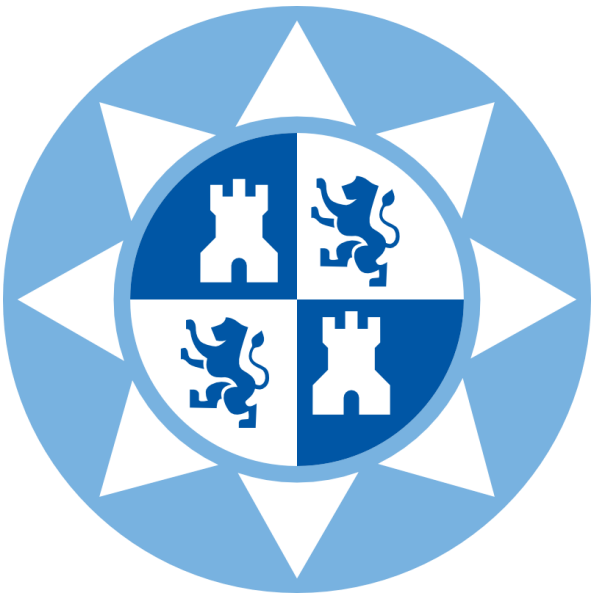
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL
UNIVERSIDAD POLITÉCNICA DE CARTAGENA



Trabajo Fin de Grado

"DESARROLLO DE UN SISTEMA ELECTRÓNICO PARA LA MONITORIZACIÓN DE VARIABLES AMBIENTALES EN HUERTOS URBANOS"

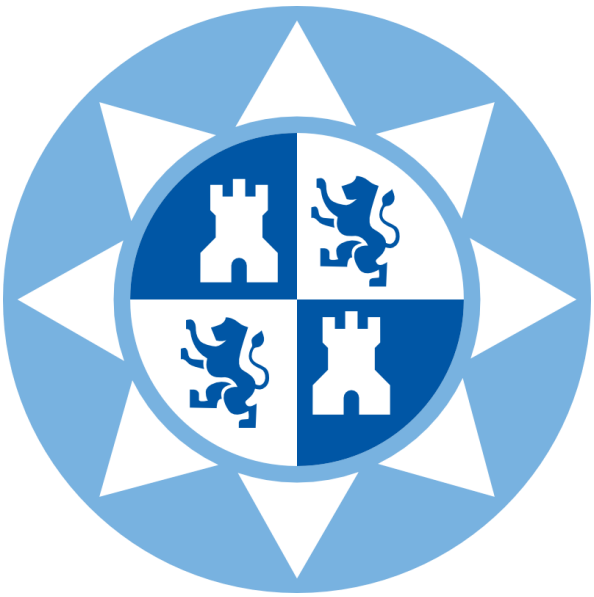
Titulación: Grado en Ingeniería Electrónica Industrial y Automática

Alumno: Alex Sánchez García

Directores: Juan Suardíaz Muro
Jesús Ochoa Rego

Cartagena, 12 de septiembre de 2014

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL
UNIVERSIDAD POLITÉCNICA DE CARTAGENA



Trabajo Fin de Grado

"DESARROLLO DE UN SISTEMA ELECTRÓNICO PARA LA MONITORIZACIÓN DE VARIABLES AMBIENTALES EN HUERTOS URBANOS"

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

Alumno: Alex Sánchez García

Directores: Juan Suardíaz Muro
Jesús Ochoa Rego

A mi abuelo,
siempre te recordaré.

Índice de Contenido

CAPÍTULO 1. INTRODUCCIÓN 1

1.1. Objetivos. 2

1.2. Estructura del Trabajo Fin de Grado. 2

CAPÍTULO 2. ESTADO DEL ARTE 5

2.1. Internet de las cosas. 6
2.1.1. Historia del IoT. 6
2.1.2. Desarrollo del IoT. 8
2.1.3. Limitaciones del IoT. 9
2.1.4. Tendencias del IoT. 9

2.2. Redes inalámbricas. 10
2.2.1. Ventajas de las Redes Inalámbricas. 11
2.2.2. Desventajas de las Redes Inalámbricas. 11
2.2.3. Tipos de redes inalámbricas. 12
2.2.4. Redes inalámbricas de sensores. 15

2.3. Huertos urbanos. 18
2.3.1. Concepto. 18
2.3.2. Beneficios de los huertos urbanos. 19

2.4. Soluciones actuales para la monitorización de huertos. 19
2.4.1. EcoDuino. 19
2.4.2. Parrot Flower Power. 21
2.4.3. Waspmote. 23
2.4.4. MEWiN. 28

CAPÍTULO 3. CASO DE ESTUDIO: INTRODUCCIÓN 29

3.1. HOBO Micro Station H21-002 (Datalogger). 30

3.2. Sensor S-SMB-M005 (Humedad del Suelo). 32

3.3. Sensor S-TMA-M006 (Temperatura del Suelo). 33

3.4. Distribución del sistema. 34

CAPÍTULO 4. SELECCIÓN DEL MICROCONTROLADOR 35

4.1. Plataforma Arduino 36
4.1.1. Arduino UNO 36
4.1.2. Arduino MEGA 38
4.1.3. Arduino NANO 40
4.1.4. Arduino MINI 41
4.1.5. Arduino LEONARDO 42
4.1.6. Arduino YÚN. 44

4.2. Otras alternativas 48
4.2.1. PIC 48
4.2.2. Parallax (Basic Stamp, Propeller…). 51
4.2.3. Raspberry PI 53
4.2.4. TI LaunchPad 54

4.3. Conclusiones. 55

CAPÍTULO 5. ARQUITECTURA HARDWARE 57

5.1. Transceptor nRF24l01. 58
5.1.1. Descripción. 58
5.1.2. Conexionado. 59

5.2. Reloj en tiempo real (SPARKFUN). 60
5.2.1. Descripción. 60
5.2.2. Conexionado. 61

5.3. Sensor de humedad y temperatura SHT10. 62
5.3.1. Descripción. 62
5.3.2. Conexionado. 63

5.4. Sensor de humedad y temperatura DHT22. 64
5.4.1. Descripción. 64
5.4.2. Conexionado. 64

5.5. Conversor ADC 12bits ADS1015. 65
5.5.1. Descripción. 65
5.5.2. Conexionado. 66

5.6. Sensores de irradiancia S1087 y S1087-01. 66
5.6.1. Descripción. 66
5.6.2. Conexionado. 68

5.7. Sensor de temperatura LM35. 69
5.7.1. Descripción. 69
5.7.2. Conexionado. 69

5.8. Relé 5VDC – 250VAC 10A. 70
5.8.1. Descripción. 70
5.8.2. Conexionado. 70

CAPÍTULO 6. ARQUITECTURA SOFTWARE 71

6.1. Plataformas IoTs. 72
6.1.1. Xively. 72
6.1.2. Plotly. 72
6.1.3. Carriots. 74
6.1.4. ThingSpeak. 75

6.2. Distribución del Sistema. 76
6.2.1. Nodo 1 – Nodo central. 77
6.2.2. Nodos 2, 3 y 4. 77

6.3. Obtención de datos 78

6.3.1. Vía ThingSpeak. 79
6.3.2. Vía navegador web. 79
6.3.3. Vía cliente FTP. 80
6.3.4. Vía tarjeta microSD. 82

6.4. Análisis y procesamiento de datos. 82
6.4.1. ThingSpeak. 82
6.4.2. Hoja de cálculo. 83

CAPÍTULO 7. RESULTADOS, CONCLUSIONES Y TRABAJOS FUTUROS 85

7.1. Resultados. 86
7.1.1. Rango de cobertura. 86
7.1.2. Fiabilidad frente a errores. 87
7.1.3. Prueba de funcionamiento. 88

7.2. Conclusiones. 89

7.3. Trabajos futuros. 89

ANEJO I. CONFIGURACIÓN DE ARDUINO YÚN 91

I.1. Configuración inicial. 91

I.2. Cambiar configuración. 93

ANEJO II. CÓDIGOS DEL SISTEMA 94

II.1. Nodo 1 – Nodo central. 94

II.2. Nodo 2 – Nodo sensor SHT10. 102

II.3. Nodo 3 – Nodo sensor DHT22. 105

BIBLIOGRAFÍA 112

Índice de Figuras

Figura 2.1. Evolución del IoT (Cisco IBSG, abril de 2011) 7
Figura 2.2. IoT como red de redes (Cisco IBSG, abril de 2011). 8
Figura 2.3. Pirámide Datos-Sabiduría. 8
Figura 2.4. Red inalámbrica. 10
Figura 2.5. Mezcla de red inalámbrica y cableada. 11
Figura 2.6. Red inalámbrica de área personal 13
Figura 2.7. Red inalámbrica de área local. 13
Figura 2.8. Red inalámbrica de área metropolitana. 14
Figura 2.9. Red inalámbrica de área extensa. 14
Figura 2.10. Red inalámbrica de sensores. 15
Figura 2.11. Nodo de red inalámbrica. 16
Figura 2.12. Aplicaciones de las redes inalámbricas de sensores. 17
Figura 2.13. Componentes EcoDuino 20
Figura 2.14. Conexión Flower Power-Smartphone-Base de Datos 21
Figura 2.15. Componentes Parrot Flower Power. 22
Figura 2.16. Waspmote 23
Figura 2.17. Parte frontal Waspmote 24
Figura 2.18. Parte trasera Waspmote 24
Figura 2.19. Smart Agriculture Waspmote 26
Figura 2.20. Waspmote Agriculture 2.0 Board 26
Figura 2.21. Main-Board (izq); GPRS & Sensor-Board (der) 28
Figura 3.1. Vista general del Huerto 29
Figura 3.2. HOBO Micro Station. 30
Figura 3.3.Sonda S-SMB-M005 32
Figura 3.4. Sonda S-TMA-M006 33
Figura 3.5. Distribución del sistema (inicial) 34
Figura 4.1. Arduino UNO 36
Figura 4.2. Patillaje (pinout) Arduino UNO 37
Figura 4.3. Arduino MEGA. 38
Figura 4.4. Patillaje (pinout) Arduino MEGA 2560 39
Figura 4.5. Arduino NANO 40
Figura 4.6. Patillaje (pinout) Arduino NANO 40
Figura 4.7. Arduino MINI 41
Figura 4.8. Patillaje (pinout) Arduino MINI 41
Figura 4.9. Arduino LEONARDO 42
Figura 4.10. Patillaje (pinout) Arduino LEONARDO 43
Figura 4.11. Arduino YÚN 44
Figura 4.12. Comunicación Arduino-Linux 44
Figura 4.13. Patillaje (pinout) Arduino YÚN 46
Figura 4.14. PIC16F84. 48
Figura 4.15. Tamaños PICAXE. 50
Figura 4.16. PICAXE-28X2 Shield Base. 50
Figura 4.17. BASIC Stamp 2px. 51
Figura 4.18. Propeller ASC+ 52
Figura 4.19. Raspberry Pi Modelo B+ 53
Figura 4.20. LaunchPad MSP430 54
Figura 5.1. Transceptor nRF24l01 58
Figura 5.2. Conector ICSP 59
Figura 5.3. Socket módulo nRF24L01 59
Figura 5.4. Real Time Clock (RTC) 60
Figura 5.5. Pines SCL/SDA 61
Figura 5.6. Conexión RTC-Arduino YUN 61
Figura 5.7. Sensor SHT10 (Humedad/Temperatura) 62
Figura 5.8. Cables SHT10 63
Figura 5.9. Patillaje (pinout) DHT22 64
Figura 5.10. ADS1015 - ADC 12bits 65

Figura 5.11. Sensor S1087 66
Figura 5.12. Espectro de la radiación visible 67
Figura 5.13. Absorción energía clorofilas. 67
Figura 5.14. Sensibilidad (A/W) - Longitud de onda (nm) 68
Figura 5.15. Conexión S1087 - ADC ADS1015 68
Figura 5.16. Sensor LM35 69
Figura 5.17. Patillas LM35 69
Figura 5.18. Relé 70
Figura 6.1. Xively 72
Figura 6.2. Plotly 73
Figura 6.3. Planes de suscripción Plotly 73
Figura 6.4. Funcionamiento Carriots 74
Figura 6.5. Planes de suscripción Carriots 74
Figura 6.6. ThingSpeak 75
Figura 6.7. Gráfica generada con ThingSpeak 75
Figura 6.8. Distribución del sistema (final). 76
Figura 6.9. Descargar datos ThingSpeak 79
Figura 6.10. Acceso a Yún vía web 79
Figura 6.11. Guardar datos vía web 80
Figura 6.12. Comandos servidor FTP 80
Figura 6.13. Configuración Sshfs Manager 81
Figura 6.14. Acceso a Yún con Sshfs Manager 81
Figura 6.15. Adaptadores microSD 82
Figura 6.16. Gráfica "Temperatura del Suelo" ThingSpeak 82
Figura 6.17. Opciones para gráficas en ThingSpeak 83
Figura 6.18. Filtro de datos en Excel 83
Figura 6.19. Importar archivo de texto a Excel 84
Figura 6.20. Gráfica dinámica Temperatura del Suelo 84
Figura 7.1. Módulo nRF24l01 con amplificador. 86
Figura 7.2. Gráficas Temperatura-Humedad del Suelo 88
Figura I.1. Página de login Arduino Yún. 91
Figura I.2. Información Arduino Yún 92
Figura I.3. Configuración Arduino Yún 92
Figura I.4. Botón Reset WiFi Arduino Yún 93

Índice de Tablas

Tabla 2.1. Comparativa Waspmote-Arduino (Memoria y Microcontrolador) 26
Tabla 2.2. Comparativa Waspmote-Arduino (I/O & Buses) 27
Tabla 2.3. Comparativa Waspmote-Arduino (Consumo) 27
Tabla 2.4. Comparativa Waspmote-Arduino (Precio) 27
Tabla 3.1. Comparativa Sondas S-SMx-M005 32
Tabla 3.2. Especificaciones comunes Sondas S-SMx-M005 33
Tabla 4.1. Comparativa placas Arduino. 47
Tabla 4.2. Shields Ethernet-WiFi 47
Tabla 4.3. Comparativa PICs 49
Tabla 5.1. Conexionado nRF24L01 59
Tabla 5.2. Conexionado RTC 61
Tabla 5.3. Conexionado SHT10 63
Tabla 5.4. Conexionado DHT22 64
Tabla 5.5. Conexionado ADS1015. 66
Tabla 5.6. Características principales de los sensores S1087 y S1087-01. 67
Tabla 5.7. Características eléctricas de los sensores S1087 y S1087-01. 68

Capítulo 1.

Introducción

En muchas zonas regables de España y de otras partes del mundo, la disponibilidad de agua para la agricultura es insuficiente para satisfacer las necesidades hídricas de los cultivos; situación que presumiblemente se acentuará en un futuro inmediato dada la dificultad para encontrar nuevas fuentes de agua, debido al aumento de la aridez como consecuencia del cambio climático y a la competencia por el recurso entre los distintos sectores de nuestra sociedad. Bajo estas condiciones, el agricultor deberá ajustar la demanda a la capacidad del abastecimiento adoptando las estrategias de manejo del riego y cultivo más adecuadas a su situación técnica y económica.

Aunque la solución a situaciones de escasez de agua es muy compleja y requiere de la acción conjunta de distintas vertientes de actuación, existe gran unanimidad en la imperiosa necesidad de mejorar la eficiencia de uso del agua. Una de las vías más prometedoras para mejorar la productividad del agua es el uso de técnicas conducentes a una agricultura de precisión.

Las redes de sensores inalámbricas (WSN, *Wireless Sensor Networks*) están comenzando a ser utilizadas en el campo de la agricultura de precisión, para gestionar con mayor eficiencia los recursos hídricos entre otras aplicaciones.

Para implementar WSNs es necesario que los dispositivos que componen dicha red incluyan la electrónica necesaria para la conexión de instrumentación externa de calidad, que habitualmente se utiliza en las aplicaciones de la agricultura de precisión, así como que tengan la robustez necesaria para que puedan ser instalados en campo.

En este proyecto se pretende desarrollar un nodo sensor inalámbrico, basado en WSN, con la robustez necesaria para que pueda sustituir a un dispositivo muy utilizado en casos de estudio agrícolas, como es el datalogger CR1000 de Campbell Scientific. Se espera que el comportamiento de nuestro sistema sea tan válido como el obtenido en otros dataloggers convencionales, con una serie de añadidos extra.

1.1. Objetivos.

Por tanto, el presente trabajo fin de grado tiene como objetivo el desarrollo de un sistema electrónico de bajo coste que permita la monitorización de variables ambientales en huerto urbano. En concreto, el sistema permitirá la monitorización remota de las siguientes variables temporales:

- Temperatura y humedad del suelo.
- Temperatura y humedad del ambiente.
- Radiación total y parcial (luz visible).

