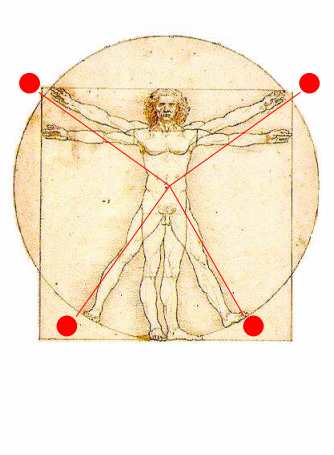
TECNOLOGÍ@ y *DESARROLLO*

Revista de Ciencia, Tecnología y Medio Ambiente

VOLUMEN VIII. AÑO 2011

SEPARATA



ARDUDROP 1.2: NUEVOS SENSORES Y MEJORAS EN LA ELECTRÓNICA DEL DISPOSITIVO

Miguel A. de Pablo Hdez. y C. de Pablo S.

UNIVERSIDAD ALFONSO X EL SABIO
Escuela Politécnica Superior
Villanueva de la Cañada (Madrid)

ISSN: 1696-8085

ARDUDROP 1.2: NUEVOS SENSORES Y MEJORAS EN LA ELECTRÓNICA DEL DISPOSITIVO

Miguel A. de Pablo Hdez.[1] y C. de Pablo S.

(1) Departamento de Geología. Universidad de Alcalá. Edificio de Ciencias. Campus Externo. Ctra. A-II km33,600. 28871 Alcalá de Henares, Madrid. España. miguelangel.depablo@uah.es

RESUMEN

El dispositivo ArduDrop ha sido desarrollado para el estudio de la humedad del suelo y su relación con parámetros ambientales. En este trabajo, centrándonos en el cálculo de la evapotranspiración, uno de los parámetros del balance hídrico, se establece la necesidad de incluir nuevos sensores al dispositivo: irradiancia y velocidad del viento, de los que se facilita una descripción. Por otro lado, la primera versión de este dispositivo se ha mejorado electrónicamente. La descripción de dichas mejoras es también el objetivo de este trabajo. Pero estas modificaciones también tienen por objeto el diseño de dos escudos para la placa Arduino Duemilanove que contiene el microcontrolador del dispositivo, con el fin de afianzar los distintos dispositivos y sensores acoplados al mismo, y mejorar el sistema de alimentación de los mismos. Estos escudos incluyen (1) un escudo para el almacenamiento de datos, con un Reloj de Tiempo Real, que además permite el establecimiento de alarmas, y una tarjeta de memoria SD; y (2) un escudo de conexión y alimentación de sensores, con bornes de tornillo donde conectar los distintos sensores periféricos del dispositivo. Finalmente se muestra el nuevo código del programa (Firmware) que permite adquirir los datos de los nuevos sensores, así como aprovechar las mejoras electrónicas para reducir el consumo de energía.

PALABRAS CLAVE: *Evapotranspiración, Escudos, Sensores, Open-hardware, Arduino.*

ABSTRACT:

ArduDrop device has been developed to study soil moisture and its relation to environmental parameters. In this work, focusing on the calculation of evapotranspiration, a water balance parameters, we justify that two new sensors should be included into the device: irradiance and wind speed, what we describe here. On the other hand, the first version of this device has been enhanced electronically. The description of these improvements is also the aim of this work. But these changes also aim to design two shields for Arduino Duemilanove containing the device microcontroller, in order to consolidate the various devices and sensors attached to it, and improve the power supply of them. These shields are (1) a shield for data storage, with a Real Time Clock, which also allows alarms configuration, and an SD memory card; and (2) a shield for sensors connection and power supply, including screw terminals which connect different peripheral sensors on the device. Finally, we show the new code of the program (Firmware) designed to acquire data from new sensors and electronic enhancements take to reduce energy consumption.
KEY-WORDS: *Evapotranspiration, Shields, Sensors, Open-hardware, Arduino.*

http://www.uax.es/publicaciones/archivos/TECMAD11_001.pdf

1. Introducción

El estudio de la humedad en el terreno tiene múltiples aplicaciones, como el control de la humedad en los regadíos, el estudio de la vegetación natural, el estudio de la hidrología superficial y subsuperficial, etc. La evolución de la humedad del suelo depende de distintos factores como las precipitaciones (o el riego), la evaporación, la transpiración a través de las plantas, o el agua que se infiltra hacia niveles más profundos del terreno. El dispositivo ArduDrop (de Pablo y de Pablo, 2010) es una herramienta de bajo coste para la toma de datos que permitan estudiar esta evolución de la humedad en el suelo con aplicaciones científicas y didácticas.

En general, el estudio de esta humedad en el suelo se basa en el balance hídrico, para lo que se requieren diversos tipos de datos (climáticos, hídricos, etc.). Entre otras cosas, este balance requiere conocer, no sólo cómo varía la humedad del agua en el suelo, sino otros parámetros ambientales como la cantidad del agua que se evapora por efecto del calentamiento del suelo por la radiación solar, y la cantidad de agua que se pierde en el suelo por haber sido extraído por parte de las plantas que se encuentran en ese lugar para su supervivencia. Profundizando en los distintos métodos existentes para el cálculo de estos datos, se ha llegado a la conclusión de que el dispositivo ArduDrop (de Pablo y de Pablo, 2010), para tener mayores aplicaciones en este campo, debería contar con dos nuevos sensores: irradiancia y velocidad del viento.

Por otro lado, y desde el punto de vista técnico, la primera versión funcional del dispositivo ArduDrop (de Pablo y de Pablo, 2010), requería un mantenimiento muy frecuente con reposición de baterías, a pesar de estar conectado a una placa solar. Este elevado consumo estaba asociado a: (1) el continuo consumo por parte de los sensores, (2) a la necesidad de mantener el microprocesador (ATmega328 en la placa Arduino Duemilanove) comprobando de manera continua la hora desde el reloj de tiempo real (RTC DS1307) para detectar el momento programado para realizar la medición de los sensores, y (3) a la baja tecnología del cargador de las baterías a través del panel solar.

De forma independiente, se detectó que existían diversas formas de mejorar y hacer más seguro el dispositivo, además de simplificar las tareas de mantenimiento y simplificar el manteniemiento. Así, se decidió mejorar la forma de conectar los sensores para (a) afianzar dichas conexiones, y (b) eliminar los dispositivos conectados al dispositivo, como la memoria microSD y el RTC, conectados a la placa sin un soporte. De esta forma, se evitará la posible desconexión accidental de esos componentes o su deterioro.

Por estos motivos se decidió realizar una serie de modificaciones en el dispositivo dirigidas a (1) incluir nuevos sensores que aumenten la funcionalidad del dispositivo, y (2) mejorar la electrónica del mismo de forma que se reduzca el consumo energético, y solucionen los inconvenientes detectados durante la fase de explotación en pruebas del mismo. En este trabajo se presenta el nuevo dispositivo ArduDrop 1.2 (Fig. 1), y (1) se justifica la necesidad de la instalación de nuevos sensores en el dispositivo, (2) se describen los dos nuevos sensores instalados, incluyendo sus componentes y circuito, (3) se explican las mejoras electrónicas llevadas a cabo, (4) se muestra el nuevo firmware (código del programa) para realizar las mediciones de los nuevos sensores y reducir el consumo de energía gracias a las mejoras electrónicas, y (5) las modificaciones en el fichero de salida de datos.

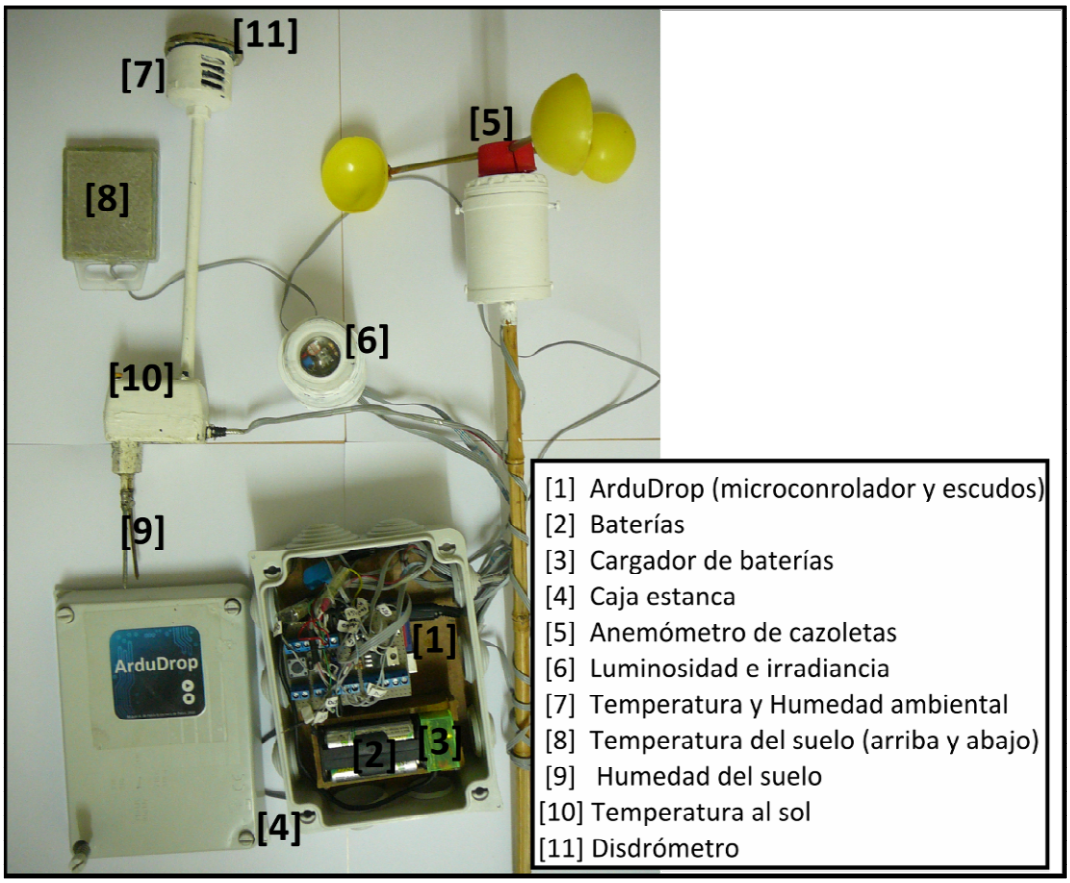


Fig. 1: Dispositivo ArduDrop en su versión 1.2 que se presenta en este trabajo y en el que se pueden observar las mejoras en la electrónica y los nuevo sensores instalados.

