFFman' value='00' onclick=\"myFunction(this,'red') & myFunction2(ONman,'black')\">OFF</button></li>"));
client.println(F("</ul>"));
```

```
client.println(F("<ul>"));

client.println(F("<li>Riego autom&aacute;tico nodo 1:   <button size=\"33\" style=\"width: 47px; height: 23px; background-color: #B2CFFF; border: 1px solid #3F7CFF\" type='submit' name='ONn1' value='11' onclick=\"myFunction(this,'red')& myFunction2(OFFn1,'black')\"/>ON </button> <button size=\"33\" style=\"width: 47px; height: 23px; background-color: #B2CFFF; border: 1px solid #3F7CFF\" type='submit' name='OFFn1' value='10' onclick=\"myFunction(this,'red') & myFunction2(ONn1,'black')\">OFF</button></li>"));

client.println(F("</ul>"));


client.println(F("<ul>"));

client.println(F("<li>Riego autom&aacute;tico nodo 2:   <button size=\"33\" style=\"width: 47px; height: 23px; background-color: #B2CFFF; border: 1px solid #3F7CFF\" type='submit' name='ONn2' value='21' onclick=\"myFunction(this,'red')& myFunction2(OFFn2,'black')\"/>ON </button> <button size=\"33\" style=\"width: 47px; height: 23px; background-color: #B2CFFF; border: 1px solid #3F7CFF\" type='submit' name='OFFn2' value='20' onclick=\"myFunction(this,'red') & myFunction2(ONn2,'black')\">OFF</button></li>"));

client.println(F("</ul>"));


client.println(F("<ul>"));

client.println(F("<li>Riego autom&aacute;tico nodo 3:   <button size=\"33\" style=\"width: 47px; height: 23px; background-color: #B2CFFF; border: 1px solid #3F7CFF\" type='submit' name='ONn3' value='31' onclick=\"myFunction(this,'red')& myFunction2(OFFn3,'black')\"/>ON </button> <button size=\"33\" style=\"width: 47px; height: 23px; background-color: #B2CFFF; border: 1px solid #3F7CFF\" type='submit' name='OFFn3' value='30' onclick=\"myFunction(this,'red') & myFunction2(ONn3,'black')\">OFF</button></li>"));

client.println(F("</ul>"));
```

```
client.println(F("</br>"));



//////////////////////Dependiendo del string leido se realiza una accion
if ((peticion.indexOf("ONman=01") > 0) || (RiegoAll == true)) {
  RiegoAll = true;
  client.println(F("<ul>"));
  client.println(F("<li>Estado riego automatico: ON</li>"));
  client.println(F("</ul>"));
}

if ((peticion.indexOf("OFFman=00") > 0) || (RiegoAll == false)) {
  RiegoAll = false;
  client.println(F("<ul>"));
  client.println(F("<li>Estado riego automatico: OFF</li>"));
  client.println(F("</ul>"));
}

if ((peticion.indexOf("ONn1=11") > 0) || (Riego1 == true)) {
  Riego1 = true;
  client.println(F("<ul>"));
  client.println(F("<li>Estado riego zona 1: ON</li>"));
  client.println(F("</ul>"));
```

```
}

if ((peticion.indexOf("OFFn1=10") > 0) || (Riego1 == false)) {
  Riego1 = false;
  client.println(F("<ul>"));
  client.println(F("<li>Estado riego zona 1: OFF</li>"));
  client.println(F("</ul>"));
}

if ((peticion.indexOf("ONn2=21") > 0) || (Riego2 == true)) {
  Riego2 = true;
  client.println(F("<ul>"));
  client.println(F("<li>Estado riego zona 2: ON</li>"));
  client.println(F("</ul>"));
}

if ((peticion.indexOf("OFFn2=20") > 0) || (Riego2 == false)) {
  Riego2 = false;
  client.println(F("<ul>"));
  client.println(F("<li>Estado riego zona 2: OFF</li>"));
  client.println(F("</ul>"));
}

if ((peticion.indexOf("ONn3=31") > 0) || (Riego3 == true)) {
```

```
  Riego3 = true;
  client.println(F("<ul>"));
  client.println(F("<li>Estado riego zona 3: ON</li>"));
  client.println(F("</ul>"));
}

if ((peticion.indexOf("OFFn3=30") > 0) || (Riego3 == false)) {
  Riego3 = false;
  client.println(F("<ul>"));
  client.println(F("<li>Estado riego zona 3: OFF</li>"));
  client.println(F("</ul>"));
}

////////////////Resto de la pagina
client.println(F("</form>"));

client.println(F("<ul>"));
client.println(F("</ul>"));
client.println(F("</td>"));
client.println(F("</tr>"));
client.println(F("<tr>"));

client.println(F("<td style=\"text-align: center;\">"));
```

```
client.println(F("<p align=\"center\" style=\"text-align: center; background: #EEEEEE; border: 1px solid black;\"><span style=\"font-size: 18px;\"><span style=\"font-size: 18px;\">Descargar fichero con los datos</span></p>"));
client.println(F("<br>"));

client.println(F("<ul>"));
client.println(F("<li style=\"text-align: left;\"><a href=\"DATALOG.TXT\">"));
client.println(F("DATALOG.TXT"));
client.println(F("</a></li>"));
client.println(F("</ul>"));
client.println(F("<br>"));
client.println(F("</td>"));

if (peticion.indexOf("Kc1=") > 0) {
  int ind1 = peticion.indexOf("Kc1=");
  ValorKc1 = peticion.substring((ind1 + 4), (ind1 + 8));
  Kc1 = ValorKc1.toFloat();

  ValorKc1 = "";
}

if (peticion.indexOf("Kc2=") > 0) {
  int ind2 = peticion.indexOf("Kc2=");
  ValorKc2 = peticion.substring((ind2 + 4), (ind2 + 8));
  Kc2 = ValorKc2.toFloat();
```

```
  ValorKc2 = "";
 }

 client.println(F("<td style=\"text-align: center;\">"));
 client.println(F("<p align=\"center\" style=\"text-align: center; background: #EEEEEE; border: 1px solid black;\"><span style=\"font-size: 18px;\">Modificar valor de Kc</span></p>"));
 client.println(F("<form>"));
 client.println(F("Introducir Kc Nodo 1   <input type = \"text\" name = \"Kc1\"/>"));
 client.println("<input type = \"submit\" value = \"Enviar\"/>    Kc actual:" + String(Kc1) );
 client.println(F("<br><br>"));

 client.println(F("Introducir Kc Nodo 2   <input type = \"text\" name = \"Kc2\"/>"));
 client.println("<input type = \"submit\" value = \"Enviar\"/>    Kc actual:" + String(Kc2) );
 client.println(F("</form>"));
 client.println(F("<br>"));

 client.println(F("<a href=\"http://rrchtr.tumblr.com\" TARGET = \"_new\">"));
 client.println(F("Tabla de coeficientes del cultivo"));
 client.println(F("</a>"));
```

```
        client.println(F("</td>"));
        client.println(F("</tr>"));
        client.println(F("</tbody>"));
        client.println(F("</table>"));

        client.println(F("<p> </p>"));
        client.println(F("</body>"));
        client.println(F("</html>"));

      }

    }
    if (c == '\n') {
      currentLineIsBlank = true;
    } else if (c != '\r') {
      currentLineIsBlank = false;
    }

  }
}
delay(1);
peticion = "";
client.stop();
Serial.println("client disconnected");
```

```
    Ethernet.maintain();
  }
}
```