El sistema, además, deberá permitir el seguimiento del mes, día y hora en que se hacen las medidas, con posibilidad de almacenamiento en local de los datos en una tarjeta microSD, posibilidad de acceso remoto, vía WiFi, y representación de los datos en una plataforma IoT (Internet of Things).

1.2. Estructura del Trabajo Fin de Grado.

A continuación se detallarán resumidamente los capítulos y anejos que formarán el presente trabajo:

- Capítulo 1 – Introducción

  Se trata del capítulo actual, el cual resume los motivos que han propiciado el desarrollo del proyecto, así como los objetivos buscados.

- Capítulo 2 – Estado del Arte

  Este capítulo describe los conceptos básicos que se manejarán en el proyecto, además se explican diferentes soluciones presentes en el mercado.

- Capítulo 3 – Caso de estudio: Introducción

  En este capítulo se describen los elementos que componen el caso de estudio donde se quiere aplicar la red de sensores desarrollada.

- Capítulo 4 – Selección del microcontrolador

  En el capítulo 4 se analizan las diferentes opciones que se tienen para construir el sistema electrónico y se justifica la elección realizada.

- Capítulo 5 – Arquitectura Hardware

  En el quinto capítulo se realiza la descripción de los elementos que formarán parte del sistema electrónico así como la conexión de éstos al sistema.

- Capítulo 6 – Arquitectura Software

  En este capítulo se realiza un análisis de varias plataformas IoTs. Tras esto, se describe el comportamiento del sistema detalladamente y, por último, se muestran distintas vías posibles para obtener o analizar los datos.

- Capítulo 7 – Resultados, conclusiones y trabajos futuros

  En el último capítulo se analiza en primer lugar problemas que podrían afectar al sistema, así como la solución a ellos. También se realiza una prueba del sistema para verificar su correcto funcionamiento. Por último, se muestran las conclusiones y las posibles vías de trabajo futuro.

- Anejo I – Configuración de Arduino Yún

  Este anejo describe los pasos que deben seguirse para realizar la configuración de Arduino Yún, ya que se debe configurar adecuadamente para su correcto funcionamiento.

- Anejo II – Códigos del Sistema

  En este anejo se adjuntan y se explican los códigos utilizados en el sistema final.

Capítulo 2.

Estado del Arte

En la actualidad, existen millones de sensores que conforman numerosos sistemas y dispositivos electrónicos (sensores de humedad en estaciones meteorológicas, giroscopios en móviles y tabletas, etc.).

Cuando estos dispositivos comenzaron a conectarse a Internet se formó lo que se conoce como “Internet de las Cosas”, este concepto se detallará más adelante.

Debido a esta nueva tendencia, durante los últimos años, ha aparecido una nueva generación de sensores, independientes de un sistema electrónico concreto, que incorporan en un mismo dispositivo el transductor (de la o las variables que interesan medir), la alimentación del propio dispositivo y un módulo de comunicación dotado de cierta inteligencia propia, capaz de organizarse a sí mismo y de interconectarse de forma inalámbrica con otros nodos semejantes.

Como resultado de su despliegue surgen las llamadas redes de sensores inalámbricos (WSN, *Wireless Sensor Networks*), consistentes en redes formadas por multitud de sensores individuales que intercambian información entre sí y/o con un nodo central sin necesidad de cables y mediante un protocolo de comunicación preestablecido.

Las WSN comenzaron siendo utilizadas en aplicaciones militares, pero en la actualidad se usan en multitud de áreas. Algunos ejemplos son la domótica, medicina o en fábricas industriales o las ambiciosas ciudades inteligentes. El diseño eficiente e implementación de las redes inalámbricas de sensores se ha convertido en un área de investigación emergente en los últimos años, debido al gran potencial de estas redes para cubrir grandes áreas con un coste relativamente bajo.

Los avances en las redes de sensores inalámbricos (WSN) han beneficiado a la agricultura, donde se busca encontrar el mayor rendimiento y beneficio, tanto es así que la agricultura que usa dichos sensores ha pasado a denominarse agricultura “de precisión”.

Estas WSN forman parte también de los huertos urbanos, se trata de espacios urbanos destinados al cultivo, tanto en tierra como en recipientes, privilegiando reutilización de envases.

Por tanto, en este capítulo, se trataran los siguientes temas:

- Internet de las cosas.
- Redes inalámbricas.
- Huertos urbanos.

Estos conceptos servirán para conocer cuál es la situación tecnológica actual, de la cual se partirá y permitirán abarcar el presente proyecto.

2.1. Internet de las cosas.

El Internet de las cosas (Internet of Things, en inglés, por cuyas siglas a veces se refiere al Internet de las cosas como IoT) es una idea que se basa en que exista una capa de conectividad digital para cosas existentes, donde "cosas" se refiere a todo tipo de objetos cotidianos, e incluso a sus componentes.

Se espera que esta idea traiga consigo beneficios, en corto plazo, en aspectos como: optimización de la cadena de abastecimiento, efectividad de costos, mejoras en las experiencias de los consumidores, y beneficios en aspectos de seguridad y servicios de emergencia.

Por supuesto, para hacer realidad una idea como el Internet de las cosas, se requiere una evolución en la tecnología que soporta a Internet (por ejemplo, el volumen de información siendo intercambiado simultáneamente), además de cambios en el paradigma de lo cotidiano.

Sin embargo, mientras que las empresas, los gobiernos, los organismos normativos y las áreas académicas trabajan conjuntamente para resolver estas dificultades, el IoT prosigue su camino [1].

2.1.1. Historia del IoT.

La primera aparición del IoT como hoy se conoce, procede del Instituto Tecnológico de Massachusetts (MIT), en concreto, al trabajo del Auto-ID Center. Este grupo, fundado en 1999, realizaba investigaciones en el campo de la identificación por radiofrecuencia en red (RFID) y las tecnologías de sensores emergentes, fue ahí donde se empezó a crear el Internet de las Cosas.

Tras esta aparición surgieron diferentes definiciones sobre lo que significa el IoT pero la más extendida es la siguiente:

> "*IoT es sencillamente el punto en el tiempo en el que se conectaron a Internet más "cosas u objetos" que personas*"

Si se observan los datos se puede ver que en 2003, había aproximadamente 6,3 mil millones de personas en el planeta, y había 500 millones de dispositivos conectados a Internet. Si se divide la cantidad de dispositivos conectados por la población mundial, el resultado indica que había menos de un dispositivo (0,08) por persona.

Por tanto, la definición anterior no era válida en 2003, pues seguía habiendo más cantidad de personas que dispositivos. Pero, el crecimiento explosivo de los Smartphones y las Tablet PC elevó a 12,5 mil millones en 2010 la cantidad de dispositivos conectados a Internet, en tanto que la población mundial aumentó a 6,8 mil millones, por lo que el número de dispositivos conectados por persona es superior a 1 (1,84 para ser exactos) por primera vez en la historia.

Analizando detalladamente esta progresión, se observa que la definición de IoT cobra sentido entre 2008 y 2009 como se muestra en el siguiente gráfico:

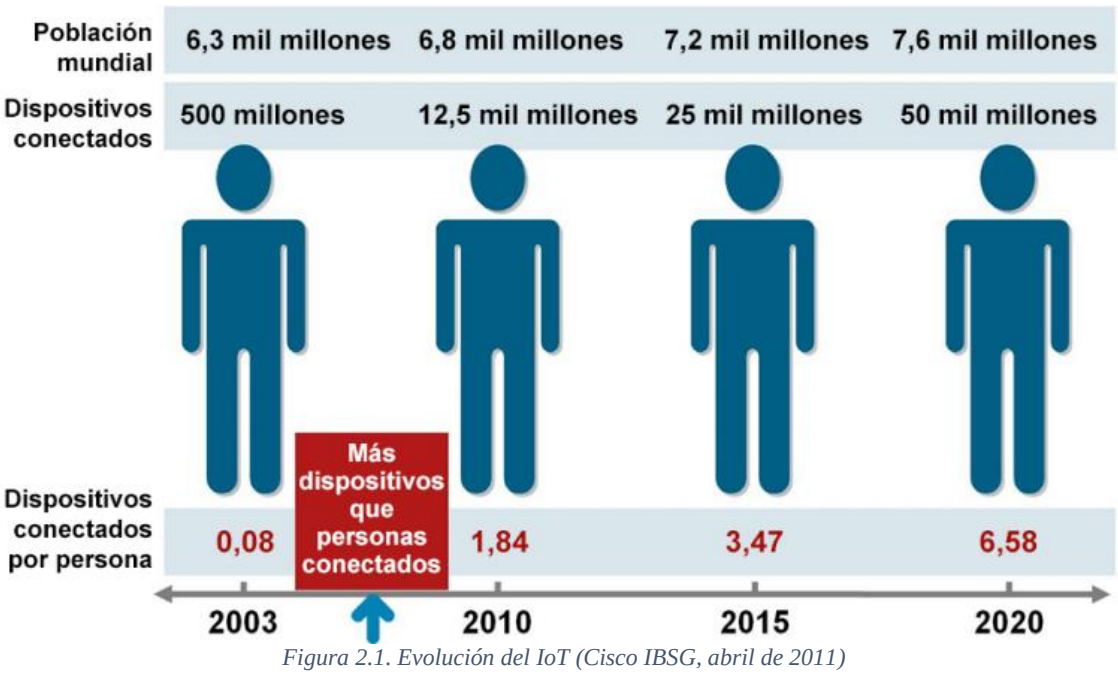


Figura 2.1. Evolución del IoT (Cisco IBSG, abril de 2011)

Por tanto, se puede observar que esta progresión tiene prevista una constante evolución y por tanto cada día el IoT es más grande.

2.1.2. Desarrollo del IoT.

El internet de las cosas está compuesto por diversas redes diferentes y con diversos fines para su uso, por ejemplo, en los automóviles para controlar el funcionamiento del motor, en edificios para el control del sistema de calefacción, seguridad, e incluso iluminación de esta manera las redes estarán conectadas con la incorporación de análisis, capacidades de seguridad y administración, para permitir que el internet de las cosas sea una herramienta aún más poderosa.

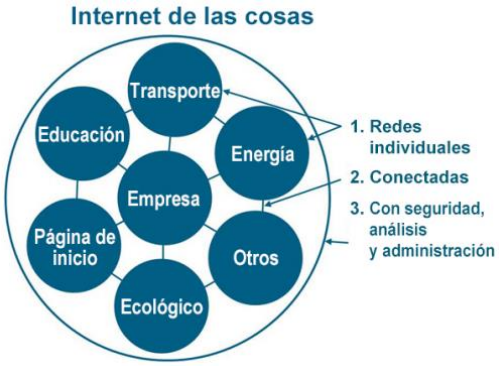


Figura 2.2. IoT como red de redes (Cisco IBSG, abril de 2011).

Por otro lado, hay que mencionar la gran importancia que supone IoT en nuestra sociedad. IoT se considera la primera evolución de Internet pues se trata de un salto que conducirá a aplicaciones revolucionarias con el potencial de mejorar drásticamente la manera en que las personas viven, aprenden, trabajan y se entretienen.

IoT ya ha logrado que Internet sea sensorial (temperatura, presión, vibración, luz, humedad, estrés), lo que nos permite ser más proactivos y menos reactivos, es decir, anticiparnos a lo que va a pasar, analizar minuciosamente cualquier proceso, comprenderlo y actuar en consecuencia.

Pero, la clave de esta evolución se encuentra en los seres humanos que evolucionan porque se comunican. Por ejemplo, después de descubrir el fuego y de haberlo compartido, ya no hacía falta redescubrirlo: solo había que comunicarlo.

Este principio (compartir información y aprovechar los descubrimientos) se puede comprender mejor analizando la forma en la que el hombre procesa los datos:

Figura 2.3. Pirámide Datos-Sabiduría.

Como se puede observar, las capas de la pirámide incluyen datos, información, conocimiento y sabiduría. Por un lado, los datos equivalen a la materia prima que se procesa para obtener información. Estos datos por si solos no valen para nada pero unidos permiten identificar tendencias y patrones. Esta y otras fuentes de información se unen para conformar el conocimiento. Luego, la sabiduría nace de la combinación de conocimiento y experiencia. Por tanto, el conocimiento cambia con el tiempo, la sabiduría es atemporal, y todo comienza con la adquisición de datos.

También resulta importante destacar que existe una correlación directa entre la entrada (datos) y la salida (sabiduría). Cuántos más datos se generan, más conocimiento y sabiduría pueden obtener las personas. IoT aumenta drásticamente la cantidad de datos que están disponibles para procesar. Este aumento, combinado con la capacidad de Internet de comunicar estos datos, hará posible que las personas avancen aún más.

2.1.3. Limitaciones del IoT.

Existen numerosas limitaciones para IoT, desde su aparición bastantes limitaciones se han ido solventando de diversas formas, pero en la actualidad persisten problemas que se pueden resumir en:

- Lagunas en torno a privacidad y seguridad.
- Falta de estándares globales.
- Infraestructuras.
- Limitaciones en los sistemas de procesamiento actuales para soportar tecnologías Big Data y Business Analytics.

Como se observa estos problemas son graves y suponen un atraso en la evolución del IoT.

Por un lado, se debe realizar una normativa que solucione los problemas de privacidad y seguridad entre los distintos nodos del IoT. Además, se debe definir una serie de estándares porque IoT necesita ser global. Por otro lado, todas las Infraestructuras de comunicaciones actuales han de adaptarse (a veces hasta crearse nuevas) para soportar dispositivos y protocolos (p. ej. Zigbee). Por último, deben evolucionarse los sistemas de procesamiento en tiempo real actuales para soportar tecnologías Big Data y Business Analytics.

2.1.4. Tendencias del IoT.

Debido a la aplicación multisectorial de las tecnologías del Internet de las Cosas, todos los perfiles profesionales pueden ser beneficiados.

Por otro lado, algunos fabricantes han realizado una “evolución” al concepto de ‹‹Internet de las Cosas (Internet of Things)›› pasando a denominarlo ‹‹Internet del Todo (Internet of EveryThing)›› pero esta evolución tanto en nombre como en concepto supone un salto exponencial:

- El móvil o cualquier otro dispositivo del usuario será “bombardeado” por servicios de la ciudad.

- Necesidad de ampliar las infraestructuras de comunicaciones actuales y mejora en las tecnologías Big Data: servicios en tiempo real.

- Business Analytics: La información deberá estar filtrada, para que el usuario sólo reciba lo que le pueda interesar: Información personalizada.

- Cloud Computing: La información deberá estar disponible de manera global y distribuida.

Este nuevo concepto supone que ya no solo habrá una comunicación entre dispositivos sino que los dispositivos van a hablar directamente con procesos, van a hablar con individuos. Por lo que el volumen de información que trasiega en este Internet de Todas las Cosas cada vez va a ser más elevado y por lo tanto no solo habrá que ampliar las infraestructuras de comunicaciones, sino que ese gran volumen de información cada vez va a necesitar un mayor y mejor tratamiento analítico para obtener el valor esencial que entregue al usuario una información relevante y personalizada.

En todos estos ámbitos hay todo un camino por recorrer y gran cantidad de desafíos que permitirán la optimización de este concepto.

2.2. Redes inalámbricas.

Las redes inalámbricas son aquellas que no utilizan un medio físico, como un par de hilos de cobre o una fibra, para establecer la comunicación. Estas tecnologías y en especial las dedicadas a las comunicaciones metropolitanas, de área local y las redes de sensores están experimentando un crecimiento muy superior al resto de tecnologías. El factor que más está favoreciendo al desarrollo de estas tecnologías es la aparición de estándares y el apoyo a los mismos por parte de los principales fabricantes del sector.

En sus inicios el principal inconveniente de las tecnologías inalámbricas era el mayor coste de los equipos necesarios, pero la tendencia a la baja de los precios y sus mayores funcionalidades están favoreciendo a su consolidación en el mercado. La posibilidad de mantener una conexión sin perder posibilidades de comunicación librándose de las ataduras que suponen las conexiones cableadas es la mayor valía de estas tecnologías.

Figura 2.4. Red inalámbrica.

Sin embargo, las redes inalámbricas no pretenden sustituir a las redes cableadas ya que estas últimas, actualmente, ofrecen velocidades mucho mayores (decenas de Gbps), que difícilmente se pueden alcanzar con las inalámbricas (centenares de Mbps). Por tanto, se puede concluir que la solución ideal es una mezcla de ambas consiguiendo de esta manera formar una única red que disponga de las ventajas y funcionalidades de los dos tipos de redes.