2. Determinación de la evapotranspiración

2.1. La evapotranspiración

El objetivo establecido para el dispositivo ArduDrop (de Pablo y de Pablo, 2010) es el estudio de la humedad del suelo con el fin realizar una adecuada gestión del agua, tanto con fines científicos como aplicados e incluso didácticos. Para realizar este estudio es fundamental cuantificar el efecto de la evaporación y la transpiración que reduce la cantidad de agua del terreno. Para ello, a continuación describimos de forma somera la evapotranspiración, su papel en la evolución de la humedad del suelo, las formas de obtener datos de evapotranspiración, los distintos métodos de cálculo y medición, y los sensores necesarios para adquirir dichos datos.

Adentrándonos levemente en este tipo de estudios de la humedad del suelo, tal y como se resume en el primer trabajo sobre este dispositivo (de Pablo y de Pablo, 2010), la evolución del agua en el subsuelo está condicionada por las características del terreno, por las características climáticas y por la vegetación existente, dando lugar a un balance hídrico simplificado con la siguiente expresión:

$$\text{Entradas} = \text{Salidas} \pm \text{Variación del agua almacenada en el terreno}$$

donde las entradas son las precipitaciones (lluvia, nieve, regadío, etc.), y las salidas la escorrentía superficial y la evapotranspiración, tal y como se refleja en la figura 2.

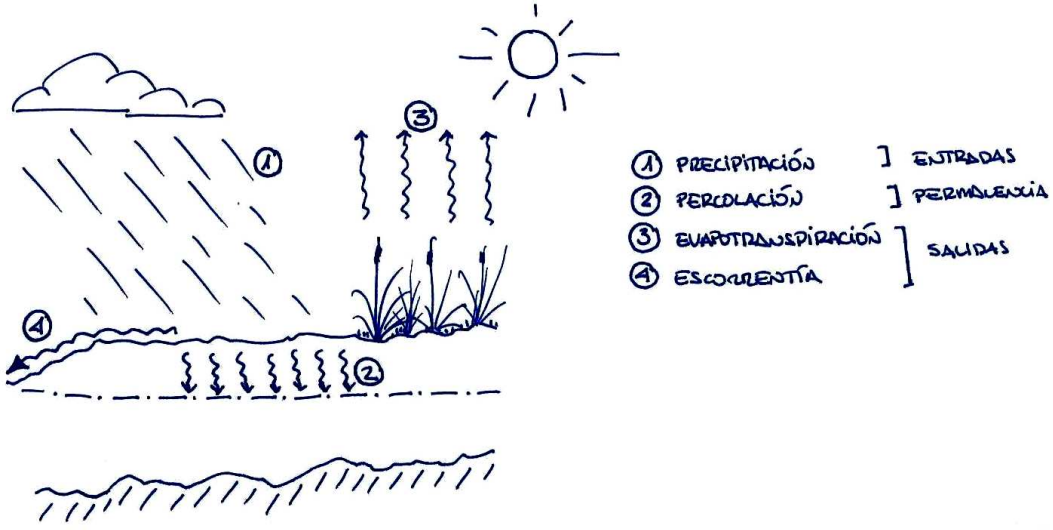


Figura 2: Esquema del balance hídrico en el terreno (de Pablo y de Pablo, 2010).

En este sentido, la vegetación juega un papel importante en el estudio de la evolución de la humedad del suelo, ya que las plantas son capaces de tomar agua del suelo para su alimentación. Y cada tipo de planta tiene un consumo diferente, que además varía en función de su fase de crecimiento, de la época del año, el clima y el momento del día. Esto refleja la complejidad de este tipo de estudios de la transpiración por parte de las plantas. Por otro lado, las propias condiciones climáticas hacen que el suelo sufra una cierta pérdida de agua por evaporación, independientemente de la pérdida que tenga por la transpiración de las plantas. Esta evaporación también varía en función de las condiciones climáticas, del periodo del año y del momento del día, además de por las propias características del suelo.

Todo esto hace que en definitiva sea complejo establecer las pérdidas reales de agua del suelo debidas a los procesos de evaporación y a los procesos de transpiración de forma independiente. Por ello generalmente se consideran de forma conjunta como evapotranspiración que, aunque sigue resultando compleja su determinación, es más sencilla que la de cada una de ellas por separado.

Para establecer la evapotranspiración existen multitud de métodos (ej., Thornthwaite, 1948; Penman, 1948; Blaney-Criddle, 1950; Turc, 1954, 1961; Makkink, 1957; Palmer and Havens, 1958; Pereira and Havens, 1958; Papadakis, 1961; Hammon, 1963; Priestly-Taylor, 1972; Hargreaves and Samani, 1985; Pereira and Pruitt, 2004), tanto empíricos (por ejemplo, basados en datos meteorológicos) como experimentales (basados, por ejemplo, en datos de lisímetros). Aunque no es el objeto de este trabajo discutir las diferentes metodologías posibles, sí queremos remarcar los fundamentos de algunas de ellas con el fin de justificar el método más sencillo y fiable que consideraremos de ahora en adelante en el diseño y configuración del dispositivo ArduDrop, con el objeto de adecuar los sensores de este dispositivo.

2.2. Métodos de medida y cálculo

Existen dos tipos fundamentales de métodos para establecer la evapotranspiración en un emplazamiento, la medición a través de diferentes experimentos, o mediante cálculos a partir de otros tipos de datos (generalmente climáticos) empleando ecuaciones empíricas válidas para distintas condiciones ambientales.

Dentro de los métodos experimentales, destacan dos, el lisímetro (a) y los tanques de evaporación (b). Con ellos se controla (a) la cantidad de agua que se moviliza en un determinado volumen de terreno (conociendo las entradas y midiendo las salidas por infiltración) de la que se deduce la cantidad de agua evapotranspirada; y (b) se mide la cantidad de agua evaporada de un tanque de agua de volumen conocido.

Por otra parte, los distintos métodos empíricos estudian el efecto de las condiciones climáticas en la evaporación, teniendo en cuenta ecuaciones físicas y datos procedentes de múltiples experimentos y de numerosas estaciones meteorológicas y agrarias repartidas por los distintos continentes. Todos estos métodos tienen en cuenta distintos factores y tienen distintos inconvenientes cuyo estudio no es el objeto de este trabajo.

Sin embargo, la Organización para la Alimentación y la Agricultura (FAO), dependiente de la Organización de las Naciones Unidas (ONU), estableció una metodología estándar basada en muchos de los métodos empíricos ya existentes, que permite realizar cálculos de la evapotranspiración con el fin de, en definitiva, calcular las necesidades de agua para diferentes tipos de cosechas en distintos tipos de terrenos. Aunque el método no es sencillo, sí es muy completo, que considera una gran variedad de factores, lo que, junto con su facilidad de aplicación y la posibilidad de solventar la ausencia de datos del emplazamiento estudiado con valores tabulados para diferentes condiciones y en distintos lugares del planeta, han hecho que lo elijamos con base para el estudio de la evapotranspiración y lo consideremos para la selección de los sensores más adecuados para instalar en el dispositivo ArduDrop. A continuación se describe brevemente los fundamentos de éste método.

2.3. El método FAO Monteith-Penman

El método FAO Monteith-Penman tiene por objeto el cálculo de la evapotranspiración de distintos tipos de cosechas en diferentes condiciones, no necesariamente las ideales. Para ello realiza, en primer lugar, el cálculo de la evapotranspiración de referencia (ET_0), que mide la cantidad de agua evapotranspirada por una parcela con una cosecha hipotética e ideal de 0.12 m de altura, con una resistencia superficial de 70 $s \cdot m^{-1}$, y un albedo de 0.23, con una extensión amplia, con un crecimiento activo continuado y regado adecuadamente (Allen et al., 1998), sometido a unas condiciones climáticas variables (las del lugar de estudio).

A partir de este valor de ET_0 el método permite calcular la evapotranspiración asociada a un tipo de cosecha determinado, pero también bajo buenas condiciones de regadío y suponiendo sus mejores condiciones agronómicas (ET_C), mediante el factor del tipo de cosecha (K_C) ya tabulado (Allen et al., 1998). Finalmente, se realiza el cálculo de la evapotranspiración real ajustada a las condiciones agronómicas reales a las que se ve sometida la cosecha (regadío escaso, suelos poco apropiados, etc.), empleando para ello un factor (K_S) también tabulado que tiene en cuenta todas estas condiciones, obteniendo así la evapotranspiración real asociada a una cosecha en unas condiciones determinadas ($ET_{C\,adj}$).

Así, el cálculo de la evapotranspitación de referencia o potencial (ET_0), es la base de los estudios, en este caso de las necesidades de agua de las distintas cosechas, pero también para cualquier estudio de la evapotranspiración asociada a cualquier lugar, ya que este método se basa en el cálculo a partir de datos ambientales, mediante la expresión:

$$ET_0 = \frac{0{,}408 \cdot (R_n - G) + \gamma \cdot \frac{900}{T + 273} \cdot u_2 \cdot (e_s - e_a)}{\Delta + \gamma \cdot (1 + 0{,}34 \cdot u_2)}$$

donde:

ET_0	Evapotranspiración potencial o de referencia [mm·día^{-1}]
R_n	Radiación neta incidente [MJ·m^{-2}·día^{-1}]
G	Flujo térmico del suelo [MJ·m^{-2}·día^{-1}]
T	Temperatura del aire a 2 metros de altura [ºC]
u_2	Velocidad del viento a 2 metros de altura [m·s^{-1}]
e_s	Presión de vapor en saturación [kPa]
e_a	Presión de vapor actual [kPa]
e_s-e_a	Déficit de presión de vapor [kPa]
Δ	Pendiente en la curva de presión de vapor [kPa·ºC^{-1}]
γ	Constante psicrométrica [kPa·ºC^{-1}]

Este método FAO Penman-Monteith (Allen et al., 1998) requiere de una serie mínima de datos para obtener el cálculo de la evapotranspiración potencial (ET_0), que son: a) Localización (Latitud, Longitud, Altitud) y b) Temperatura del aire. Además de éstos, resultan de gran interés otros datos ambientales como: c) Humedad relativa del aire, d) Radiación incidente, y e) Velocidad del viento. En caso de no disponer de dichos datos, en el propio manual de aplicación de éste método (Allen et al., 1998) se facilitan tablas de donde extraer estos datos, calculados de forma empírica a partir de datos de centenares de estaciones meteorológicas repartidas por todo el mundo, lo que resulta de gran interés para los estudios de suelo donde se carece de datos, o donde éstos no tienen la continuidad y/o calidad suficiente.

2.4. Sensores

Como se ha visto, sólo para el cálculo de uno de los términos de la ecuación del balance hídrico del suelo hace faltan una serie de datos mínimos que deberán ser adquiridos por distinto tipo de sensores. Algunos de ellos ya están disponibles en el dispositivo ArduDrop (de Pablo y de Pablo, 2010), pero existen otros que resultarían de interés instalar.

Entre los datos disponibles en la actualidad, se cuenta con la temperatura del aire (b), que es el parámetro fundamental y sin el cual resultaría imposible aplicar el método FAO Monteith-Penman. Este parámetro se mide a través de un sensor que mide temperatura y humedad ambiental (SH15), por lo que también queda cubierto otro de los parámetros relevantes para la aplicación del método, la humedad relativa del aire (c). Además, ArduDrop 1.0 contaba con un sensor de temperatura del aire al sol, constituido por un sensor de temperatura (DS18B20) sin protector de radiación solar.

El segundo de los parámetros necesarios para la aplicación del mencionado método, la localización (a), aunque resulta vital (especialmente en los casos en los que se quieran aplicar los parámetros empíricos tabulados en Allen et al. 1998), únicamente deben ser adquiridos una sola vez por emplazamiento, por lo que la instalación de un sensor GPS integrado en el dispositivo ArduDrop, elevaría los costes de éste, complicaría la electrónica y reduciría las posibilidad de instalar otros sensores dadas las limitaciones de la placa Arduino Duemilanove usada como base electrónica. Por este motivo, se ha establecido que los datos de localización del dispositivo sean adquiridos por el operador en el momento de la instalación en un emplazamiento, mediante el uso de dispositivos GPS, o extrayendo los datos necesarios (Latitud, Longitud y Altitud) a partir de mapas topográficos.

De este modo, existen dos parámetros que resultarían de gran interés para aplicar de forma más precisa el método FAO Monteith-Penman sin recurrir a datos tabulados y empíricos: la radiación incidente (d) y la velocidad del viento (e).

El dispositivo ArduDrop, en su versión 1.0, cuenta con un sensor de luminosidad (célula fotoresistente CdS -LDR- modelo PDV-P9203) que, aunque no puede ser empleado para calcular la radiación neta incidente, si permite aproximar el número de horas de luz y la iluminancia (en lux), que servirían como una primera aproximación y para el cálculo de la radiación neta incidente mediante distintas ecuaciones y tablas existentes para tal fin (ej., Allen et al., 1998). Respecto a la velocidad del viento, ArduDrop 1.0 no cuenta con ningún sensor para poder obtener este tipo de dato.

Aunque existen formas tabuladas de obtener estos datos, la filosofía del dispositivo ArduDrop es la de contar con los sensores necesarios para estudiar en detalle la humedad del suelo, manteniendo el coste reducido, la sencillez en su construcción y en su mantenimiento. Por este motivo se ha establecido el interés y necesidad de que la nueva versión del dispositivo ArduDrop (que se presenta en este trabajo, 1.2) cuente con sensores para medir radiación incidente y velocidad del viento.

3. Sensor de irradiancia

3.1. Sensores de irradiancia

Existen diversos sensores comerciales para la medición de la radiación incidente, ya sea radiación neta, incidente, reflejada, infrarroja, visible, etc. Sin embargo, el coste de los mismos es muy elevado. La reducción de costes hace necesario el empleo de sensores menos sofisticados que, aunque adquieran datos menos precisos o menos directos, permitan un buen acercamiento a la radiación incidente. Aunque existen valores tabulados de radiación incidente en función de la localización, época del año y momento del día (ej., Allen et al., 1998), estos datos medios varían de forma importante respecto a los valores empíricos derivados de esos métodos. Así que disponer de un sensor capaz de estimar de forma aproximada la radiación neta incidente, será de gran importancia para calcular la evapotranspiración, pero también para interpretar y discutir la evolución de la humedad del suelo.

3.2. TSL235R

Para medir la irradiancia nos hemos decantado por el uso de un sensor detector de luminosidad y conversión frecuencia/intensidad, el sensor TSL235R (comercializado por TAOS). Este sensor, requiere el uso de un único terminal de la placa Arduino y conlleva una electrónica muy sencilla (Figura 3), consistente en un condensador de 0.1uF entre sus terminales de alimentación (+5V y GND), situado lo más cerca posible del sensor. El tercer terminal del sensor va conectado directamente a un pin digital de la placa Arduino capaz de recibir interrupciones, ya sea por hardware (pins digitales 2 y 3) o por software (pin digital 5 y el uso de la librería Frequency Counter, desarrollada por M. Nawrath - Link 1). Esta última es la configuración empleada en el dispositivo Ardudrop 1.2 (Anexo 1).

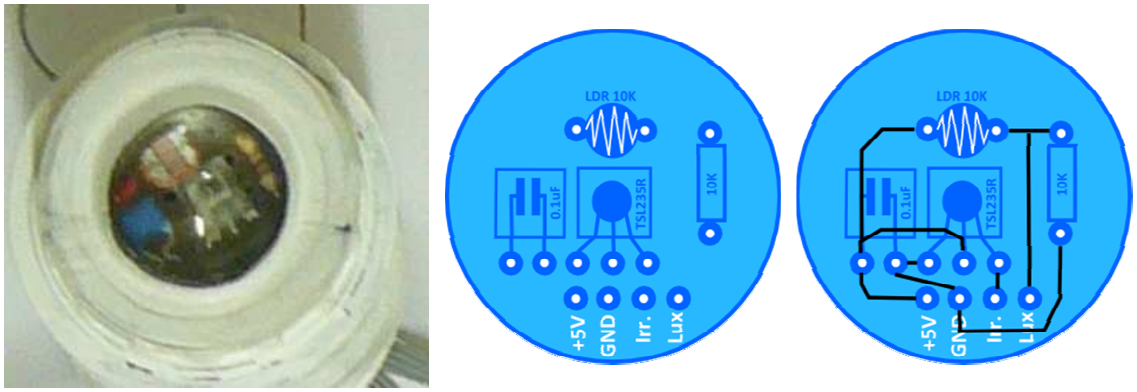


Fig. 3: Montaje de los sensores de irradiancia y luminosidad y circuito electrónico.

3.4. Código

Mediante el uso de la librería "Frequency Counter", la forma de medición de la irradiancia a través de este sensor se realiza mediante el conteo de las oscilaciones producidas por el sensor (y que son proporcionales a la intensidad de la luz recibida y de la longitud de onda), en un espacio de tiempo determinado (100 milisegundos para una resolución de 100 Hz). Teniendo en cuenta el amplio rango de longitudes de onda que pueden ser detectadas por este sensor, y de sus características técnicas, se asume, que 1kHz es equivalente a $1\mu W/cm^2$. Aunque el sensor tiene cierta dependencia de la temperatura ambiente, las variaciones son menores del 1%, por lo que se aceptan estos rangos de error sin necesidad del cálculo, por código, del rango de la medición resultante. El Anexo 1 contiene el código empleado para el cálculo de la irradiancia ($\mu W/cm^2$) mediante el dispositivo ArduDrop 1.2.