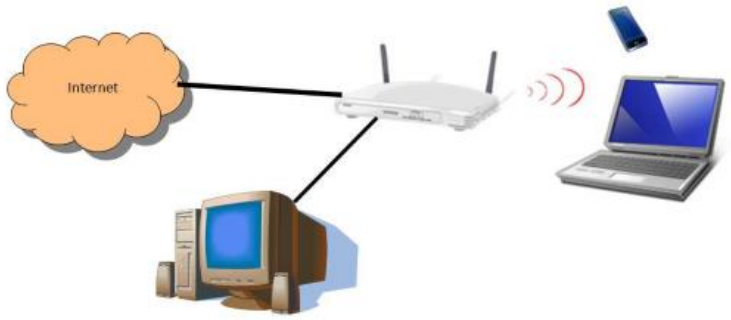


Figura 2.5. Mezcla de red inalámbrica y cableada.

2.2.1. Ventajas de las Redes Inalámbricas.

Teniendo en cuenta las tecnologías disponibles en el mercado actual y comparándolas con las redes cableadas se pueden encontrar las siguientes ventajas de las redes inalámbricas:

- Flexibilidad: al no estar los dispositivos unidos por cables éstos tienen la posibilidad de trabajar como elementos móviles dentro de la red con la única restricción de estar dentro de la zona de cobertura.

- Escalabilidad: las redes inalámbricas pueden ser diseñadas y desplegadas con una gran variedad de topologías permitiendo la incorporación de nuevos dispositivos de forma sencilla y rápida.

- Costo: la tendencia del mercado actual hace que el coste total de una red inalámbrica sea cada vez menor. Esta disminución de costes está basada en la bajada de precio de los dispositivos.

2.2.2. Desventajas de las Redes Inalámbricas.

A continuación se enumerarán las principales desventajas que presentan este tipo de redes:

- Interferencias: este es el mayor problema que se puede tener a la hora de desplegar una red inalámbrica. Las interferencias se pueden dar entre diferentes redes existentes en un mismo espacio, produciendo problemas en las transmisiones de datos o afectando a los equipos de las redes.

- Seguridad: un objetivo muy perseguido en las transmisiones inalámbricas es la seguridad en las mismas. Puesto que el canal de comunicación es accesible por cualquier dispositivo dentro del radio de cobertura se hace necesaria la utilización de algoritmos de encriptación para proteger la información de intrusismos y ataques.

- Influencia en las personas: desde los inicios de las comunicaciones inalámbricas se ha estudiado mucho el efecto de las mismas en la salud de las personas. Los principales estudios que se hace evalúan el efecto que produce la exposición a las ondas electromagnéticas, los resultados suelen indicar que los niveles de intensidad a los que están expuestos los seres humanos no son perjudiciales para la salud.

2.2.3. Tipos de redes inalámbricas.

Los tipos de redes se pueden clasificar según distintas características, a continuación se detallarán las clasificaciones más comunes.

Configuración

Según su configuración se puede clasificar las redes inalámbricas en dos tipos, redes ad-hoc y redes infraestructura.

Las redes ad-hoc son redes en las que la comunicación es directa entre los dispositivos finales de la red y no hay nodos intermedios como pudieran ser los routers de las redes cableadas o los puntos de acceso de las redes inalámbricas. En este tipo de redes todas las máquinas están conectadas entre sí y todas están al mismo nivel y son ellas mimas las encargadas de enviar la información a los demás dispositivos de la red.

Las redes infraestructura son la unión de varias redes mediante un elemento común. Este dispositivo central hace posible la comunicación entre las máquinas de las diferentes redes sin que haya una conexión directa entre ellas.

Alcance

En función del radio de alcance se tienen redes de área personal (WPAN, *Wireless Personal Area Networks*), redes de área local (WLAN, *Wireless Local Area Networks*) redes de área metropolitana (WMAN, *Wireless Metropolitan Area Networks*) o las redes de área extensa (WWAN, *Wireless Wide Area Networks*). A continuación, se detallará cada una de ellas:

- Redes inalámbricas de área personal, WPAN:

  Una red de área personal está pensada para interconectar dispositivos en el radio de acción de una persona, esto es, conectar un PC con todos sus periféricos, teléfono móvil, PDA, etc... en un radio de unos 10 metros.

Figura 2.6. Red inalámbrica de área personal

- Redes inalámbricas de área local, WLAN:

  Las redes WLAN son sistemas de comunicaciones inalámbricas alternativos a las redes cableadas LAN. El hecho de que no haya cables interconectando dispositivos en una red, da una flexibilidad y movilidad a los usuarios de la red, los cuales pueden estar conectados a la misma en cualquier lugar donde haya un punto de acceso.

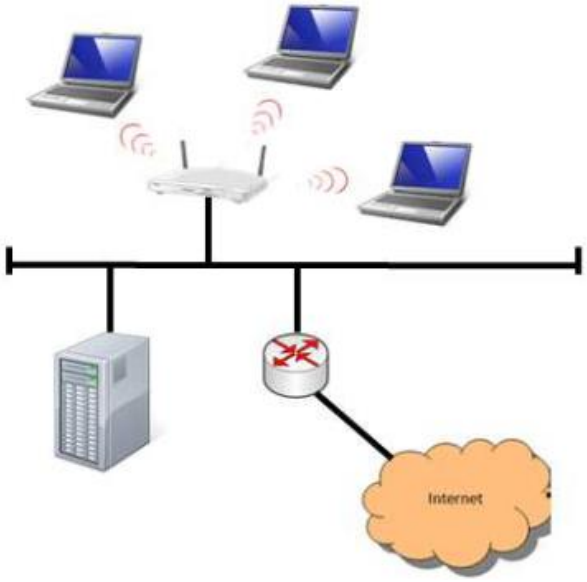


Figura 2.7. Red inalámbrica de área local.

Las velocidades de transmisión tradicionalmente estaban en un rango de 1 a 20 Mbps, pero el creciente desarrollo de aplicaciones multimedia ha provocado la aparición de las redes WLAN de alta velocidad, el último estándar (802.11ac) aumenta la velocidad hasta 1300 Mbps, este valor es teórico ya que las paredes ofrecen una limitación importante.

- Redes inalámbricas de área metropolitana, WMAN:

  Las redes inalámbricas de área metropolitana están pensadas para dar servicio en radios similares al entorno de una ciudad. Fundamentalmente están siendo utilizadas por los operadores para sus enlaces, aunque hay una parte mercado orientado a ofrecer acceso a Internet de alta velocidad en zonas de dificultad orográfica y zonas rurales.

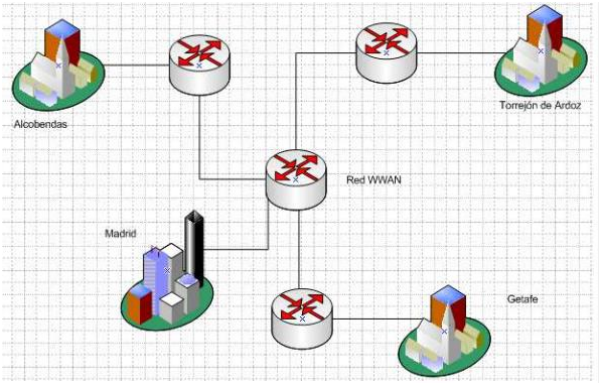


Figura 2.8. Red inalámbrica de área metropolitana.

Las redes inalámbricas de área metropolitana abarcan centenares de kilómetros y son tratadas en la mayoría de ocasiones como redes WLAN extensas o redes WWAN de pequeño tamaño.

El estándar que más se utiliza en la actualidad es WiMAX (Worlwide Interoperability for Microwave Access) con velocidad entorno a 100Mbps, aunque esta velocidad está en continuo aumento.

- Redes Inalámbricas de área extensa, WWAN:

  Las redes WWAN suelen ser siempre redes de datos de dominio público, son las que tiene un alcance más amplio, llegan a dar una cobertura de miles de kilómetros, de ahí que todos los teléfonos móviles estén conectados a una red de este tipo.

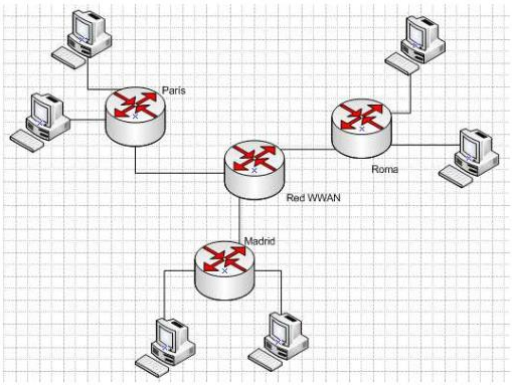


Figura 2.9. Red inalámbrica de área extensa.

Dentro de las WWAN el último estándar que se está empleando en España es la tecnología LTE también conocida como 4G que ofrece una velocidad entorno a los 100Mbps, diez veces mayor que el anterior estándar (3G, 10Mbps).

2.2.4. Redes inalámbricas de sensores.

Las redes de sensores inalámbricas consisten en una serie de sensores autónomos distribuidos espacialmente con el fin de medir propiedades físicas o medioambientales tales como, presión, temperatura, presión sonora o partículas contaminantes. Aunque inicialmente se pensó para aplicaciones militares, hoy es en día en la industria donde se da su principal aplicación.

Figura 2.10. Red inalámbrica de sensores.

Entre los factores que han contribuido a la implantación de las redes de sensores se deben destacar dos:

- Crecimiento exponencial en la industria de los semiconductores. El cumplimiento de la Ley de Moore, hace posible integrar en un mismo chip casi todo el hardware necesario para formar un nodo sensor.

- Aparición de estándares de comunicación inalámbrica de bajo consumo unido al aumento de capacidad de las baterías existentes.

Características

Entre las características de las redes de sensores se pueden destacar:

- Facilidad de despliegue.
- No se utiliza infraestructura de red.
- Topología dinámica, nodos autoconfigurables, tolerancia a fallos.
- Utilización de *broadcast*.

- Ultra bajo consumo, funcionamiento con baterías, larga autonomía.
- Muy bajo coste.
- Pequeño tamaño.
- Operación sin mantenimiento durante largos periodos de tiempo.

Los nodos pueden ser considerados como pequeños ordenadores muy básicos en cuanto al número de interfaces y componentes. Estos nodos, normalmente, están formados por un pequeño microcontrolador con poca memoria y de muy bajo consumo, un dispositivo sensor con la circuitería necesaria para hacer el acondicionamiento de la señal, un dispositivo encargado de establecer la comunicación y una fuente de energía que normalmente será una batería.

Otros tipos de nodos pueden incluir sistemas de alimentación autónomos como pueda ser una célula sola o incluso otra interfaz de comunicación como USB o RS-232 para su comunicación con un PC.

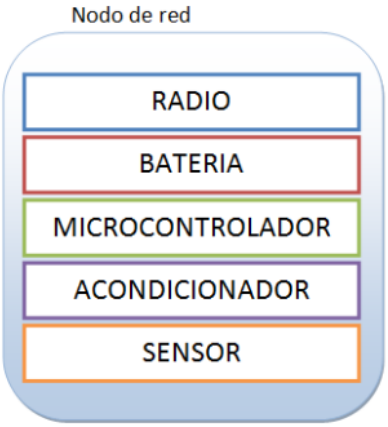


Figura 2.11. Nodo de red inalámbrica.

Problemas de las redes inalámbricas de sensores

Este tipo de redes presentan una serie de problemas por sus características, entre ellos se encuentran:

- Optimización del consumo: es una variable muy a tener en cuenta a la hora de desplegar una red de este tipo ya que la mayoría de los elementos que la forman estarán alimentados con baterías. Para solucionarlos se persigue un software eficiente y la posibilidad de dotar a los nodos de modos de muy bajo consumo.

- Ancho de banda y cobertura de red limitados: esta característica en este tipo de redes puede limitar el tipo de aplicación, no se puede pensar en aplicaciones que requieran de grandes tasas de transmisión de información.

- Recursos de procesamiento limitados: como ya se ha comentado anteriormente, los nodos están dotados de un pequeño microcontrolador con poca memoria por lo que no se debe pedirle operaciones muy complejas. Frente a esto la tendencia es transmitir el mínimo de información y procesarla en estaciones base.

Aplicaciones

Desde un principio se ha pensado que en un periodo de tiempo no demasiado largo las redes de sensores estarán presentes en todo el planeta. Estos nodos distribuidos por toda la superficie podrán dar información que permita entender los distintos fenómenos físicos y medioambientales.

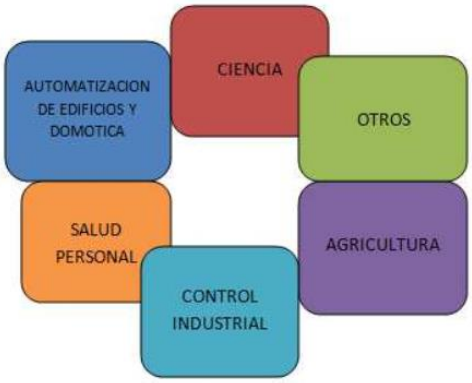


Figura 2.12. Aplicaciones de las redes inalámbricas de sensores.

Una clasificación de los tipos de aplicaciones de las redes de sensores podría ser esta:

- Monitorización de espacios.
- Monitorización de seguridad.
- Monitorización de objetos.
- Redes híbridas.

La monitorización de espacios consiste en lecturas en un entorno inaccesible y hostil en un período de tiempo para detectar cambios, tendencias, etc…Entre sus características destacan:

- Gran número de nodos sincronizados midiendo y transmitiendo periódicamente.
- Tiempo de vida alto de la red.
- Sincronización precisa en la red.
- Topología física relativamente estable.
- Reconfiguración de la red poco frecuente.

Como ejemplos de este tipo de red se encuentran la agricultura de precisión o los estudios ambientales.

La monitorización de seguridad se aplica para la detección de anomalías o ataques en entornos monitorizados continuamente por sensores. A continuación se enumeran algunas de sus características:

- Los nodos no están continuamente enviando datos.
- Menor consumo de energía.
- Se da importancia a rol de un nodo.

- Hay requisitos de tiempo real.

Como ejemplos de este tipo de red existen aplicaciones domóticas, la detección de incendios o las aplicaciones militares.

La monitorización de objetos se aplica para controlar equipos que están dotados de un sensor en una región determinada. Sus principales características son:

- La topología de red es muy dinámica debido al continuo movimiento de nodos sensores.

- La red debe ser capaz de descubrir nuevos sensores y formas topologías.

Como ejemplos de estas redes se incluyen la logística, el control de recursos y la monitorización industrial.

Por último, las redes híbridas se dan en escenarios con aspectos de las tres categorías anteriores, un ejemplo de red híbrida es la monitorización de un hospital.

2.3. Huertos urbanos.

En los países desarrollados, la mayor parte de la población vive en los núcleos urbanos. En éstos el único contacto con el medio natural son los parques y jardines.

Los huertos urbanos suponen un nuevo concepto de espacio verde, donde la participación ciudadana es primordial pues son fruto del trabajo y cuidado de los propios usuarios, por lo que adquieren un valor desde el punto de vista social.

Estos huertos son un instrumento que mejora la calidad ambiental de las ciudades así como la calidad de vida de sus ciudadanos, esto se ha demostrado en los diferentes lugares donde se está llevando a cabo: Estados Unidos, Gran Bretaña, Alemania y en los últimos años España.

2.3.1. Concepto.

Los huertos urbanos son terrenos públicos o privados que son divididos en pequeñas parcelas. La gestión de los huertos y las prácticas agrícolas que se llevan a cabo deben regirse por criterios de sostenibilidad, buscando la creación de un espacio no contaminante, propiciador de biodiversidad y que mejora la calidad ambiental del municipio.

El propietario (normalmente la administración) se encarga de las infraestructuras mínimas para el abastecimiento de agua y la accesibilidad, así como de preparar y acondicionar el espacio (delimitar parcelas, pasillos, accesos, vallado…) y proporcionar un equipamiento.

El huerto debe tener un reglamento de uso que regule su buen funcionamiento y donde se determinen en mayor o menor medida una serie de normas de cultivo atendiendo a criterios de agricultura ecológica.

2.3.2. Beneficios de los huertos urbanos.

Los huertos urbanos, como se ha mencionado anteriormente, son instrumentos para la sostenibilidad con influencia positiva tanto en aspectos sociales como medioambientales:

- Constituyen un nuevo espacio verde en la ciudad.
- Sirven para recuperar espacios vacíos, solares o terrenos abandonados.
- Mejoran la calidad ambiental del municipio.
- Permiten desarrollar cultivos siguiendo criterios de agricultura ecológica.
- Suponen una herramienta muy importante para la educación ambiental.

2.4. Soluciones actuales para la monitorización de huertos.

En el mercado actual, se pueden encontrar una amplia cantidad de soluciones para realizar la monitorización de las distintitas variables de un huerto, unas son más complejas que otras algo que naturalmente repercute en el precio final.

A continuación se analizarán algunas soluciones que se encuentran disponibles para tener una idea general del panorama actual.