4. Sensor de velocidad del viento

4.1. Anemómetro de cazoletas y sensor QRD1114

Existen diferentes tipos de sensores comerciales para determinar la velocidad del viento (anemómetros). Por su sencillez, se ha seleccionado desarrollar un anemómetro de cazoletas, y medir las revoluciones mediante las interrupciones generadas por un disco sobre un haz lumínico (en este caso infrarrojo), usando para ello los pins capaces de detectar interrupciones de la placa Arduino Duemilanove (pins 2 y 3).

El sensor seleccionado es el QRD1114, que conlleva una electrónica muy sencilla al tratarse de un led infrarrojo y un fotodetector (Figura 4). Este sensor tiene una respuesta de 10 microsegundos, por lo que para producir las interrupciones se ha recurrido a un disco codificado (Blanco/Negro) sujeto al eje de rotación del anemómetro. La parte mecánica del dispositivo (Figura 4) consiste en un sistema de cazoletas elaboradas son semiesferas de plástico endurecido de 43 mm de diámetro, unidas a un brazo metálico de varilla de latón de 3 mm de diámetro y 69 mm de longitud. Se optó por un sistema de tres cazoletas, cuyos brazos están unidos entre sí a un eje central, al que se encuentra atornillado. El eje central está constituido por una varilla de igual diámetro que las anteriores, pero con una longitud de 30 mm. En la parte inferior del eje se ha dispuesto un disco de metacrilato de 30 mm de diámetro y 2 mm de espesor, sobre el que se pegó un disco de papel codificado con 2 sectores (blanco y negro) de igual área (la mitad del disco). El eje se aloja en el interior de un tapón de PVC para tubo de 40 mm de diámetro, que cuenta con una junta estanca. Para permitir el giro del eje central del anemómetro se instalaron dos rodamientos de bolas, uno situado en la parte superior, y otra en la parte inferior del eje, sujeto a dos discos de metacrilato creados como tapa inferior del dispositivo.

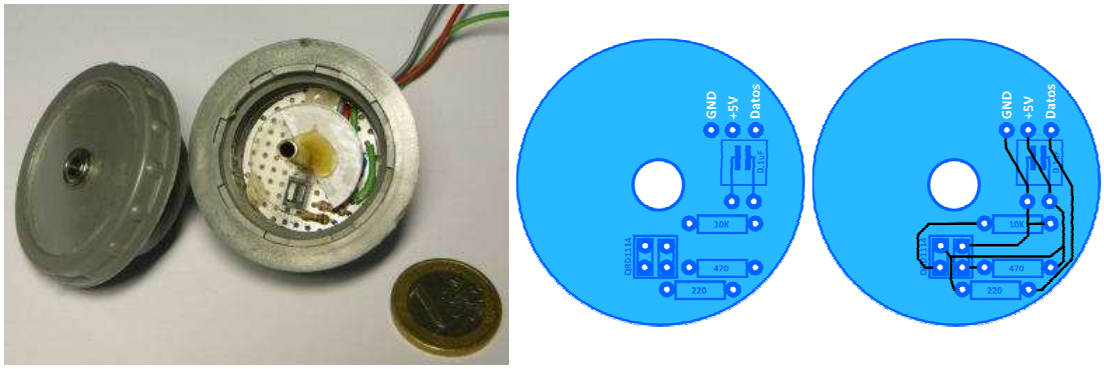


Fig. 4: Anemómetro de cazoletas basado en el sensor QRD1114 y circuito electrónico.

5. Mejoras en la electrónica

Las mejoras electrónicas realizadas al dispositivo ArduDrop han implicado el desarrollo de dos "escudos" (del inglés "shield") para la placa base del dispositivo (Figura 5), la placa Arduino Duemilanove. Estos escudos no son más que placas con diferentes componentes que se conectan a la placa Arduino Duemilanove a través de todos sus pins, tomando la alimentación de ella, y permitiendo superponer otros escudos de forma apilada.

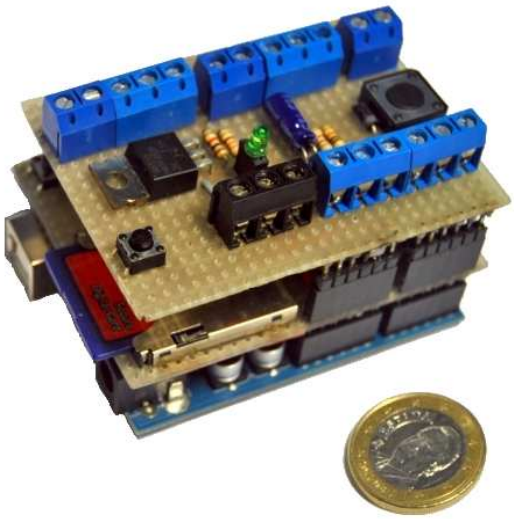


Fig. 5: Placa base (Arduino Duemilanove, en la parte inferior) del dispositivo ArduDrop 1.2, conectado a los prototipos de los escudos de almacenamiento de datos (sobre la placa base), y de conexión de sensores (en la parte superior).

Existen diferentes escudos comerciales desarrollados para la adquisición y almacenaje de datos (Link 2, Link 3), así como para la conexión de dispositivos satélite a través de bornes de tornillo que afianzan dichas conexiones (Link 4). Inspirándonos en estos últimos escudos, y en otros desarrollados por diferentes autores hemos desarrollado dos escudos dirigidos a (1) toma de datos y (2) conexión y alimentación de sensores, que describiremos más adelante en este trabajo. Estos escudos hacen que el dispositivo ArduDrop 1.2 aumente de tamaño, pero todos los componentes (tarjeta de memoria y RTC) queden integrados en el propio dispositivo, y los sensores bien afianzados al mismo. A continuación se describen cada uno de estos escudos, sus objetivos concretos, los componentes necesarios para su construcción, el circuito requerido.

6. Escudo de almacenamiento de datos (Datalogger Shield)

6.1. Descripción general

Este primer escudo (Figura 6) tiene por objetivos: (1) disponer de un RTC capaz de disparar alarmas en los momentos establecidos por código para realizar la toma de datos, y (2) disponer de una tarjeta de memoria flash SD de alta capacidad para almacenar los datos registrados. Con él se pretende reducir el consumo del dispositivo al hacer posible configurar el microcontrolador para que se mantenga en estado de letargo hasta que es despertado por el RTC mediante una alarma configurada inicialmente por código (o de forma manual mediante un botón instalado en el escudo de conexión y alimentación de sensores, como se describirá más adelante en este trabajo). Para alcanzar el mínimo consumo hemos establecido emplear la opción de mínimo consumo (desconexión de alimentación: Power_Down).

Por otro lado, el integrar una tarjeta de memoria flash SD dentro de este mismo escudo, sustituyendo al componente comercial con el que se contaba en la primera versión del dispositivo ArduDrop (de Pablo y de Pablo, 2010), hace que este dispositivo sea más estable, reduce los pins empleados de 5 a 3, ya que pasa a alimentarse por el pin de 5V y no por un pin I/O digital, y manteniendo su sistema de comunicaciones SPI pins 10 a 13.

6.2. Componentes

Los componentes necesarios para desarrollar este escudo (Tabla 1) son pocos, económicos y, en general, sencillos de obtener en las tiendas de electrónica. Los componentes esenciales para desarrollar este escudo son (1) un RTC DS1337+, comercializado por Maxim (Link 5), y (2) un zócalo para tarjetas de memoria flash de formato SD/MMC (Link 6).

Tabla 1: listado de componentes empleados en el desarrollo del escudo datalogger

3 resistencia 3,3 KΩ carbono 5% 1/4W	1 RTC DS1337
3 resistencia 10K Ω carbono 5% 1/4W	2 terminales macho/hembra de 6 pins
3 resistencia 1,8 KΩ carbono 5% 1/4W	2 terminales macho/hembra de 8 pins
1 zócalo para circuito integrado 8 pins	1 tarjeta de memoria SD
1 zócalo para tarjeta SD/MMC	1 condensador cerámico 0.1µF
1 zócalo pila de botón	1 oscilador de cristal de cuarzo 32,768 kHz
1 pila botón litio 3V CR2032	Cables para prototipado 0,5 mm diámetro
2 diodos 1N4001	Placa microperforada de prototipazo
1 botón de placa pequeño	

6.3. Circuito

El circuito electrónico que lleva asociado este escudo no es complejo, pero al disponer de numerosos componentes y algunos de ellos de gran tamaño, el resultado es algo más laborioso que el del escudo de conexión y alimentación de sensores.

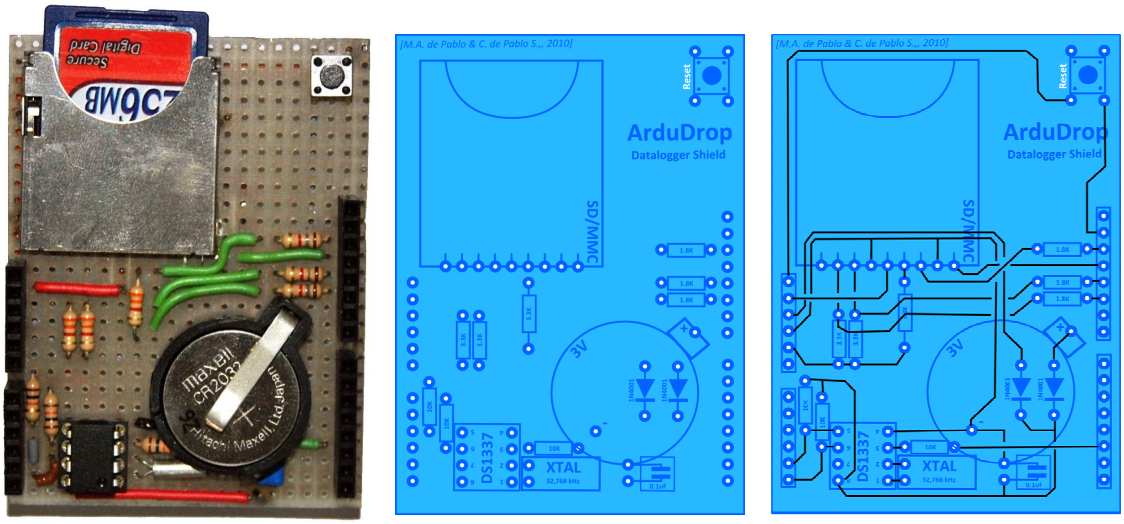


Figura 6. Escudo de almacenamiento de datos, componentes y circuito electrónico.

Una consideración importante es que el RTC empleado no dispone de opción de alimentar el dispositivo mediante baterías de respaldo para cuando el dispositivo se encuentre desconectado y evitar la pérdida de su configuración. Para solventar este problema, se ha integrado un pequeño circuito de alimentación alternativa para el RTC mediante una pila de botón de ión Litio de 3V (CR2032), conectada al RTC, que también se alimenta del pinde 5V de la placa Arduino. La placa también dispone de un botón de reinicio.

7. Escudo de conexión y alimentación de sensores (sensors shield)

7.1. Descripción general

El escudo para la conexión y alimentación de sensores pretende (1) afianzar las conexiones de los sensores evitando desconexiones accidentalmente; (2) reducir el consumo de energía haciendo que los sensores se alimenten únicamente durante el momento en el que realizarán las mediciones; (3) la activación manual del sistema; y (4) permitir la medición del voltaje de las baterías que alimentan el dispositivo.

El primer objetivo se consigue mediante el uso de bornes de tornillo, conectados a todos los pins I/O excepto el pin digital 2, empleado para recibir alarmas desde el RTC instalado en el escudo datalogger previamente descrito, y el pin digital 8, empleado para disparar el circuito de alimentación de los sensores. Este circuito de alimentación se realiza mediante el uso de un elemento mosfet (TP510) activado, como se ha dicho, a través del pin digital 8. De esta manera, aunque los sensores se encuentren conectados a los bornes de alimentación, éstos no reciben corriente hasta que se activa por código el pin digital 8. Para conocer de manera visual si los sensores están siendo alimentados, se ha dispuesto un circuito con un led conectado también al pin digital 8.

Una tercera funcionalidad es la de activar el dispositivo ArduDrop, de forma similar a como hacen las alarmas del RTC instalado en el escudo previamente descrito, mediante un botón conectado al pin digital 2. El objetivo de este botón es poder sacar al dispositivo de su letargo sin esperar a la hora prefijada para realizar las mediciones. Y finalmente, como este escudo cubre la totalidad de la placa Arduino, se ha instalado en esta placa un nuevo botón de reinicio de fácil acceso.

En este escudo se optó por instalar un circuito para el control del nivel de la alimentación del sensor, mediante un divisor de voltaje, similar a los empleados en otros proyectos desarrollados por diferentes autores (Link 7). Este circuito, consistente en dos resistencias y un condensador, está conectado a pin Vin, que tiene acceso directo al voltaje de alimentación del dispositivo Arduino. Los datos, junto con los de voltaje y temperatura internos de la placa Arduino (ya configurados por código en la versión 1.0 del dispositivo – de Pablo y de Pablo, 2010), permitirán el control de fallos del dispositivo y el seguimiento del consumo energético.

7.2. Componentes

Los componentes necesarios para desarrollar este escudo (Tabla 2) son muy pocos, económicos y sencillos de obtener en las tiendas de electrónica.

Tabla 2: listado de componentes empleados en el desarrollo del escudo de sensores

1 mosfet TP510	4 terminal de tornillo de 3 cables
1 resistencia 1KΩ carbono 5% 1/4W	3 terminales de tornillo de 2 cables
1 resistencia 330 Ω carbono 5% 1/4W	1 led verde 3 mm diámetro
1 resistencia 8,2 KΩ carbono 5% 1/4W	1 botón pequeño
1 resistencia 10 KΩ carbono 5% 1/4W	1 botón grande
1 condensador de 10µF	Cables para prototipado 0,5 mm diámetro
1 diodo 1N4001	Placa microperforada de prototipado
1 terminal hembra de 2 pin	

7.3. Circuito

El circuito electrónico que lleva asociado este escudo (Figura 7) es sencillo, ya que, como se ha descrito previamente, pretende la comunicación entre los distintos pins de entrada y salida (I/O) de la placa Arduino, tanto digitales como analógicos. Únicamente incluye cuatro sencillos circuitos para el sistema de alimentación para el led de aviso visual de actividad, el control de la alimentación y un último circuito para el botón de activación.

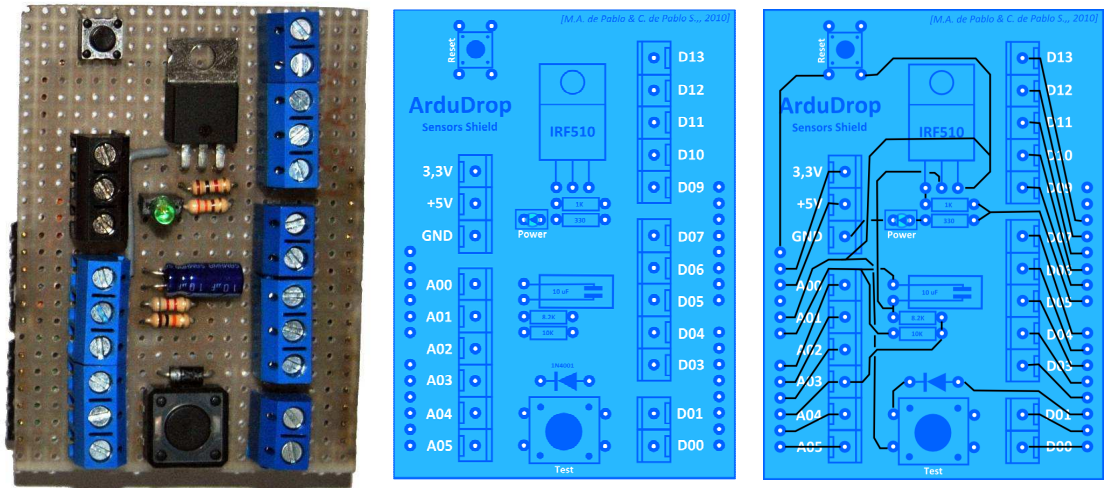


Figura 7: Escudo de conexión y alimentación de sensores, componentes y circuito.

8. Firmware

Las mejoras electrónicas introducidas en el dispositivo y los nuevos sensores instalados en el dispositivo, requieren a su vez de profundas modificaciones en el firmware, y el almacenamiento de más datos en la tarjeta de memoria. Por ello, en este trabajo también se incluye un nuevo firmware (Anexo 1) que contiene las modificaciones necesarias para incluir la adquisición de datos por parte de los dos nuevos sensores, y su almacenamiento en la tarjeta de memoria, permitiendo, además, el poner el microcontrolador en estado de letargo para reducir el consumo, y activando la alimentación de los sensores sólo instantes antes de realizar las mediciones. Este nuevo firmware incluye la realización de medidas de la irradiancia y de la velocidad del viento a través de los nuevos sensores, y su integración entre los datos adquiridos por el dispositivo, y que también incluye el control del voltaje de la alimentación del dispositivo. Todos estos datos se graban en un fichero ASCII en la tarjeta de memoria SD instalada, en esta nueva versión 1.2 del dispositivo ArduDrop, en el escudo desarrollado para tal fin.

Cuadro1: Pseudocódigo del firmware 0.7 para el dispositivo ArduDrop 1.2.

1. Declaración de librerías
2. Definición de pin analógicos
3. Definición de pins digitales
4. Activación y configuración de librerías
5. Definición de constantes
6. Definición de variables
7. Configuración de las librerías
8. CONFIGURACIÓN INICIAL
    - 4.1. Configuración de los pins
    - 4.2. Activación del RTC
    - 4.3. Establecimiento y activación de la alarma de medición de sensores
    - 4.2. Configuración de la comunicación por el puerto serie (*)
    - 4.3. Mostrar la pantalla de inicio en el puerto serie (*)
9. DESARROLLO DEL PROGRAMA
    - 9.1. Despertar de la fase de letargo
    - 9.2. Activar la alimentación de los sensores
    - 9.3. Realizar la medida a través de todos los sensores
    - 9.4. Grabación los datos en la memoria flash SD
    - 9.5. Muestra los datos a través del puerto serie (*)
    - 9.6. Desconecta la alimentación de los sensores
    - 9.7. Entra en estado de letargo

Explicado de forma general, el funcionamiento del dispositivo (Cuadro 1) se simplifica de forma importante respecto al de la primera versión del dispositivo (de Pablo y de Pablo, 2010) gracias al nuevo firmware (Anexo 1), y a la entrada en estado de letargo del microcontrolador. Esto hace que el dispositivo no tenga que comprobar continuamente la hora y hacer parpadear un led mientras espera al siguiente momento para la realización de las medidas. Por el contrario, ahora el dispositivo configura la alarma del RTC, conectado a través del escudo previamente descrito, al inicializar el sistema, realiza una primera medición y entra en modo letargo hasta que recibe una alarma a través del RTC. Cuando recibe la alarma, realiza una nueva medida a través de todos los sensores y vuelve al estado de retardo, sin realizar ninguna otra actividad entre mediciones consecutivas.