2.4.1. EcoDuino.

EcoDuino pertenece a la empresa DFRobot, en este sistema los sensores se usan para almacenar los datos como temperatura, humedad, intensidad luminosa, etc…

Ofrece la posibilidad de programar EcoDuino para que envíe un mensaje para comunicar el estado del huerto de forma remota.

También puede programarse un riego cuando la cantidad de agua sea menor a la indicada o en un intervalo determinado. Esto es posible mediante la conexión del sistema EcoDuino con el PC.

Una de las ventajas que ofrece EcoDuino es que está basado en Arduino, esto significa que puede programarse desde el IDE (Integrated Development Environment) de Arduino y además todos los dispositivos hardware compatibles con Arduino lo serán con EcoDuino.

Diagrama de bloques constitutivos

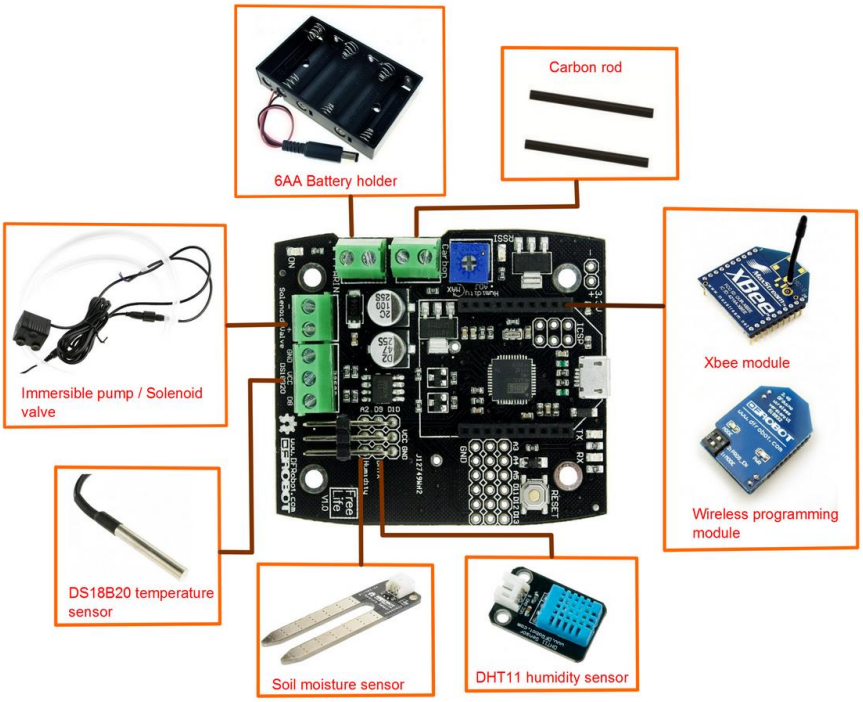


Figura 2.13. Componentes EcoDuino

Especificaciones

- Alimentación: 6-12V DC
- Microcontrolador: Atmega32U4
- 4 puertos I/O analógicos
- 5 puertos I/O digitales
- Slot Xbee
- Micro-USB
- Terminal para conectar un sensor de humedad del suelo
- Terminal para conectar un motor o una válvula solenoide
- Dimensiones: 70x60 mm
- Peso: 15g
- Alimentación de la bomba de agua: 3.5-12V DC
- Altura de bombeo: 200cm
- Caudal: 100-350 L/H
- Rango de potencia: 0.5W-5W
- Dimensiones bomba: 38x38x29mm
- Peso bomba: 125g

Precio

En el mercado se encuentran diferentes alternativas para adquirir EcoDuino, la más básica y económica solamente incluye la placa de EcoDuino por 49€, este pack no incluye el módulo Xbee aunque si incluye los demás componentes. Si se añade el módulo Xbee el precio es de 99€.

Este dispositivo ofrece una amplia funcionalidad y además al estar basado en Arduino se tiene asegurada su ampliabilidad. Por tanto, la relación calidad/precio es buena.

2.4.2. Parrot Flower Power.

La empresa francesa Parrot ha desarrollado una solución comercial para la monitorización de plantas y huertos.

De esta alternativa cabe destacar que está enfocada al usuario final por lo que presenta una serie de desventajas claras ante otras soluciones. El motivo de incluirla en esta comparativa es corroborar que en este sector hay empresas que empiezan a desarrollar productos comerciales para la monitorización de las variables hortícolas.

Para utilizar Flower Power se necesita un dispositivo con Bluetooth 4.0, esta tecnología es conocida por su bajo consumo. Además el fabricante ha desarrollado aplicaciones para Smartphone y Tablet, para poder acceder a los datos de una forma más sencilla.

Otra característica interesante en este dispositivo es la base de datos que ofrece, con más de 7000 plantas, árboles y hortalizas que permitan optimizar el tratamiento de cada una.

Figura 2.14. Conexión Flower Power-Smartphone-Base de Datos

El funcionamiento del dispositivo es el siguiente:

1) Medición de la humedad del suelo: Para medir la humedad del suelo, Flower Power mide la constante dieléctrica. Cuanta más agua hay en el suelo, más elevada es la constante dieléctrica.

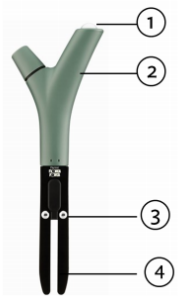
2) Medición de temperatura: Flower Power dispone de 2 termistores, uno para medir la temperatura del aire y el otro para medir la del suelo.

La temperatura se mide especialmente debido a la resistencia de las plantas ya que no todas las plantas soportan igual las temperaturas mínimas.

3) Medición de irradiación solar: La fotosíntesis es fundamental para una planta, Flower Power considera que 510nm es un buen promedio de irradiación solar para las plantas. También mide el promedio en términos de tiempo de exposición.

4) Medición de la conductividad eléctrica del suelo (para fertilizante): Cuando se habla de conductividad eléctrica (CE), lo que se mide es la cantidad de iones presentes en el suelo. Cuantos más iones se miden, mayor es la cantidad de fertilizante que contiene el suelo.

Diagrama



Sensor	Permite...
1	medir la luminosidad
2	medir la temperatura ambiente
3	medir el nivel de abono existente
4	medir el porcentaje de humedad del suelo

Figura 2.15. Componentes Parrot Flower Power.

Especificaciones

- Alimentación: 1 pila Alcalina tipo AAA de 1.5V
- Autonomía: hasta 6 meses
- Bluetooth 4.0
- Índice de protección al agua: IPx5 y IPx7
- Temperatura de funcionamiento: de -10°C a 55°C

Precio

Este dispositivo puede encontrarse en las tiendas especializadas en electrónica por un precio recomendado de 49€.

Como se observa se trata de una alternativa con una baja relación calidad/precio, pues, aunque ofrece un sistema sencillo y robusto, sus componentes son menos precisos que otras alternativas.

Además, como se ha comentado anteriormente, se trata de una solución comercial esto implica que se trata de un dispositivo “cerrado” sin la capacidad de ampliación que ofrecen otras alternativas basadas en Arduino.

2.4.3. Waspmote.

Waspmote es un dispositivo desarrollado por la empresa española Libelium, es similar a Arduino pero posee una serie de características diferentes, se trata de un dispositivo mucho más robusto y completo.

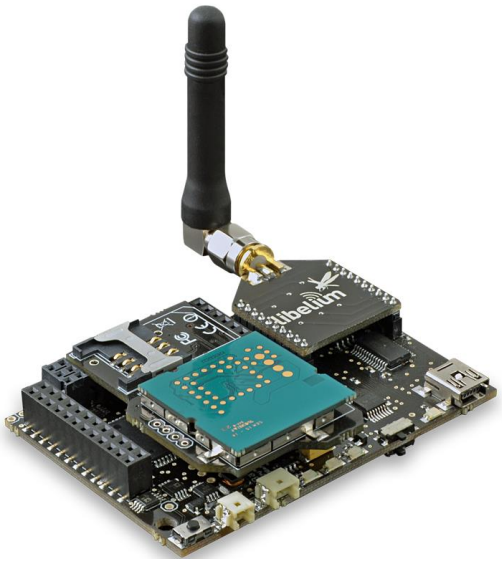


Figura 2.16. Waspmote

Las especificaciones de Waspmote son las siguientes:

Generales

- Microcontrolador: Atmega1281
- Frecuencia: 14MHz
- SRAM: 8KB
- EEPROM: 4KB
- FLASH: 128KB
- SD Card: 2GB
- Peso: 20 gr
- Dimensiones: 73.5x51x13m m
- Rango de Temp.: [-10ºC,+65ºC]
- Reloj: RTC (32Khz)

Input/Outputs

7 Entradas Analógicas

8 I/O Digitales

2 UART

1 I2C

1 SPI

1 Mini-USB

Sockets Específicos para sensores de temperatura, humedad y luz

Diagrama (Waspmote)

Parte frontal:

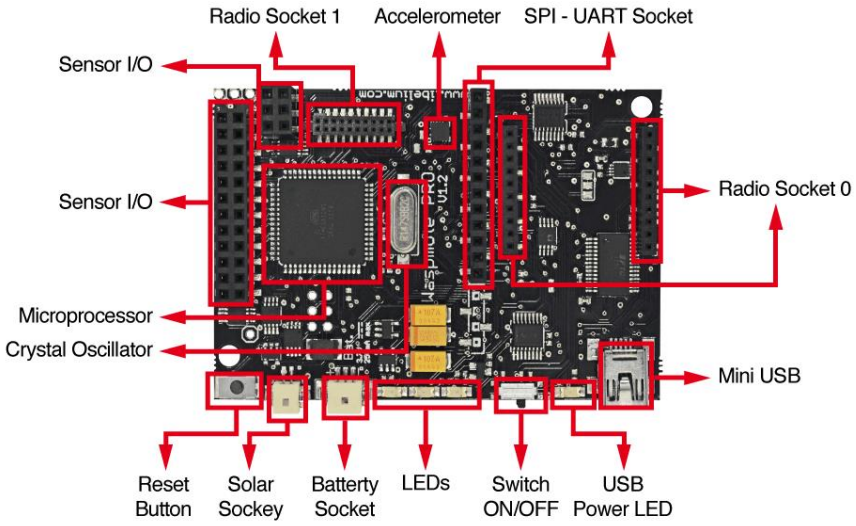


Figura 2.17. Parte frontal Waspmote

Parte trasera:

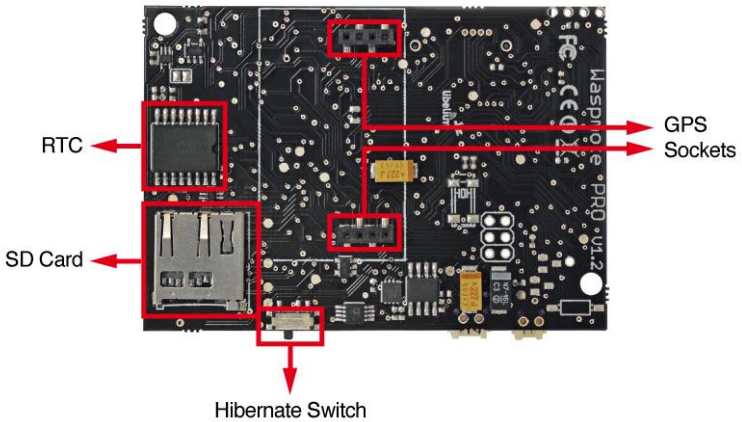


Figura 2.18. Parte trasera Waspmote

Waspmote cuenta con una serie de características muy interesantes:

- Carcasa impermeable (IP65)
- Alimentación mediante energía solar
- Radios disponibles: Zigbee, 802.15.4, Wifi, 868MHz, 900MHz y 3G/GPRS
- Programación “over the air” (OTAP) de varios nodos a la vez
- Programación mediante una interfaz gráfica e intuitiva
- Datos almacenados en la nube

Para WaspMote, Libelium ha desarrollado una serie de “kits” aquí solamente se describirá el kit de agricultura denominado “Waspmote Agriculture 2.0 Board”.

Esta placa permite monitorizar múltiples variables ambientales que forman un amplio rango de aplicaciones, desde el análisis del desarrollo del cultivo hasta la observación meteorológica [2]. Para ello, dispone de una serie de sensores distribuidos en dos kits diferentes, por un lado en el kit básico:

- Sensor de temperatura MCP9700A (*Microchip*)
- Sensor de humedad 808H5V5 (*Sencera*)
- Sensor de temperatura y humedad SHT15 (*Sensiron*)
- Sensor de humedad del suelo WATERMARK (*Irrometer*)
- Sensor de presión atmosférica MPX4115A (*Freescalek*)
- Sensor de humedad de hoja LWS
- Estación meteorológica WS-3000 (Anemómetro, Veleta y Pluviómetro)
- Sensor de luminosidad LDR

Por otro lado, disponen de una versión “PRO” con los siguientes sensores:

- Sensor de radiación solar SQ-110 (*Apogee*)
- Sensor de radiación ultravioleta SU-100 (*Apogee*)
- Dendómetros DC2, DD y DF (*Ecomatik*)
- Sensor de temperatura del suelo PT100

Además, con el objetivo de extender la durabilidad del dispositivo, la placa dispone de una serie de detectores de estado que facilita la regulación de la potencia y prolonga la vida de la batería.

Figura 2.19. Smart Agriculture Waspmote

Diagrama (Placa Agricultura)

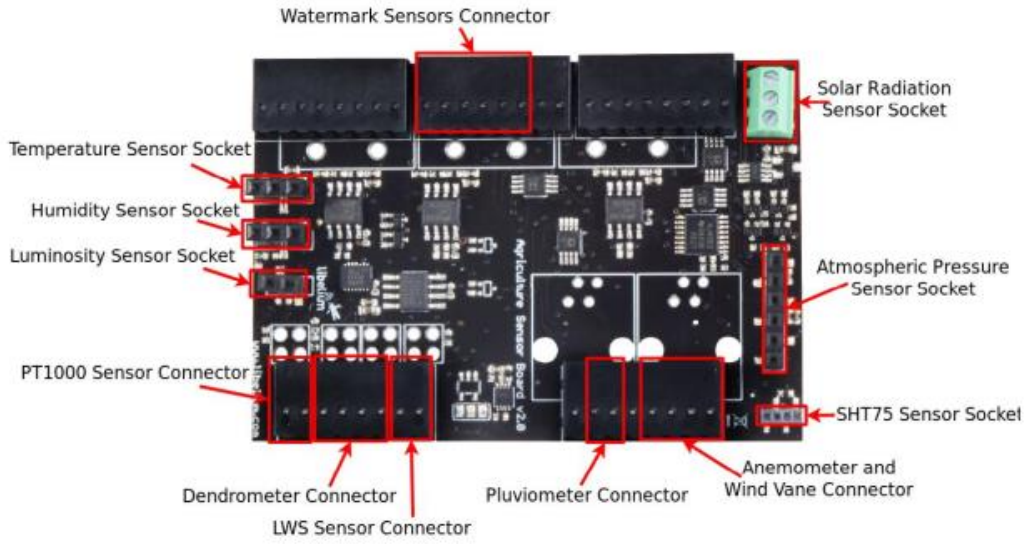


Figura 2.20. Waspmote Agriculture 2.0 Board

Comparativa Waspmote-Arduino [3]

Ahora se realizará una pequeña comparativa con Arduino para observar de forma más clara las diferencias de este dispositivo.

Memoria y microcontrolador:

Modelo	Microcontrolador	Frecuencia	RAM	EEPROM	FLASH	SD Card
Arduino	ATMega328	16MHz	2KB	1KB	32KB	NO
Arduino Mega	ATMega2560	16MHz	8KB	4KB	256KB	NO
Waspmote	ATMega1281	14MHz	8KB	4KB	128KB	2GB

Tabla 2.1. Comparativa Waspmote-Arduino (Memoria y Microcontrolador)

I/O & Buses:

Modelo	Entradas Analog.	Digital I/O	UART's	SPI	I2C	PWM	USB
Arduino	6	8	1	Si	Si	6	Si
Arduino Mega	16	54	4	Si	Si	15	Si
Waspmote	7	8	2	Si	Si	1	Si

Tabla 2.2. Comparativa Waspmote-Arduino (I/O & Buses)

Consumo:

Modelo	Consumo (ON)	Modo "Sleep"	Consumo (Sleep)	Modo Hibernación	Consumo (Hibernación)
Arduino	50mA	Si	2mA	No	-
Arduino Mega	50mA	Si	2mA	No	-
Waspmote	15mA	Si	55uA	Si	0.7uA

Tabla 2.3. Comparativa Waspmote-Arduino (Consumo)

Precio:

	Arduino UNO	Arduino Mega	Waspmote
Placa	22,00 €	41,00 €	155,00 €
Arduino Xbee 802.15.4 + 2dBi antena	45,00 €		
Acelerómetro de 3 ejes	7,75 €		
RTC DS3234 + Batería	16,00 €		
MicroSD Adaptador	20,00		
Zócalo panel solar	47,00 €		
Batería 6600mAh			30,00 €
Total	**157,75 €**	**176,75 €**	**185,00 €**

Tabla 2.4. Comparativa Waspmote-Arduino (Precio)

Precio

Por otro lado, la placa del kit de agricultura tiene un coste de 225,00 €, sin añadir el coste de los sensores. Sin embargo, para poder utilizar la placa de agricultura es necesario adquirir también el dispositivo Waspmote, su precio actual es de 155,00 €. Por tanto, el precio total es de 380,00 €

Esta solución es la más completa vista hasta ahora, ya que ofrece una gran cantidad de sensores, muy variados, con una calidad excelente y además tiene una gran optimización del consumo.