El código completo del firmware se muestra en el Anexo 1, y muestra las ecuaciones empleadas para los diferentes cálculos necesarios para el procesado de los datos para la obtención de los datos las respectivas unidades del Sistema Internacional. También puede verse las distintas operaciones necesarias por poner el microcontrolador en estado de letargo y su posterior activación, así como para la alimentación de los distintos sensores.

Cuadro 2: Ejemplo de datos grabados en formato ASCII en el fichero "datos.csv", en la tarjeta de memoria SD.

```
#;Fecha;Hora;Bateria;Vin;Tin;Tamb;Hamb;TRocio;Irradiancia;TExt;HSuelo;T_Ar;T_Ab;Lluvia;Luminosidad;Viento
1;20/12/2010;0:00:00;7,84;5,05;22,89;6,15;92,76;5,07;17;6,15;35;4,56;4,68;0;0;0,00
2;20/12/2010;1:00:00;7,82;5,05;23,00;6,15;92,76;5,07;18;6,15;34;4,68;4,75;0;0;0,00
3;20/12/2010;2:00:00;7,74;5,02;23,32;6,15;92,76;5,07;23;6,15;27;5,06;4,93;0;0;0,00
4;20/12/2010;3:00:00;7,61;5,02;23,54;4,86;89,42;3,26;18;4,86;38;4,68;4,81;0;0;0,00
5;20/12/2010;4:00:00;7,52;5,02;23,75;4,86;89,42;3,26;16;4,86;42;5,50;5,06;0;0;0,00
6;20/12/2010;5:00:00;7,47;5,05;23,86;4,86;89,42;3,26;15;4,86;29;7,62;6,56;0;0;0,00
7;20/12/2010;6:00:00;7,47;5,02;24,07;4,86;89,42;3,26;16;4,86;36;10,81;8,37;0;73;0,00
8;20/12/2010;7:00:00;7,44;5,02;23,64;9,79;66,65;3,88;19;9,79;36;10,62;9,18;0;272;0,00
9;20/12/2010;8:00:00;7,34;5,02;23,32;9,79;66,65;3,88;25;9,79;36;11,00;10,43;0;456;0,00
10;20/12/2010;9:00:00;7,26;5,05;23,00;9,79;66,65;3,88;10983;9,79;34;10,87;10,50;0;737;0,00
11;20/12/2010;10:00:00;7,21;5,02;22,89;9,34;65,31;3,15;74896;9,34;37;9,43;9,68;0;803;0,00
12;20/12/2010;11:00:00;7,15;5,05;22,79;9,29;65,99;3,26;95232;9,29;32;9,37;9,43;0;1016;0,00
13;20/12/2010;12:00:00;7,10;5,05;22,57;15,34;76,66;11,27;95010;15,34;40;17,43;17,12;0;651;0,00
14;20/12/2010;13:00:00;7,06;5,05;22,57;18,56;68,13;12,56;94925;18,56;42;19,62;18,68;0;268;0,00
15;20/12/2010;14:00:00;7,02;5,05;22,57;19,15;59,94;11,19;94513;19,15;47;19,56;20,00;0;168;0,00
16;20/12/2010;15:00:00;6,99;5,05;22,57;19,74;56,38;10,81;94242;19,74;35;19,87;19,81;0;15;0,00
17;20/12/2010;16:00:00;6,97;5,05;22,57;20,13;54,89;10,77;93910;20,13;33;19,81;19,87;0;13;0,00
18;20/12/2010;17:00:00;6,94;5,05;22,57;21,17;52,81;11,15;94357;21,17;34;19,68;19,87;0;7;0,00
19;20/12/2010;18:00:00;6,92;5,05;22,57;20,77;52,81;10,79;94757;20,77;36;19,62;19,81;0;8;0,00
20;20/12/2010;19:00:00;6,89;5,05;22,36;20,92;51,74;10,61;963;20,92;38;19,56;19,81;0;8;0,00
21;20/12/2010;20:00:00;6,88;5,05;22,36;21,86;59,52;13,62;40;21,86;36;19,56;19,81;0;10;0,00
22;20/12/2010;21:00:00;6,86;5,05;22,36;19,72;54,99;10,42;40;19,72;35;19,50;19,75;0;10;0,00
23;20/12/2010;22:00:00;6,86;5,05;22,46;19,68;55,01;10,39;34;19,68;33;19,43;19,68;0;10;0,00
24;20/12/2010;23:00:00;6,85;5,05;22,36;19,65;55,10;10,38;42;19,65;34;19,40;19,70;0;8;0,00
```

9. Salida de datos

Respecto a la salida de datos (Cuadro 2), se han realizado algunas modificaciones importantes respecto a la versión anterior del dispositivo y del firmware. En primer lugar se ha separado la cabecera del fichero de datos, en el que se explican los distintos datos grabados y sus unidades, que queda grabado en un fichero ASCII independiente. Por otro lado el fichero de datos contiene únicamente una fila con el nombre de cada uno de los datos grabados. Este fichero de datos, en formato ASCII, pasa a tener extensión CSV, lo que junto con la grabación de los datos separados por (;) en vez de en por espacios, permite que pueda ser importado de forma inmediata por programas informáticos dirigidos a la gestión de hojas de cálculo (ej., Excel) o a la realización de gráficas (ej., KST, Link 8). Por otro lado, al fichero de datos se le han añadido tres nuevas columnas correspondientes a los datos de irradiancia (mW/cm^2), a la velocidad del viento (m/s) y al nivel de batería (v).

10. Futuras mejoras

Como futuras mejoras se pretende el desarrollo de nuevos sensores para la medición de la humedad y temperatura del suelo, combinando ambos sensores, y creando tres de ellos para que el estudio de la humedad del suelo (objetivo principal de éste dispositivo) sea más completo y permita un estudio más detallado de la humedad, también en profundidad.

Por otro lado se contempla la mejora del sistema de alimentación, basado en el uso de paneles solares, permitiendo una carga eficiente de las baterías. Este cargador se desarrollará según las necesidades que se comprueben que requiere el dispositivo tras el periodo de pruebas en el que se encuentra en la actualidad tras su actualización a la versión 1.2, incluyendo todas las mejoras y modificaciones presentadas en este trabajo.

Además, se expandirá la memoria del dispositivo añadido una memoria EEPROM de 32Kb, mediante el uso del circuito integrado 24LC256. Esta memoria no volátil, aunque puede emplearse para el almacenamiento de datos, servirá principalmente para la grabación de parámetros de la configuración inicial del dispositivo, de forma que éstas no se pierdan ante una eventual pérdida de alimentación, y que permitan simplificar en mayor medida el código (firmware) del dispositivo. La memoria EEPROM permitirá el cálculo de valores medios diarios y semanales antes de ser grabados en la memoria SD.

Finalmente, se plantea como un reto futuro el desarrollo de una caja estanca para el alojamiento de la electrónica y las baterías con su cargador, sujeta aun mástil sobre el que irán los distintos sensores de luminosidad, lluvia, velocidad del viento y temperatura y humedad ambiental, además del soporte para la placa solar que recargue las baterías.

11. Conclusiones generales

El dispositivo ArduDrop 1.2 que aquí se ha presentado es una importante mejora respecto a la versión anterior, no sólo en la parte electrónica, sino en los datos que adquiere, permitiendo un estudio más detallado de la evolución de la humedad del suelo. En la actualidad el dispositivo dispone de una amplia variedad de sensores ambientales suficientes para los cálculos de la evapotranspiración. En el futuro este equipo de trabajo se centrará en el desarrollo de mejores sensores de la humedad del suelo lo que finalmente permitirá desarrollar estudios muy detallados.

12.- Agradecimientos

Este trabajo ha sido posible gracias a la plataforma libre (open-hardware) Arduino, así como al entorno de desarrollo Arduino 017 (*www.arduino.cc*), pero especialmente a la comunidad de usuarios de Arduino a través de su foro. Algunas partes del código han sido desarrolladas gracias a discusiones y debates con otros usuarios de la plataforma Arduino, y gracias también a los tutoriales y librerías de código desarrollados para los distintos sensores. Nuestro especial agradecimiento a los asistentes a la I Arduino Barcamp (*https://arduinobarcamp.jottit.com/*), por la aportación de ideas que realizaron y, en especial a José Manuel Amuedo y Antonio Pascual, por sus ideas sobre el uso de RTC y de elementos mosfet para reducir el consumo energético del dispositivo, respectivamente. Finalmente, los autores deseamos agradecer muy especialmente a nuestra familia su paciencia por las horas que le hemos dedicado al desarrollo y mejora del dispositivo.