Pero el precio es uno de los grandes inconvenientes, aunque se ha visto como se trata de una alternativa excelente, su precio supone una clara desventaja frente a las demás.

2.4.4. MEWiN.

WIDHOC [4] es una Spin-off de la Universidad Politécnica de Cartagena, esta empresa se dedica al desarrollo de sistemas inteligentes de apoyo al riego. El objetivo es aportar al agricultor más información sobre su plantación para optimizar los recursos.

El equipo de WIDHOC ha desarrollado la placa MEWiN junto con una serie de dispositivos, como la *Main-Board* o la *GPRS & Sensor-Board*, capaces de transmitir los datos remotamente (entre nodos o al servidor).

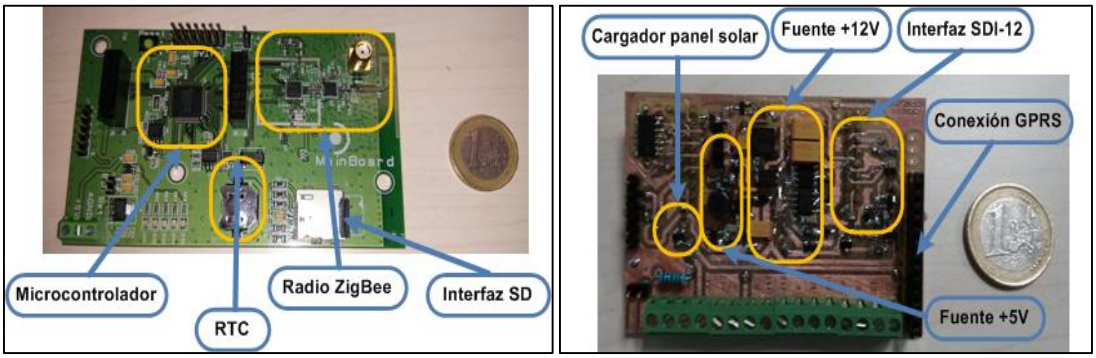


Figura 2.21. Main-Board (izq); GPRS & Sensor-Board (der)

Pero, obviamente, existen muchas diferencias entre esta solución y las anteriores, básicamente WIDHOC ofrece una solución completa, pues se encarga del montaje de toda la parte electrónica, los datos se almacenan en sus servidores, siendo accesibles al usuario desde cualquier dispositivo conectado a Internet, y además proporciona el mantenimiento de la instalación. Todo esto se realiza en régimen de alquiler cuyo coste varía en función de los requerimientos de cada cliente.

Capítulo 3.

Caso de Estudio: Introducción

En este capítulo se introducirá el sistema que se pretende mejorar en el presente proyecto. Para ello se describirá el montaje que se tiene en campo, así como los componentes que lo conforman.

Conviene matizar que este trabajo no es sustitutivo del sistema actual, sino que trata de complementarlo y mejorarlo, enfocado eso sí en el bajo coste, pero sin menospreciar la validez de los resultados.

Por ello es importante describir los distintos elementos que contiene el sistema para tener una idea más clara y objetiva de la dirección de este trabajo.

Más adelante, tras la descripción y selección de los nuevos elementos que se introducirán, se mostrará la composición del sistema final.

Figura 3.1. Vista general del Huerto

A continuación se procede a describir los diferentes elementos qdel actual sistema de monitorización:

3.1. HOBO Micro Station H21-002 (Datalogger).

El H21-002, también conocido como HOBO Micro Station, es un registrador de datos multicanal (1 a 4 canales o más con duplicadores) desarrollado por Onset Computer Corporation.

Figura 3.2. HOBO Micro Station.

Esta unidad cuenta con 500.000 lecturas de memoria y baterías alcalinas de 1 año de duración.

Puede suministrarse con una caja para soportar las condiciones a la intemperie y hasta 4 adaptadores para 0-20 Vdc o hasta 4 adaptadores de 0-20 mA.

Las especificaciones del dispositivo son las siguientes:

- Rango de Operación — Con baterías alcalinas: -20º a 50ºC; Con baterías de litio: -40º a 70ºC
- Entradas (Sensores) — Hasta 4 directamente (más si se usan duplicadores)
- Comunicación — Puerto Serie 3.5mm
- Dimensiones — 90mm(Ancho)x115mm(Alto)x55mm (grosor)
- Peso — 0.36kg
- Memoria — 512Kb almacenamiento flash

- Indicadores de operación — 7 leds (logging y estado de red)
- Intervalo de registro — Desde 1 segundo hasta 18h
- Vida de batería — 1 año (con 4 sensores y registro cada 1 minuto o más)
- Tipo de batería — 4 pilas alcalinas tipo AA (incluidas), o pilas de litio
- Precisión (Tiempo) — De 0 a 2 segundos para el primer punto y ±5 seg./semana a 25ºC
- Modos de inicio de registro — Inmediato, 'push-button', o retardado
- Clasificación ambiental — Resistente a la intemperie, con Gore™ Vent
- Precio — 270€

Este dispositivo es compatible con una gran variedad de sensores, entre los que se pueden destacar los siguientes:

- S-WSx-M0xx Anemómetro (Velocidad/Ráfagas).
- S-SMx-M0xx Sensor de Humedad de Suelo.
- S-UCx-M0xx Sensor de Pulsos (hasta 120Hz).
- S-BPx-CM50 Sensor de Presión Barométrica.
- S-LIx-M0xx Sensor de Luz.
- S-LIx-M0xx Sensor de Radiación Solar.
- S-RGx-M0xx Sensor Inteligente de Lluvia.
- S-TMx-M0xx Sensor de Temperatura.

Nota: Dentro de cada tipo de sensor, Onset ofrece diferentes modelos. Ej: S-TMA**-M0**05** (Modelo A con cable de 5m)*

Se trata de un dispositivo muy completo capaz de ofrecer una gran capacidad sensorial. Pero en el caso que nos concierne se cuenta con la siguiente distribución:

El datalogger MicroStation recibe la información de 6 sensores, como se ha comentado anteriormente este dispositivo solo tiene 4 entradas pero éstas se pueden ampliar con duplicadores por lo que el sistema posee los siguientes componentes:

- (3x) S-SMB-M005 – Sensor de Humedad de Suelo.
- (3x) S-TMA-M006 – Sensor de Temperatura.
- (2x) S-ADAPT-5 – Duplicador de Sensores.

Estos sensores se describirán de forma detallada a continuación.

3.2. Sensor S-SMB-M005 (Humedad del Suelo).

El sensor S-SMB-M005 es un sensor inteligente que mide la humedad del suelo, ha sido diseñado para trabajar adecuadamente en los dataloggers HOBO, de Onset. Combina las innovaciones en el ámbito de la medición de la humedad mediante sondas dieléctricas ECH_2O, de la compañía Decagon Devices, con la tecnología de sensores inteligentes de Onset.

Figura 3.3.Sonda S-SMB-M005

Todos los parámetros del sensor se almacenan en el interior del mismo, que comunica de forma automática la información de su configuración con el datalogger sin necesidad de programarlo, calibrarlo o realizar configuración manual.

Conviene mencionar que existen, dentro del catálogo de Onset, otros modelos con unas características diferentes como se muestra en la siguiente tabla:

Especificaciones	**S-SMA-M005**	**S-SMB-M005**	**S-SMC-M005**
Rango de medida	In soil: 0 to 0.405 m³/m³ (0 to saturated volumetric water content) Extended range: -0.29 to 1.4475 m³/m³ (full scale)	In soil: 0 to 0.450 m³/m³ (0 to saturated volumetric water content) Extended range: -0.376 to 1.964 m³/m³ (full scale)	In soil: 0 to 0.550 m³/m³ (0 to saturated volumetric water content) Extended range: -0.401 to 2.574 m³/m³ (full scale)
Precisión	±0.041 m³/m³ (±4%) typical 0 to +50°C; ±0.020 m³/m³ (±2%) with soil specific calibration	±0.041 m³/m³ (±4%) typical 0 to +50°C; ±0.020 m³/m³ (±2%) with soil specific calibration	±0.031 m³/m³ (±3%) typical 0 to +50°C; ±0.020 m³/m³ (±2%) with soil specific calibration
Resolución	6±0.0004 m³/m³ (±0.04%)	±0.0006 m³/m³ (±0.06%)	±0.0007 m³/m³ (±0.07%)
Dimensiones	254 x 32 x 1.0 mm	152 x 32 x 1.0 mm	89 x 15 x 1.5 mm
Peso	200 gr.	190 gr.	180 gr.
Tipo de Sonda Decagon ECH_2O	EC-20	EC-10	EC-5
Precio	250€	200€	150€

Tabla 3.1. Comparativa Sondas S-SMx-M005

Por otro lado, las especificaciones comunes son las siguientes:

Especificaciones	**Todos los modelos**
Temperatura de funcionamiento	0° to +50°C (+32° to +122°F). While the sensor probe and cable can safely operate at below-freezing temperatures (to -40°C/F) and up to +75°C (+167°F), the soil moisture data collected at these extreme temperatures is outside of the sensor's accurate measurement range.
Bits por muestra	12
Longitud del Cable	5 m

Tabla 3.2. Especificaciones comunes Sondas S-SMx-M005

Como se observa, las diferencias residen en el uso de las distintas sondas Decagon EC-5, EC-10 o EC-20, aportando esto unas características diferentes.

El modelo presente en el caso de estudio ha sido descatalogado por Onset, el propio fabricante recomienda el modelo que ha sustituido a éste, se trata del modelo S-SMD-M005. Este sensor cuenta con una de las últimas sondas desarrolladas por Decagon Devices la sonda 10HS, obviamente con unas especificaciones mejores.

3.3. Sensor S-TMA-M006 (Temperatura del Suelo).

El sensor S-TMA-M006 es un sensor inteligente de temperatura, ha sido diseñado para trabajar adecuadamente en los dataloggers HOBO, de Onset. El sensor tiene un conector modular que permite conectarlo fácilmente al datalogger.

Figura 3.4. Sonda S-TMA-M006

Todos los parámetros del sensor se almacenan en el interior del mismo, que comunica de forma automática la información de su configuración con el datalogger sin necesidad de programarlo, calibrarlo o realizar configuración manual.

Las especificaciones de este sensor son las siguientes:

- Rango de Medida -40° to +75°C
- Precisión ± 0.7°C @ +25°C
- Resolución 0.4°C @ +25°C

- Desvío < 0.1°C por año
- Tiempo de respuesta < 2 min
- Temperatura de func. -40° to +75°C
- Dimensiones 0.6 cm x 3.2 cm
- Peso 14gr
- Bits por muestra 8
- Precio 90€

3.4. Distribución del sistema.

El sistema actual está compuesto por 18 pequeñas cajas con diferentes elementos cada una. Pero, en realidad, realizan el seguimiento de 3 de ellas, solamente de la humedad y de la temperatura. De manera gráfica:

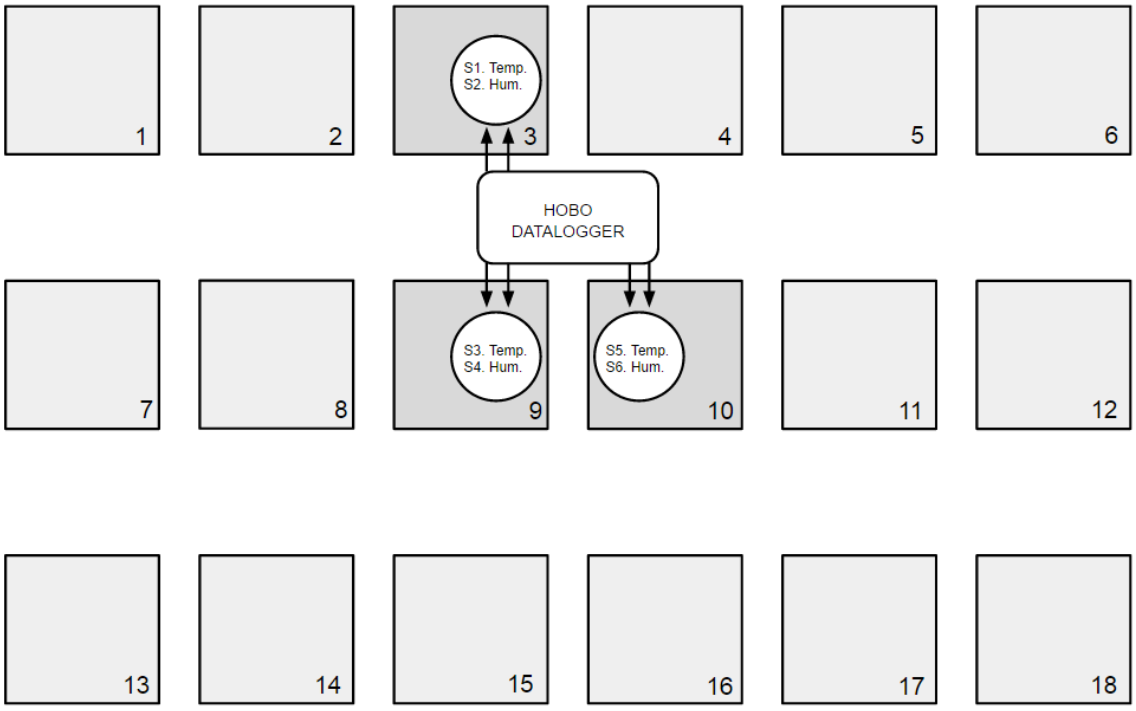


Figura 3.5. Distribución del sistema (inicial)

Lo que se pretende con el presente trabajo es aumentar la información obtenida, buscando una serie de ventajas frente al sistema actual, como por ejemplo, el menor precio de los sensores, la comunicación inalámbrica o la incorporación de otro tipo de sensores, que arrojen mayor cantidad de datos.

Esto permitirá la formación de una red sensorial mucho más completa y útil, que permita comprender el comportamiento del sistema de una manera más detallada.

Capítulo 4.

Selección del Microcontrolador

Hoy en día existe una gran variedad de microcontroladores en el mercado, por ello es conveniente realizar un análisis de las distintas alternativas.

En primer lugar, se mostrarán las alternativas presentes en la plataforma Arduino y se realizará un breve análisis de las diferentes placas de arduino. Ya que existen multitud de opciones dentro esta plataforma y se tienen una serie de características diferentes entre una placa y otra.

Tras esto, se analizarán otras alternativas como, por ejemplo, Raspberry PI, BASIC Stamp, PIC… esto permitirá entender con mayor claridad cuáles son las distintas alternativas disponibles y así tener una visión más global.

Por último, en este capítulo, se realizará una breve conclusión, indicando hacia donde se enfocará el presente proyecto.

4.1. Plataforma Arduino

Arduino es una plataforma electrónica abierta que permite controlar todo tipo de sistemas, ya que cuenta tanto con un software como con un hardware flexible. Arduino fue creado bajo la visión de lo que se conoce como open hardware, permitiendo a las personas poder crear sus propias placas de acuerdo a las necesidades de cada uno.

El funcionamiento de Arduino es muy sencillo, por lo que se lo puede utilizar tanto en el ámbito estudiantil como en grandes proyectos especializados. Permite a través de la computadora, por medio de programación, que el usuario logre interactuar con circuitos electrónicos y controlarlos por software. De igual forma el Arduino es capaz de actuar de manera autónoma sin estar conectado a una computadora.

Existen múltiples modelos de Arduino con diferentes características. Cada modelo posee un nombre, formas, capacidades y funciones distintas.

Por tanto, en este apartado se analizarán las placas de Arduino más relevantes para poder seleccionar las adecuadas al sistema.

4.1.1. Arduino UNO

En primer lugar, se encuentra el Arduino Uno [5] esta placa se ha convertido en la más estandarizada de la plataforma Arduino, su ajustado precio unido a la gran capacidad de ampliación que ofrece mediante shields, la ha convertido en la más utilizada.