13.- Referencias

ALLEN, R.G., PEREIRA, L.S., RAES, D., SMITH, M. (1998) "*Crop evapotranspiration – guidelines for computing crop water requirements*" Irrigation and drainage paper, 56. FAO, Rome, Italy. 300 pp.

BLANEY, H.F. and CRIDDLE, W.D. (1950). "*Determining water requeriments in irrigated areas From climatological and irrigation data*". U.S. Dep. Agri. S.C.S.-TP – 96.44P.

DE PABLO H., M.A. y DE PABLO S., C. 2010. "*Ardudrop 1.0: Dispositivo electrónico para el estudio de la humedad del suelo*". Tecnologí@ y Desarrollo, 10: 06. 31 pp.

HAMON, W.R. (1963) "*Computation of direct runoff amounts from storm rainfall*". Int. Assoc. Sci. Hydrol. Pub. 63:52-62.

HARGREAVES, G.H. and SAMANI, Z.A. (1985) "*Reference crop evapotranspiration from temperature*". Applied Engineering in Agriculture, 1(2):96-99.

MAKKINK, G.F. (1957) "*Testing the Penman formula by means of lysimeters*". J. Inst. of Water Eng. 11:277-288.

PALMER, W.C., and HAVENS, H.A. (1958) "*A graphical technique for determining evapotranspiration by the Thornthwaite method*". Monthly Weather Review, 86. 123-128.

PAPADAKIS, J. (1961). "*Climates of the World*". Buenos Aires.

PENMAN, H.L. (1948) "*Natural evaporation from open water, bare soil and grass*". Proc. Roy. Soc. London, A193:120-146.

PEREIRA, A.R., PRUIT, W.O. (2004). "*Adaptation of the Thornthwaite scheme for estimating daily reference evapotranspiration*".Agricultural Water Management, 66. 251–257.

PRIESTLEY, C.H.B. and TAYLOR, J.R. (1972) "*On the assessment of surface heat flux and evaporation using large scale parameters*". Mon. Weath. Rev. 100:81-92.

THORNTHWAITE, C.W. (1948) "*An approach toward a rational classification of climate*". Geographic Review, 38. 55-94.

TURC, L. (1954) "*Le bilan d´eau des sols*". Ann.Agron 5. 491-569

TURC, L. (1961) "*Evaluation de besoins en eau d'irrigation, ET potentielle*". Ann. Agron. 12:13-49.

14.- Direcciones web

1: http://interface.khm.de/index.php/lab/experiments/arduino-frequency-counter-library/
2: http://www.ladyada.net/make/logshield/lighttemp.html
3: http://www.instructables.com/id/Logger-Shield-Datalogging-for-Arduino/
4: http://wingshieldindustries.com/products/screwshield/
5: http://www.maxim-ic.com/datasheet/index.mvp/id/3128
6: http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1206874649/8
7: http://www.janspace.com/b2evolution/arduino.php/2010/06/26/scooterputer
8: http://kst-plot.kde.org/