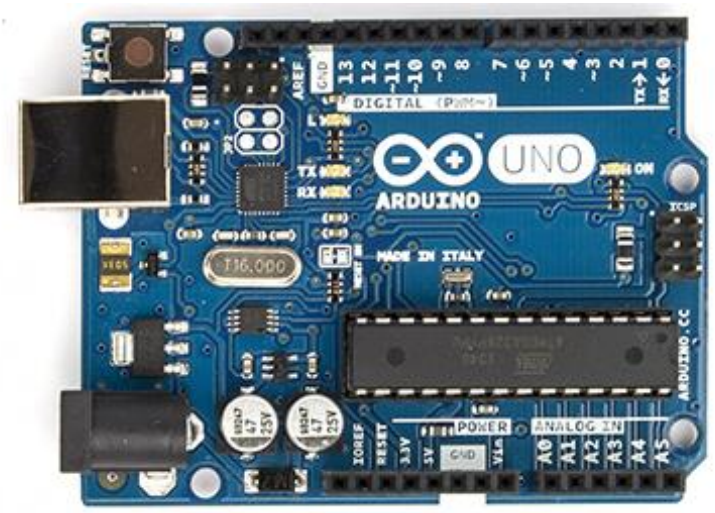


Figura 4.1. Arduino UNO

Las características de la rev.3 de Arduino UNO son las siguientes:

▪ Microcontrolador	ATmega328
▪ Voltaje de funcionamiento	5V
▪ Voltaje de entrada	7-12V
▪ Voltaje de entrada (límites)	6-20V
▪ Pines Digitales I/O	14 (de los cuales 6 proveen salida PWM)
▪ Pines Analógicos de ent.	6
▪ Corriente DC por Pin I/O	40 mA
▪ Corriente DC para Pin 3.3V	50 mA
▪ Memoria Flash	32 KB (de los cuales 0.5KB son usados por el bootloader)
▪ SRAM	2 KB (ATmega328)
▪ EEPROM	1 KB (ATmega328)
▪ Velocidad de reloj	16 MHz

Por otro lado, el *pinout* es:

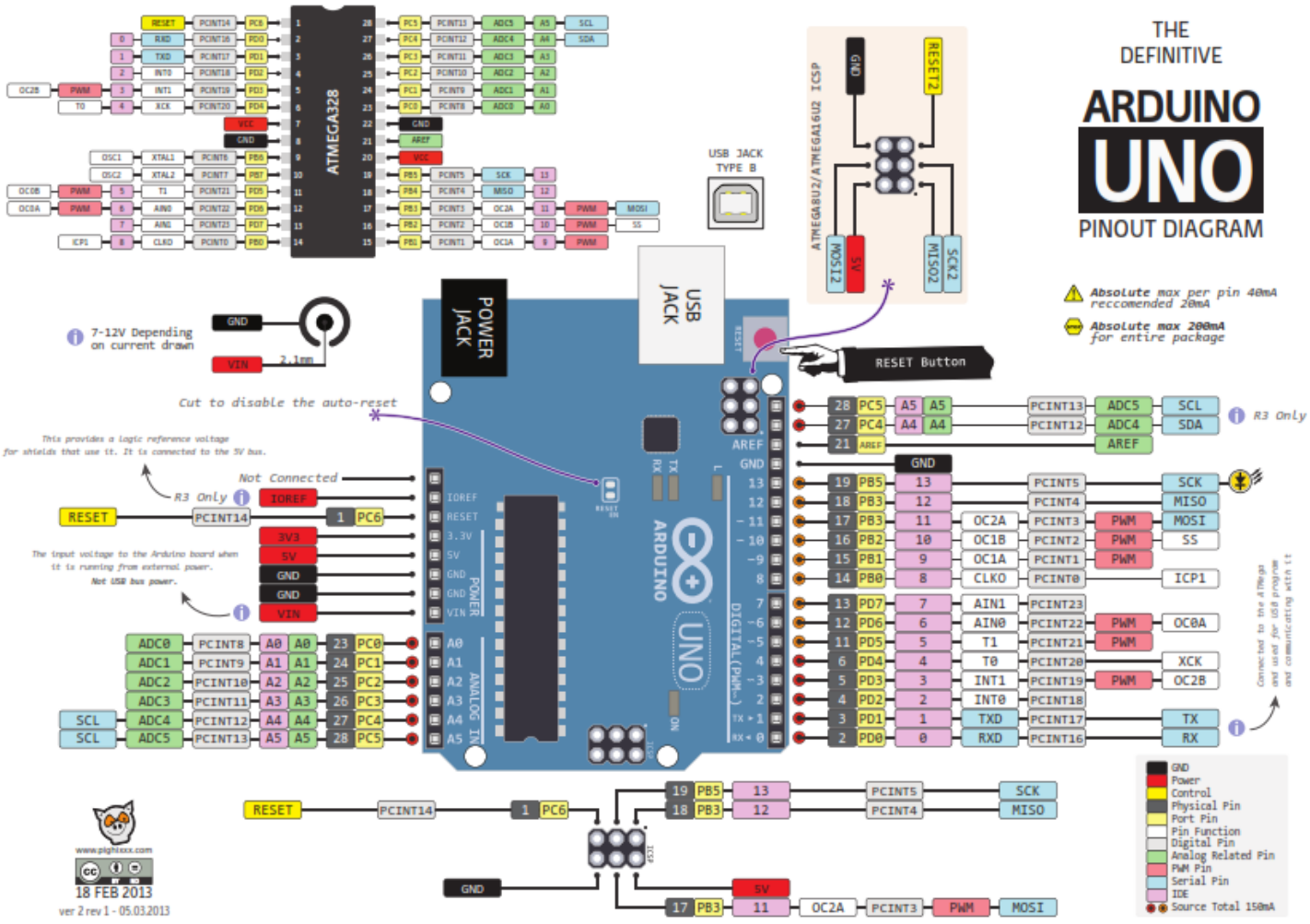


Figura 4.2. Patillaje (*pinout*) *Arduino UNO*

4.1.2. Arduino MEGA

El Arduino Mega [6] tiene una serie de características más avanzadas que Arduino UNO, ofrece más memoria RAM, más pines y más capacidad de procesamiento. A su vez es compatible al 100% con Arduino UNO.

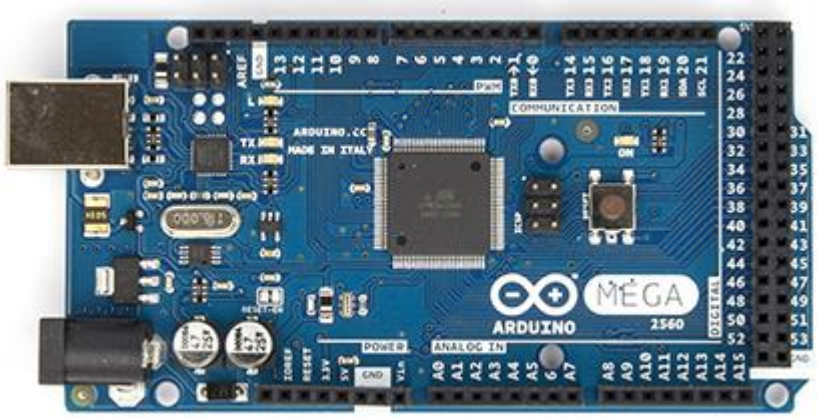


Figura 4.3. Arduino MEGA.

Las características de Arduino MEGA 2560 (Rev.3) son:

- Microcontrolador — ATmega2560
- Voltaje de funcionamiento — 5V
- Voltaje de entrada — 7-12V
- Voltaje de entrada (límites) — 6-20V
- Pines Digitales I/O — 54 (de los cuales 15 proveen salida PWM)
- Pines Analógicos de ent. — 16
- Corriente DC por Pin I/O — 40 mA
- Corriente DC para Pin 3.3V — 50 mA
- Memoria flash — 256 KB (de los cuales 8 KB son usados por el bootloader)
- SRAM — 8 KB
- EEPROM — 4 KB
- Velocidad de reloj — 16 MHz

Se puede ver como las características superan con creces a Arduino UNO, algo lógico puesto que esta placa está pensada para dar respaldo a proyectos de mayor envergadura.

El patillaje (*pinout*) correspondiente a esta placa se muestra detalladamente en la figura 4.4.

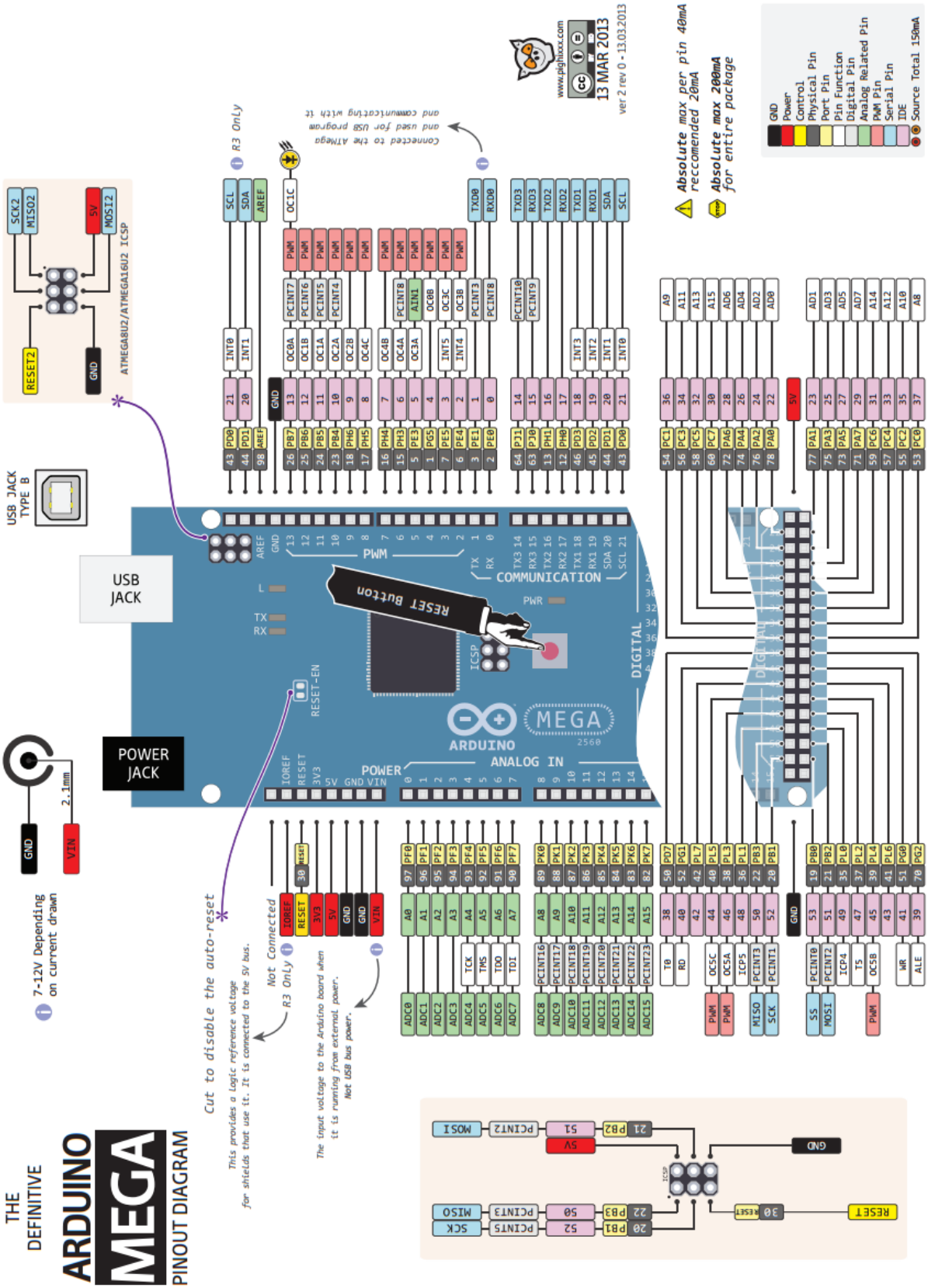


Figura 4.4. Patillaje (*pinout*) *Arduino MEGA 2560*

4.1.3. Arduino NANO

Tras los dos principales miembros de la familia Arduino, el resto de placas complementan a estas dos, tanto en funciones como en precio. En el caso de Arduino Nano [7] nos encontramos con una placa mucho más compacta que se usa conectándola a una protoboard. Se conecta al PC con un cable Mini-B USB.

Figura 4.5. Arduino NANO

Las características del Arduino Nano son:

- Microcontrolador — ATmega328
- Voltaje de funcionamiento — 5V
- Voltaje de entrada — 7-12V
- Voltaje de entrada (límites) — 6-20V
- Pines Digitales I/O — 14 (de los cuales 6 proveen salida PWM)
- Pines Analógicos de ent. — 8
- Corriente DC por Pin I/O — 40 mA
- Memoria Flash — 32 KB (de los cuales 2KB son usados por el bootloader)
- SRAM — 2 KB (ATmega328)
- EEPROM — 1 KB (ATmega328)
- Velocidad de reloj — 16 MHz

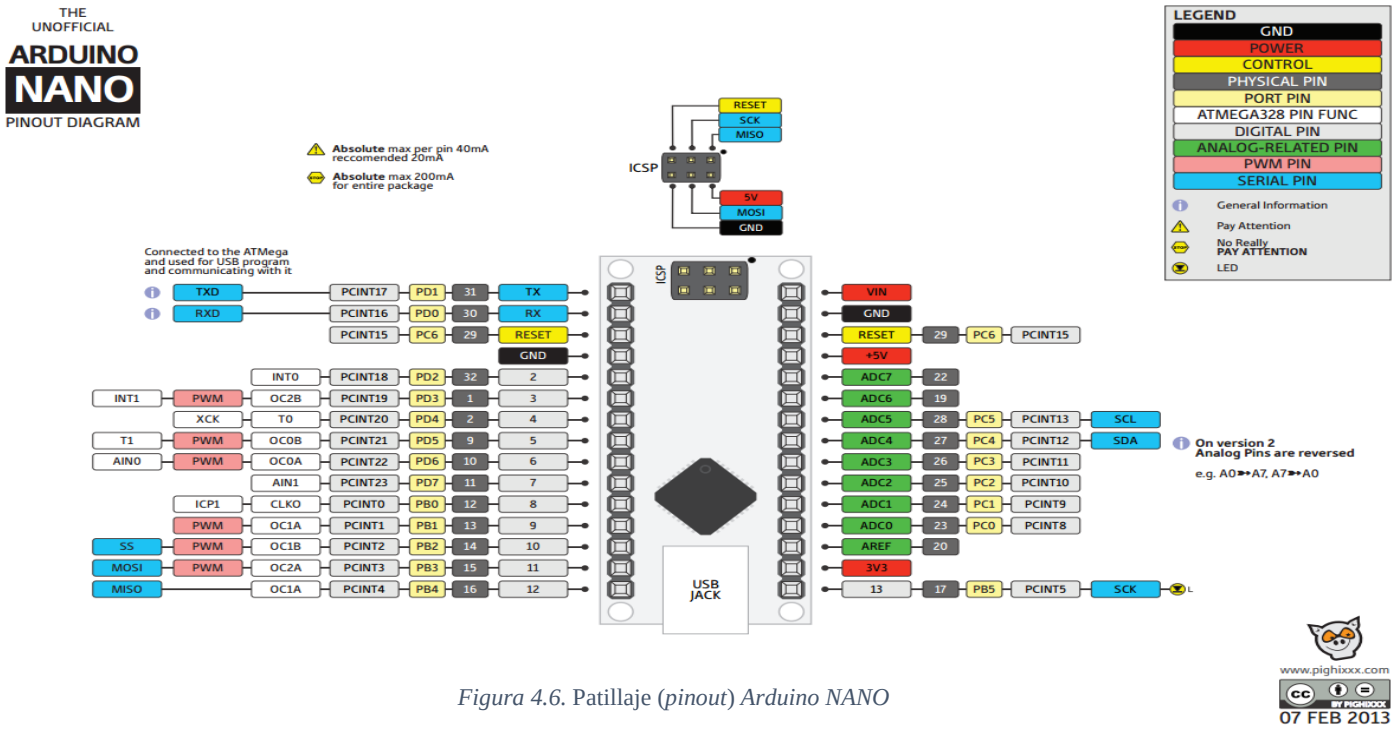


Figura 4.6. Patillaje (*pinout*) *Arduino NANO*

4.1.4. Arduino MINI

La placa Arduino Mini [8] es la más pequeña, algo muy importante en aplicaciones donde el espacio es primordial. Se conecta al PC usando el adaptador mini USB.

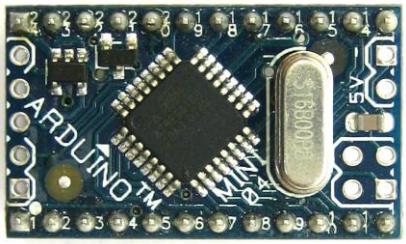


Figura 4.7. Arduino MINI

Las características de esta placa son las siguientes:

▪ Microcontrolador	ATmega328
▪ Voltaje de funcionamiento	5V
▪ Voltaje de entrada	7-9V
▪ Pines Digitales I/O	14 (de los cuales 6 proveen salida PWM)
▪ Pines Analógicos de ent.	8
▪ Corriente DC por Pin I/O	40 mA
▪ Memoria Flash	32 KB (de los cuales 2KB son usados por el bootloader)
▪ SRAM	2 KB (ATmega328)
▪ EEPROM	1 KB (ATmega328)
▪ Velocidad de reloj	16 MHz

Por otro lado, el patillaje (*pinout*) correspondiente a esta placa es el siguiente:

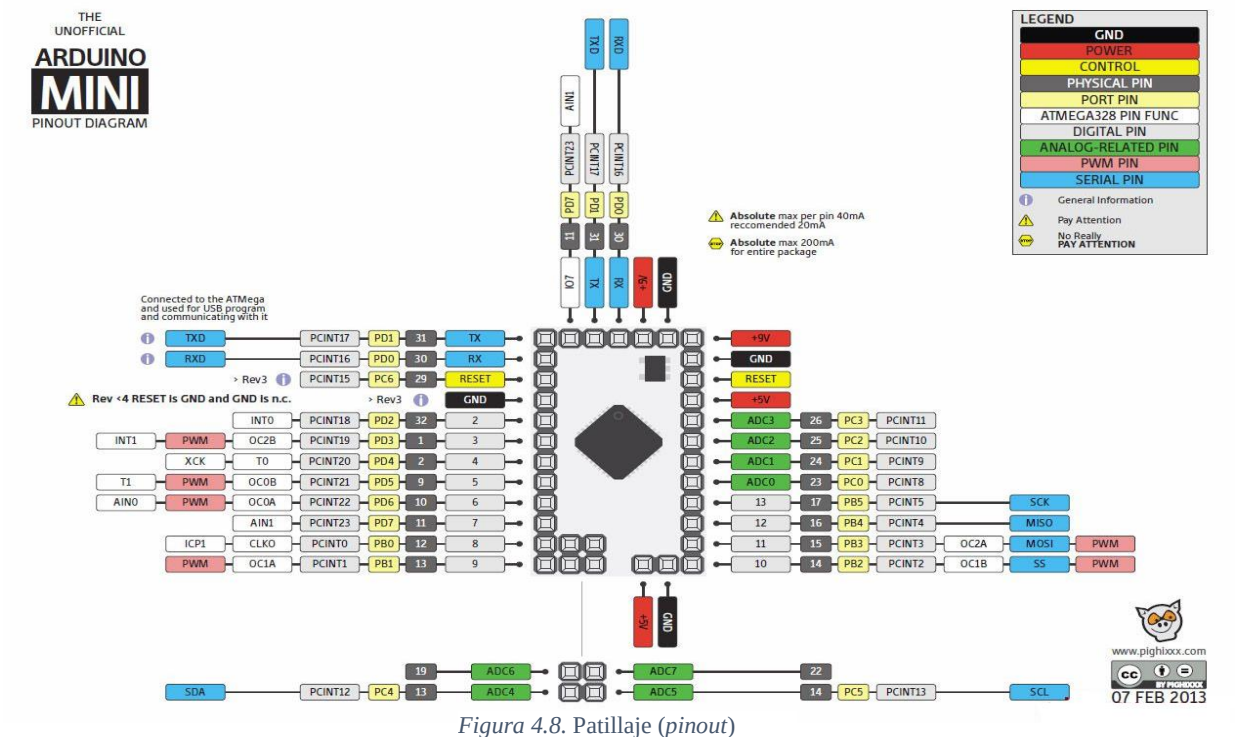


Figura 4.8. Patillaje (*pinout*)

4.1.5. Arduino LEONARDO

Arduino Leonardo [9] se distingue de los anteriores en que es una placa basada en el Atmega32u4, este microcontrolador incorpora comunicación USB, eliminando la necesidad de un procesador secundario, esto permite que Leonardo tome el control de las entradas USB (usando librerías).

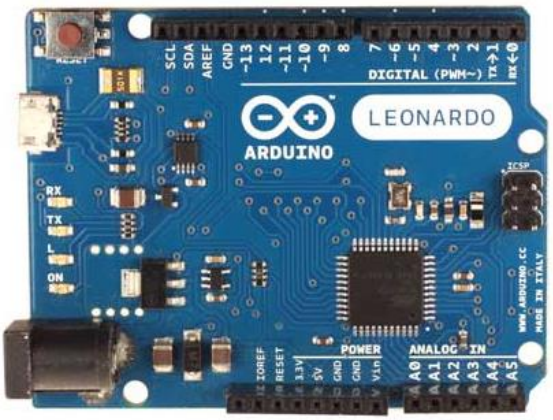


Figura 4.9. Arduino LEONARDO

Las demás características de esta placa son las siguientes:

- Microcontrolador ATmega32u4
- Voltaje de funcionamiento 5V
- Voltaje de entrada 7-12V
- Voltaje de entrada (límites) 6-20V
- Pines Digitales I/O 20 (de los cuales 7 proveen salida PWM)
- Pines Analógicos de ent. 12
- Corriente DC por Pin I/O 40 mA
- Corriente DC en el Pin de 3.3V 50 mA
- Memoria Flash 32 KB (de los cuales 4KB son usados por el bootloader)
- SRAM 2.5 KB (ATmega32u4)
- EEPROM 1 KB (ATmega32u4)
- Velocidad de reloj 16 MHz

Esta placa, por tanto, se encuentra entre Uno y Mega, en lo referente a características, además añade una serie de funcionalidades muy útiles. Como parte negativa mencionar que el microcontrolador que incorpora es más difícil de reemplazar en caso de rotura.

El patillaje (*pinout*) correspondiente a esta placa se muestra a continuación:

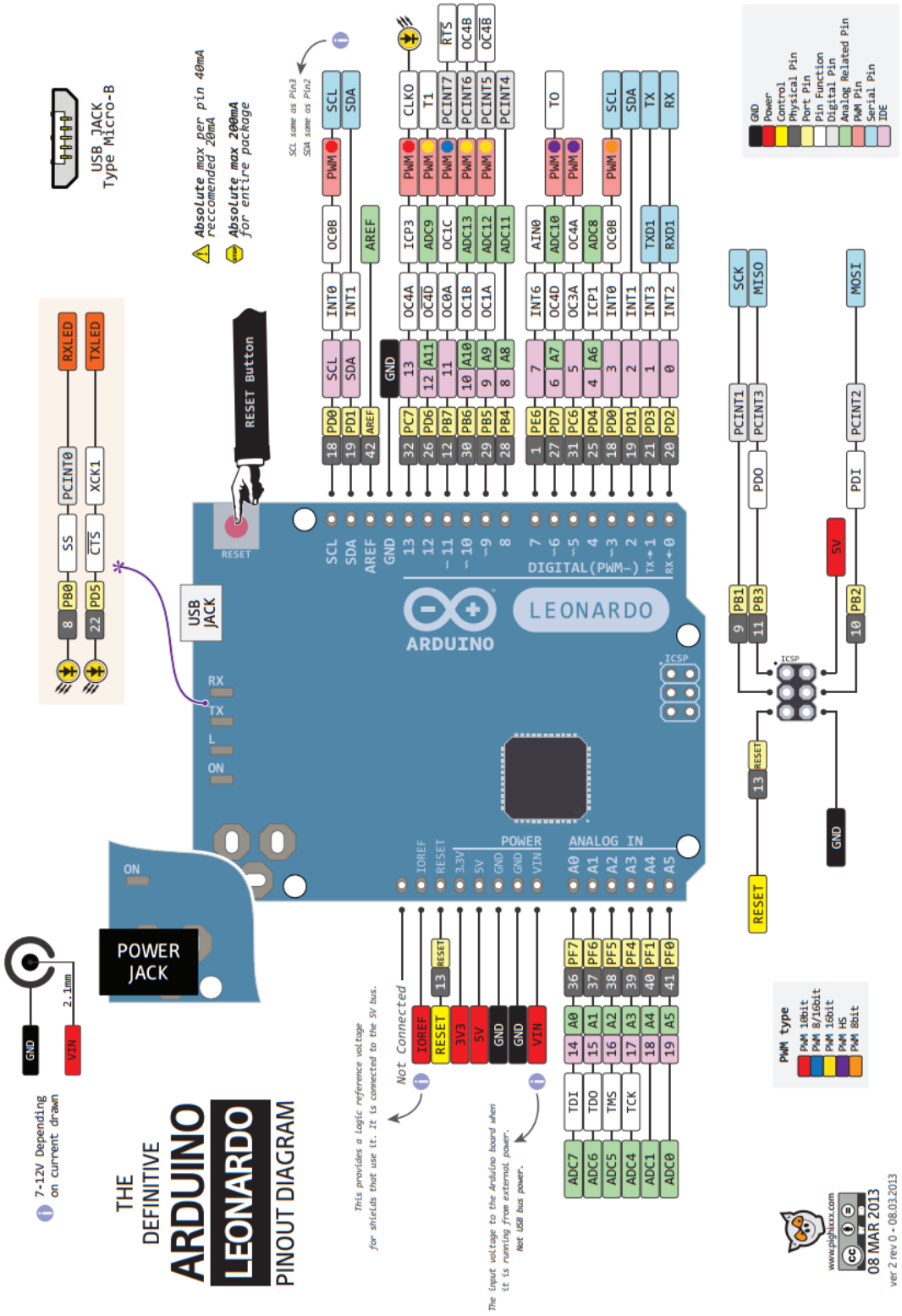


Figura 4.10. Patillaje (*pinout*) *Arduino LEONARDO*

4.1.6. Arduino YÚN.

Arduino Yún [10] es uno de los últimos modelos que han aparecido en el ecosistema de Arduino. Se trata de un modelo enfocado al "Internet de las Cosas" (Yún significa 'nube' en chino), cuenta con dos partes: un microcontrolador Atmega32u4 como el que tiene Arduino Leonardo y un microprocesador SoC (System on Chip) Atheros AR9331 (de arquitectura MIPS) que corre Linino, una versión GNU/Linux basada en OpenWRT (la distribución Linux más popular en sistemas embebidos) para procesadores MIPS.

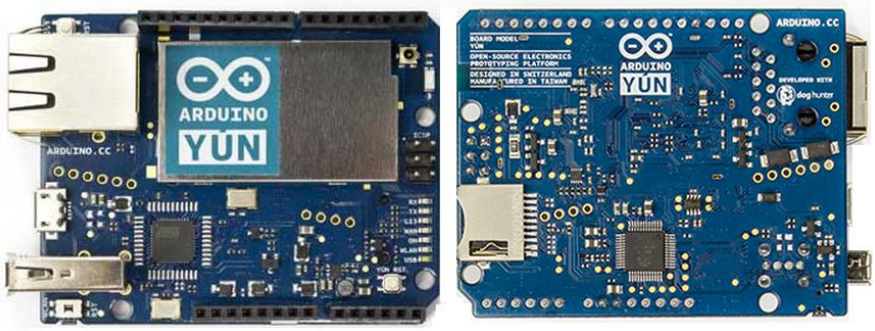


Figura 4.11. Arduino YÚN

Yún posee su propio módulo WiFi el cual viene incorporado en la placa, además de un adaptador Ethernet, un conector USB Tipo A estándar para conectar dispositivos USB a la máquina Linux y una ranura microSD para el almacenamiento adicional de datos.

En el Arduino Yún toda la funcionalidad necesaria para las conexiones de red son delegadas a la máquina Linux, que se encarga de manejar el procesamiento de transacciones y protocolos de red generalmente basado en texto (XML, HTTP, etc.) eliminando así muchas de las limitaciones que tienen las placas Arduino para aplicaciones con Internet.

Se pueden cargar sketches vía WiFi, escribir scripts shell o python, comunicarnos mediante SSH o acceder a los distintos pines de Arduino a través de un API REST. Todo es posible gracias a una librería desarrollada para tal efecto, llamada Bridge. La librería Bridge comunica el AtMega32u4 con el AR9331 para balancear la carga de trabajo y distribuir tareas, dejando a la parte Linux el pesado procesamiento de datos.

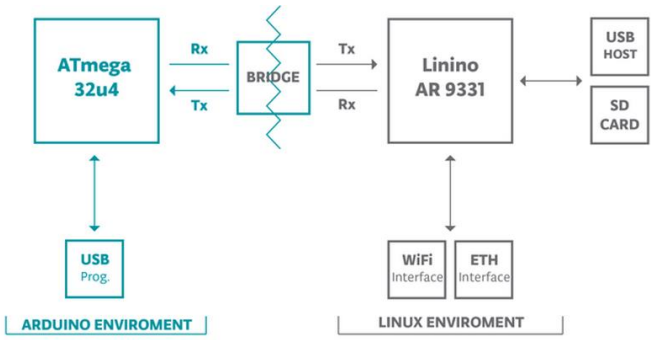


Figura 4.12. Comunicación Arduino-Linux

Las características se dividen en dos partes, por un lado la parte del microcontrolador Arduino:

- Microcontrolador ATmega32u4
- Voltaje de funcionamiento 5V
- Voltaje de entrada 5V (no posee regulador)
- Pines Digitales I/O 20 (de los cuales 7 proveen salida PWM)
- Pines Analógicos de ent. 12
- Corriente DC por Pin I/O 40 mA
- Corriente DC en el Pin de 3.3V 50 mA
- Memoria Flash 32 KB (de los cuales 4KB son usados por el bootloader)
- SRAM 2.5 KB (ATmega32u4)
- EEPROM 1 KB (ATmega32u4)
- Velocidad de reloj 16 MHz

Por otro lado, la parte relacionada con el microprocesador Linux:

- Microprocesador Atheros AR9331
- Arquitectura MIPS @400MHZ
- Voltaje de funcionamiento 3.3V
- Ethernet IEEE 802.3 10/100Mbit/s
- WiFi IEEE 802.11b/g/n
- USB Type-A 2.0 Host
- Card Reader Micro-SD
- RAM 64 MB DDR2
- Memoria Flash 16 MB (Ampliable mediante microSD)

En la siguiente página se muestra detalladamente el patillaje (*pinout*) de esta placa.

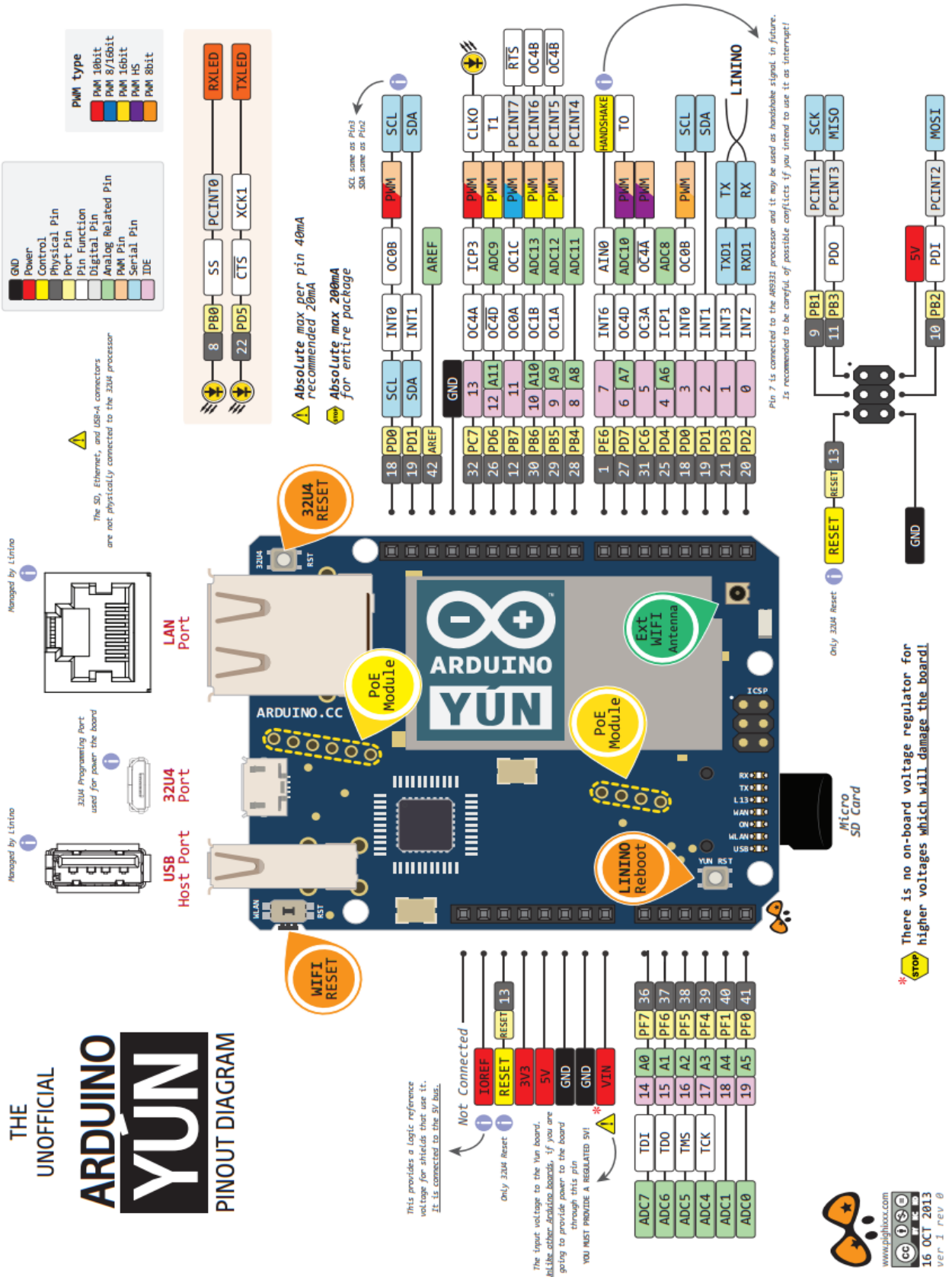


Figura 4.13. Patillaje (*pinout*)

4.1.7. Comparativa de las placas Arduino.

	UNO	MEGA 2560	NANO	MINI	LEO.	YÚN
Microcontrolador	ATmega328	ATmega2560	ATmega328	ATmega328	ATmega32u4	ATmega32u4 + AR9331
Pines Digitales I/O	14 (6 PWM)	54 (15 PWM)	14 (6 PWM)	14 (6 PWM)	20 (7 PWM)	20 (7 PWM)
Pines Analógicos	6	16	8	8	12	12
Memoria flash (KB)	31.5	248	30	30	28	28
SRAM (KB)	2	8	2	2	2.5	2.5
EEPROM (KB)	1	4	1	1	1	1
OTROS	Es el más extendido.	Compatible con Shields de Arduino UNO.	Necesita protoboard.	Se trata del Arduino más pequeño.	Menos memoria; manejo USB.	MicroSD, WiFi, Ethernet, USB, Linux…
Precio (€)	23.25	42.75	39.20	14.45	21.75	64.00

Tabla 4.1. Comparativa placas Arduino.