ANEXO 1 Código comentado del Firmware versión 0.7 para el dispositivo Ardudrop 1.2

```
/*  ArduDrop 1.2
 *
 *  Dispositivo para el estudio de la evolución de la humedad
 *  en el suelo y su relación con parámetros ambientales
 *
 *  Autores: M.A. de Pablo Hdez. y C. de Pablo S.
 *  Hardware: 1.2 20101209
 *  Firmware: 0.8 20101209
 */

// Definición de las librerías a utilizar
#include <FileLogger.h>                 // Librería para almacenamiento de datos en  memoria SD
#include <DallasTemperature.h>          // Librería para el sensor de temperatura DS18B20
#include <OneWire.h>
#include <Wire.h>
#include <DS1337.h>                     // Librería para el RTC DS1307
#include <Sensirion.h>                  // Librería para el sensor de temperatura y humedad SH15
#include <FreqCounter.h>                // Librería para el sensor de irradiancia TSL235R
#include <avr/power.h>
#include <avr/sleep.h>                  // Librería para modo letargo del microcontrolador

// Definición de los pin analógicos
#define luz 0                           // Luminosidad LDR
#define suelo 1                         // Humedad del suelo
#define lluvia 2                        // Intensidad de la lluvia
#define bateria 3                       // Voltaje de entrada

// Definición de los pin digitales
#define alarma 2                        // Alarmas del RTC DS1337
#define viento 3                        // Anemómetro QRD1114
#define tempsuelo 4                     // Temperatura del suelo DS18B20
#define irradiancia 5                   // Irradiancia TSL235R
#define temphumdat 6                    // Temperatura/humedad/Punto de rocío SH15
#define temphumcl 7                     // Temperatura-humedad SH15
#define power 8                         // Alimentación de los sensores
#define temp 9                          // Temperatura al sol DS18B20

// Activación de librerías
DS1337 RTC = DS1337();                  // Configuración librería RTC DS1337
OneWire oneWire(temp);                  // Configuración librería termómetro DS18B20
DallasTemperature sensors(&oneWire);
OneWire oneWire2(tempsuelo);            // Configuración librería termómetro DS18B20
DallasTemperature sensors2(&oneWire2);
Sensirion SH15 = Sensirion(temphumdat, temphumcl);
```

```
// Definición de constantes
const float pi = 3.14159265;              // Numero PI
const float volt = 0.0108480556;          // Resolución voltaje de entrada
const short periodo1 = 100;               // Periodo de media de irradiancia (ms)
const float area1 = 0.0092;               // Área de medida del sensor TSL235R (cm2)
const int periodo2 = 10000;               // Periodo de medida del anemómetro (ms)
const int diametro = 181;                 // Diámetro de anemómetro (mm)
const int temp_precision = 12;            // Resolución de los sensores de temperarura (bits)
const int sensibilidad = 40;              // Sensibilidad del disdrometro
const float area2 = 0.001654;             // Área de medida del sensor disdrómetro (m^2)

// Definición de variables
long contador = 0;                        // Contador de datos almacenados
unsigned int year = 0000;                 // Año
int month = 00;                           // Mes
int day = 00;                             // Día
int hora = 00;                            // Hora
int minuto = 00;                          // Minuto
int segundo = 00;                         // Segundos
float batt = 0;                           // Voltage de entrada
float innerVcc;                           // Voltaje interno
float innertemp;                          // Temperarura interna
float ambtemp = 0;                        // Temperatura ambiental SH15 (ºC)
float humedad = 0;                        // Humedad ambiental SH15 (%)
float TRocio = 0;                         // Punto de rocío (ºC)
unsigned long BWCounter = 0;              // Pulsos Blanco/Negro del anemómetro
float velviento = 0;                      // Velocidad del viento (m/s)
long lux = 0;                             // Luminosidad (LUX)
unsigned long irrad = 0;                  // Irradiancia mW/cm2
float tempext = 0;                        // Temperatura (ºC)
float suelohum = 0;                       // Humedad del suelo (%)
float TSAr = 0;                           // Temperatura del suelo arriba (ºC)
float TSAb = 0;                           // Temperatura del suelo abajo (ºC)
unsigned long intensidad = 0;             // Intensidad de lluvia (gotas/horas/m2)


void setup(){
  // Inicializacion RTC
  RTC.start();
  // Configuración de la alarma
  RTC.enable_interrupt();
  RTC.setSeconds(55);
  RTC.setMinutes(59);
  RTC.setAlarmRepeat(EVERY_HOUR);
  RTC.writeAlarm();
  pinMode(alarma, INPUT);                 // Receptor de alarmas
```

```
  digitalWrite(alarma, HIGH);
  // Configuración de pins
  pinMode(power, OUTPUT);                 // Alimentación de sensores
  // Inicializacion del puerto serie
  //Serial.begin(9600);                   // Inicia comunicaciones
  // Mostrar cabecera por puerto serie (*)
  //SplashScreen();
}

void loop(){
  // Alimentación de sensores
  digitalWrite(power, HIGH);
  delay(5000);
  // Inicialización de sensores
  sensors.begin();                        // Inicia el sensor de temperatura DS18B20
  sensors2.begin();
  // Inicio de la medición de parámetros ambientales
  tiempo();                               // Toma la hora
  delay(20);
  readVcc();                              // Lee el voltaje interior
  delay(20);
  midebateria();                          // Mide el voltaje de entrada
  delay(20);
  readTemp();                             // Mide la temperatura interna
  delay(20);
  midegotas();                            // Mide la intensidad de lluvia (disdrómetro)
  delay(20);
  temperatura();                          // Mide la temperatura exterior (DS18B20)
  delay(20);
  sueloH();                               // Mide la humedad del suelo
  delay(20);
  suelotemp();                            // Mide la temperatura del suelo
  delay (20);
  temphumroc();                           // Lee temperatura y humedad (SH15) y calcula el punto de rocío
  delay (20);
  luminosidad();                          // Mide la luminosidad (LDR)
  delay  (20);
  radiacion();                            // Mide la radiación luminosa, irradiancia (TSL235R)
  delay(20);
  velocidadviento();                      // Mide la velocidad del viento (QRD1114)
  // Muestra los datos a través del puerto serie (*)
  //mostrar();
  // Grabación de los resultados
  grabar();                               // Graba los datos en formato ascii en un soporte microSD
  delay(5000);
  contador = contador + 1;                // Actualiza el contador de medidas
```

```
  digitalWrite(power, LOW);
  delay(500);
  // Fin de la medición, Inicio del letargo
  Dormir();
}

// Función tras despertar
void Despertar(){
}

// Configuración del estado de letargo
void Dormir(){
  attachInterrupt(0, Despertar, FALLING);
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  sleep_enable();
  RTC.enable_interrupt();
  sleep_mode();
  // -->FASE DE LETARGO DEL DISPOSITIVO<--
  sleep_disable();
  RTC.disable_interrupt();
  detachInterrupt(0);
}

// Lee el Reloj de Tiempo Real (DS1337)
void tiempo(){
  // Lee el RTC
  RTC.readTime();
  // Memoriza la fecha actual
  day = RTC.getDays();
  month = RTC.getMonths();
  year = RTC.getYears();
  //Memoriza la hora actual
  hora = RTC.getHours();
  minuto = RTC.getMinutes();
  segundo = RTC.getSeconds();
  return;
}

// Lee el voltaje de entrada
float midebateria(){
  batt = (analogRead(bateria))*volt;
  return batt;
}

//Lee el voltaje interno
long readVcc() {
```

```
  long readVcc=0;
  ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  delay(2);
  ADCSRA |= _BV(ADSC);
  while (bit_is_set(ADCSRA,ADSC));
  readVcc = ADCL;
  readVcc |= ADCH<<8;
  innerVcc = 1126400L / readVcc;                    // Voltaje interno en mV
  innerVcc = innerVcc / 1000;
  return innerVcc;
}

// Obtiene la temperatura interna de dispositivo Arduino
long readTemp() {
  long readTemp=0;
  ADMUX = _BV(REFS1) | _BV(REFS0) | _BV(MUX3);
  delay(20);
  ADCSRA |= _BV(ADSC);
  while (bit_is_set(ADCSRA,ADSC));
  readTemp = ADCL;
  readTemp |= ADCH<<8;
  innertemp = (readTemp - 125) * 1075;          // Temperatura en 10^-4 ºC
  innertemp = innertemp / 10000;                // Temperatura en ºC
  return innertemp;
}

// Mide la temperatura y humedad ambiental y calcula el punto de rocío
void temphumroc(){
  SH15.measure(&ambtemp, &humedad, &TRocio);
}

// Lee la irradiancia (TSL235R)
void radiacion(){
  volatile long pulsos = 0;
  for (int i=0; i <= 10; i++){
    FreqCounter::f_comp = 10;
    FreqCounter::start(periodo1);
    while (FreqCounter::f_ready == 0)
    pulsos=FreqCounter::f_freq;
    delay(20);
  }
  irrad = (pulsos*1000)/(periodo1*area1*1000);
  return;
}
```

```
// Lee la temperatura superficial (DS18B20)
float temperatura() {
  sensors.requestTemperatures();
  tempext=sensors.getTempCByIndex(0);
  return tempext;
}

// Mide la humedad del suelo
int sueloH(){
  suelohum=analogRead(suelo);
  suelohum=(suelohum*0.1904761905);          // Calcula la humedad del suelo calibrada (lectura*100/525)
  return suelohum;
}

// Mide la temperatura del suelo (DS18B20)
void suelotemp() {
  sensors2.requestTemperatures();
  TSAr=sensors2.getTempCByIndex(0);
  TSAb=sensors2.getTempCByIndex(1);
  return;
}

// Mide la luminosidad (LDR)
unsigned int luminosidad(){
  float photocellReading0 = analogRead(luz);
  float Vout0=photocellReading0*0.0048828125;       // Calcula el voltaje
  lux=500/(10*((5-Vout0)/Vout0));                    // Calcula la intensidad de la luz en Lux
  return lux;
}

// Mide la intensidad de lluvia (disdrómetro)
unsigned long midegotas() {
  volatile unsigned long gotas=0;
  unsigned long millis();
  long startTime = millis();                         // Mide el numero de gotas en 30 segundos
  while(millis() < startTime + 30000) {
     int sensorReading = analogRead(lluvia);
     if (sensorReading >= sensibilidad) {
        gotas=gotas+1;
     }
  }
  intensidad = ((gotas / area2) / 30);               // Calcula la intensidad de la lluvia en gotas/m2/sec
}

// Mide la velocidad del viento
void velocidadviento(){
```

```
  BWCounter = 0;
  attachInterrupt(1, addcount, CHANGE);
  unsigned long millis();
  long startTime = millis();
  while(millis() < startTime + periodo2) {
  }
  detachInterrupt(1);
  unsigned int RPM=((BWCounter/2)*60)/(periodo2/1000);  // Calcula revoluciones por minuto (RPM)
  velviento = ((pi * diametro * RPM)/60) / 1000;                  // Calcula velocidad del viento en m/s
}

// Contaje de pulsos anemómetro
void addcount(){
  BWCounter++;
}

/*
// Pantalla de inicio
void SplashScreen(){
   Serial.println("ARDUDROP 1.2");
   Serial.println("------------");
   Serial.println(" V.20101209 ");
   Serial.println();
   tiempo();
   Serial.print(day);
   Serial.print("/");
   Serial.print(month);
   Serial.print("/");
   Serial.print(year);
   Serial.print("   -   ");
   Serial.print(hora);
   Serial.print(":");
   Serial.print(minuto);
   Serial.print(":");
   Serial.println(segundo);
   Serial.println();
   Serial.println("# ; Fecha ; Hora ; Vin ; Vcc ; Tint ; Tamb ; Hamb ; TRocio ; Irrad ; TSup ; Hsuelo ; TS-
      Ar ; TS-Ab ; Lluvia ; Lum ; Viento");
   return;
}

// Muestra las mediciones a través del puerto serie
void mostrar(){
  Serial.print(contador);
  Serial.print(" ; ");
  Serial.print(day);
```

```
  Serial.print("/");
  Serial.print(month);
  Serial.print("/");
  Serial.print(year);
  Serial.print(" ; ");
  Serial.print(hora);
  Serial.print(":");
  Serial.print(minuto);
  Serial.print(":");
  Serial.print(segundo);
  Serial.print(" ; ");
  Serial.print(batt);
  Serial.print(" ; ");
  Serial.print(innerVcc);
  Serial.print(" ; ");
  Serial.print(innertemp);
  Serial.print(" ; ");
  Serial.print(ambtemp);
  Serial.print(" ; ");
  Serial.print(humedad);
  Serial.print(" ; ");
  Serial.print(TRocio);
  Serial.print(" ; ");
  Serial.print(irrad);
  Serial.print(" ; ");
  Serial.print(tempext);
  Serial.print(" ; ");
  Serial.print(suelohum, DEC);
  Serial.print(" ; ");
  Serial.print(TSAr);
  Serial.print(" ; ");
  Serial.print(TSAb);
  Serial.print(" ; ");
  Serial.print(intensidad);
  Serial.print(" ; ");
  Serial.print(lux, DEC);
  Serial.print(" ; ");
  Serial.println(velviento);
  return;
}
*/

// Graba los datos en formato ASCII en unamemoria flash SD
void grabar(){
  // Convierte los datos decimales en valores enteros para ser grabados
  volatile short batt1= (int)batt;
```

```
  volatile short batt2= (abs)((int)((batt-batt1)*100));
  volatile short innerVcc1= (int)innerVcc;
  volatile short innerVcc2= (abs)((int)((innerVcc-innerVcc1)*100));
  volatile short innertemp1= (int)innertemp;
  volatile short innertemp2= (abs)((int)((innertemp-innertemp1)*100));
  volatile short ambtemp1= (int)ambtemp;
  volatile short ambtemp2= (abs)((int)((ambtemp-ambtemp1)*100));
  volatile short humedad1= (int)humedad;
  volatile short humedad2= (abs)((int)((humedad-humedad1)*100));
  volatile short TRocio1= (int)TRocio;
  volatile short TRocio2= (abs)((int)((TRocio-TRocio1)*100));
  volatile short tempext1= (int)tempext;
  volatile short tempext2= (abs)((int)((tempext-tempext1)*100));
  volatile short suelohum1=(int)suelohum;
  volatile short TSAr1= (int)TSAr;
  volatile short TSAr2= (abs)((int)((TSAr-TSAr1)*100));
  volatile short TSAb1= (int)TSAb;
  volatile short TSAb2= (abs)((int)((TSAb-TSAb1)*100));
  volatile short velviento1= (int)velviento;
  volatile short velviento2= (abs)((int)((velviento-velviento1)*100));
  // Grabación de los datos
  char message[120];

      sprintf(message,"%ld;%02u/%02u/%u;%02u:%02u:%02u;%hd,%02hd;%hd,%02hd;%hd,%02hd;%hd
      ,%02hd;%hd,%02hd;%hd,%02hd;%ld;%hd,%02hd;%hu;%hd,%02hd;%hd,%02hd;%lu;%lu;%hd,%0
      2hd\n", contador, day, month, year, hora, minuto, segundo, batt1, batt2, innerVcc1, innerVcc2,
      innertemp1, innertemp2, ambtemp1, ambtemp2, humedad1, humedad2, TRocio1, TRocio2, irrad,
      tempext1, tempext2, suelohum1, TSAr1, TSAr2, TSAb1, TSAb2, intensidad, lux, velviento1,
      velviento2);
  unsigned long length = strlen(message);
  FileLogger::append("datos.csv", (byte*)message, length);
  return;
}
```