Como se observa Arduino Yún supone un aumento considerable del coste, pero esto es debido a toda la funcionalidad que ofrece, pues ninguno de los demás Arduino incorpora microSD, Ethernet ni WiFi, ni por supuesto un microprocesador en paralelo.

Para conseguir estos añadidos (microSD, Ethernet o WiFi) hay que recurrir a shields que suponen un gasto adicional, esto se muestra en la siguiente tabla:

	Precio (€)
Slot microSD	6.95
Ethernet + microSD	19.25
WiFi + microSD	72.00

Tabla 4.2. Shields Ethernet-WiFi

Se muestra que para obtener una comunicación WiFi el coste solamente del Shield es mayor que el coste del propio Arduino Yún, por tanto para soluciones que requieran conexión WiFi será más rentable utilizar éste.

4.2. Otras alternativas

Como se ha visto anteriormente, dentro de Arduino se tienen multitud de opciones disponibles las cuales por sí mismas o combinadas podrían ser la solución de multitud de proyectos.

Pero Arduino es una plataforma joven, al igual que el concepto de “open hardware”, por tanto existen multitud de opciones que han optado por migrar a este nuevo concepto, ya sea por medio de los propios responsables de la plataforma o mediante las grandes comunidades online que circulan por Internet. Otras, sin embargo, mantienen su dispositivo más “cerrado”, algo que también tiene sus ventajas, como se verá más adelante.

Por otro lado, existen otras plataformas que han partido desde la base de arduino añadiendo otras características o bien eliminando componentes innecesarios, como ejemplo de este grupo está Waspmote descrito anteriormente.

4.2.1. PIC

Los PIC son microcontroladores desarrollados por la empresa Microchip Technology Inc. Estos se convirtieron en la columna vertebral de los proyectos durante muchos años, debido a una combinación de factores que incluyen bajo costo, fácil disponibilidad, y la proliferación de herramientas de programación libres.

El PIC es un MCU (una unidad completa de microcontrolador) con procesador incorporado, memoria e I/O (in/outputs) programables.

Dentro de esta familia existe una gran diversidad de microcontroladores que podrían utilizarse en multitud de proyectos. Si bien es cierto que existen unos modelos predominantes, como son el PIC16F84 (considerado obsoleto pero muy popular), PIC16F88 o PIC16F87X.

Figura 4.14. PIC16F84.

En la siguiente tabla se muestran las diferencias más relevantes dentro de los PIC:

Familia	ROM [Kbytes]	RAM [bytes]	Pines	Frecuencia de reloj. [MHz]	Entradas A/D	Resolución del convertidor A/D	Comparadores	Temporizadores de 8/16 bits	Comunicación serial	Salidas PWM	Otros
Arquitectura de la gama baja de 8 bits, palabra de instrucción de 12 bits											
PIC10FXXX	0.375 - 0.75	16 - 24	6 - 8	4 - 8	0 - 2	8	0 - 1	1 x 8	-	-	-
PIC12FXXX	0.75 - 1.5	25 - 38	8	4 - 8	0 - 3	8	0 - 1	1 x 8	-	-	EEPROM
PIC16FXXX	0.75 - 3	25 - 134	14 - 44	20	0 - 3	8	0 - 2	1 x 8	-	-	EEPROM
PIC16HVXXX	1.5	25	18 - 20	20	-	-	-	1 x 8	-	-	Vdd = 15V
Arquitectura de la gama media de 8 bits, palabra de instrucción de 14 bits											
PIC12FXXX	1.75 - 3.5	64 - 128	8	20	0 - 4	10	1	1 - 2 x 8 1 x 16	-	0 - 1	EEPROM
PIC12HVXXX	1.75	64	8	20	0 - 4	10	1	1 - 2 x 8 1 x 16	-	0 - 1	-
PIC16FXXX	1.75 - 14	64 - 368	14 - 64	20	0 - 13	8 or 10	0 - 2	1 - 2 x 8 1 x 16	USART I2C SPI	0 - 3	-
PIC16HVXXX	1.75 - 3.5	64 - 128	14 - 20	20	0 - 12	10	2	2 x 8 1 x 16	USART I2C SPI	-	-
Arquitectura de la gama alta de 8 bits, palabra de instrucción de 16 bits											
PIC18FXXX	4 - 128	256 - 3936	18 - 80	32 - 48	4 - 16	10 or 12	0 - 3	0 - 2 x 8 2 - 3 x 16	USB2.0 CAN2.0 USART I2C SPI	0 - 5	-
PIC18FXXJXX	8 - 128	1024 - 3936	28 - 100	40 - 48	10 - 16	10	2	0 - 2 x 8 2 - 3 x 16	USB2.0 USART Ethernet I2C SPI	2 - 5	-
PIC18FXXKXX	8 - 64	768 - 3936	28 - 44	64	10 - 13	10	2	1 x 8 3 x 16	USART I2C SPI	2	-

Tabla 4.3. Comparativa PICs

Como se observa los últimos modelos de la familia PIC incluyen funciones más avanzadas, acercándose cada vez más a las características que posee Arduino, como por ejemplo, la comunicación I2C o SPI, la conectividad Ethernet, etc.

En las tiendas se pueden encontrar alguno de estos microcontroladores por el módico precio de 2€ lo que supone un coste muy ajustado. Pero los PIC pueden ser difíciles de programar para las personas no habituadas a lenguajes de bajo nivel, por ello han surgido soluciones como los chips PICAXE.

El sistema PICAXE [11] es un sistema de microcontrolador fácil de programar que utiliza un lenguaje BASIC muy simple. El poder del sistema PICAXE radica en su sencillez, no necesita de ningún programador, borrador o complejo sistema electrónico.

Un chip PICAXE es básicamente un PIC que ha sido pre-programado con el código de arranque del firmware PICAXE.

Los microcontroladores PICAXE están disponibles en 6 tamaños físicos (8, 14, 18, 20, 28 y 40 pines) y se diferencian en dos series: la serie M2 y la X2.

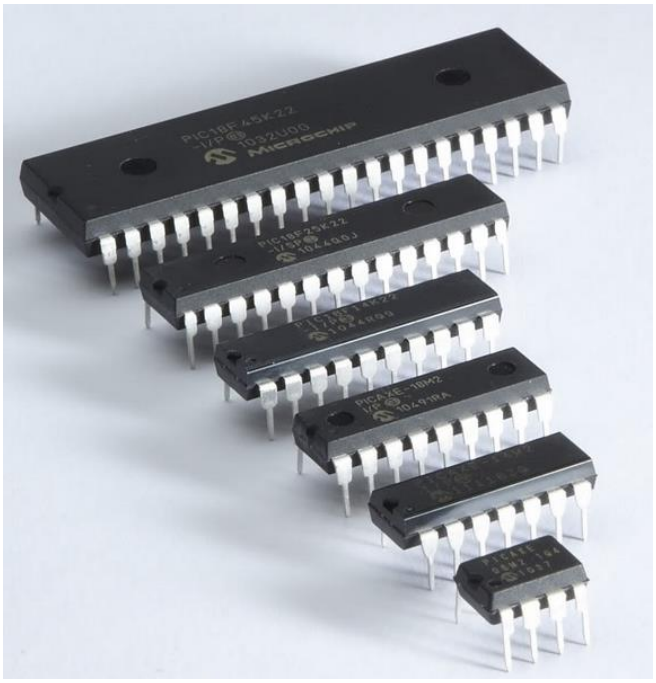


Figura 4.15. Tamaños PICAXE.

Los chips de la serie M2 son los dispositivos estándar que permiten hasta 1,800 líneas de código BASIC y protocolos de interfaz común, tales como RS232 (serie), infrarrojos y I2C. También apoyan el procesamiento de tareas en paralelo.

La serie X2 tiene una mayor capacidad de memoria para programas más largos y más variables (RAM). También tiene un protocolo de interconexión un poco más avanzada, como “1-wire” y “UNIO”.

Por último, mencionar que una de las últimas novedades de PICAXE es el uso de shields como las se utilizan en Arduino, para ello dispone de una “base” para shields como se muestra en la figura 4.16.

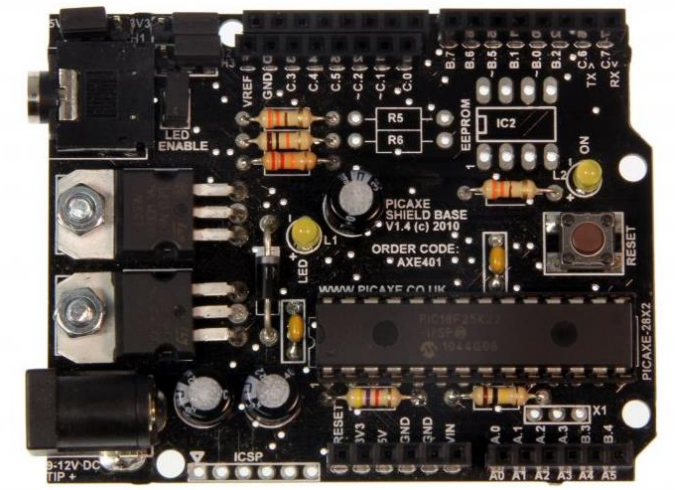


Figura 4.16. PICAXE-28X2 Shield Base.

4.2.2. Parallax (Basic Stamp, Propeller…).

Parallax es una empresa que se dedica al diseño y fabricación de microcontroladores, a continuación se analizarán los productos que ha desarrollado.

BASIC Stamp

El BASIC Stamp (BS) es un pequeño microcontrolador, basado inicialmente en PIC, que ejecuta programas en lenguaje PBASIC muy similar a BASIC, con un editor que se distribuye de manera gratuita. El código del programa se almacena en una memoria EEPROM, que también puede ser utilizada para el almacenamiento de datos.

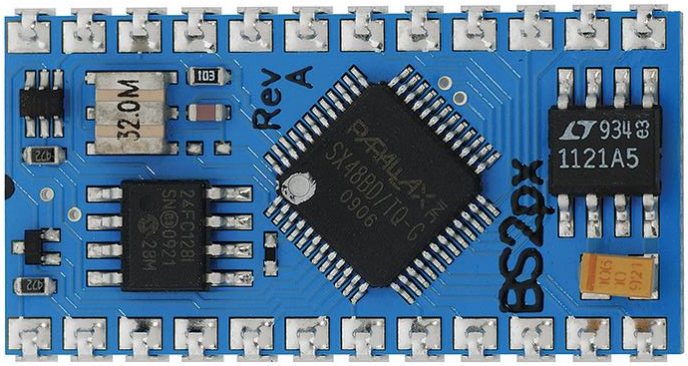


Figura 4.17. BASIC Stamp 2px.

El BASIC Stamp 2px cuenta con las siguientes características:

- Velocidad del procesador: 32MHz Turbo; ~19000 PBASIC instrucciones/seg.
- Comandos PBASIC: 63
- Encapsulado: 24-pin DIP
- Pines I/O: 16 + 2 dedicados a serial.
- RAM: 38 Bytes
- SPM: 128 bytes
- EEPROM: 16 KB; ~4000 instrucciones PBASIC.
- Voltaje: 5.5 a 12 VDC.
- Corriente: 55mA (Encendido), 450uA (Sleep).

Propeller

El modelo más reciente que ha comercializado Parallax es el microcontrolador Parallax Propeller.

Propeller es un nuevo concepto de microcontroladores, en realidad son ocho microcontroladores en un mismo chip. Trabajan al unísono con un mismo reloj, que cubre frecuencias, desde DC hasta más de 80MHz, esto permite hacer programas que se ejecuten en verdadera "multifunción", no como es costumbre hacer "multitarea" por medio de interrupciones. Eso significa que ocho procesos independientes se pueden ejecutar de forma simultánea, monitoreando y respondiendo a sensores y otras entradas.

Propeller cuenta con diversidad de modelos, pero los más representativos son: Propeller Mini y Propeller ASC (Arduino Shield Compatible). Este último cuenta con una disposición muy similar a la que tiene Arduino, pero con un procesador de 8 núcleos mucho más potente que el Atmega de Arduino.

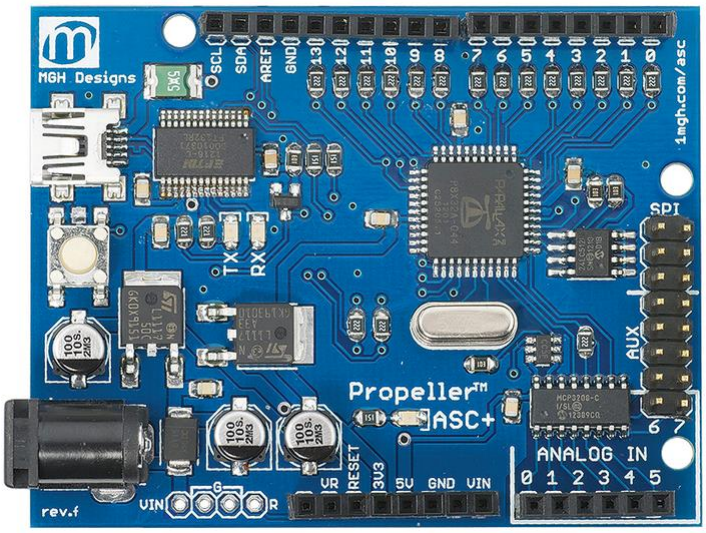


Figura 4.18. Propeller ASC+

Entre las características de esta placa destaca su compatibilidad con shields Arduino, incluye una ranura microSD, posee un conversor ADC de 12 bits (arduino tiene 10bits), 64KB de EEPROM, comunicación I2C, SPI y la programación se puede realizar en varios lenguajes: Propeller SPIN, Propeller Assembly, C y C++.

Por tanto, esta placa es una buena solución si se necesita gran capacidad de procesamiento en paralelo, con las ventajas que supone la compatibilidad con shields arduino y la gran variedad que existe en el mercado.

Su precio se sitúa alrededor de los 40€, un precio muy razonable teniendo en cuenta todo lo que ofrece este dispositivo.

4.2.3. Raspberry PI

Raspberry Pi es una placa computadora (SBC) de bajo costo desarrollada en Reino Unido por la Fundación Raspberry Pi.

Fue diseñada desde el principio como una plataforma de bajo costo para que los niños aprendieran programación (una herramienta educativa barata). Pese a esto miles de proyectos creativos de computación embebidos se están construyendo en torno a la tarjeta. Al igual que con el Arduino, es la floreciente comunidad de la Raspberry Pi lo que la ha hecho exitosa.

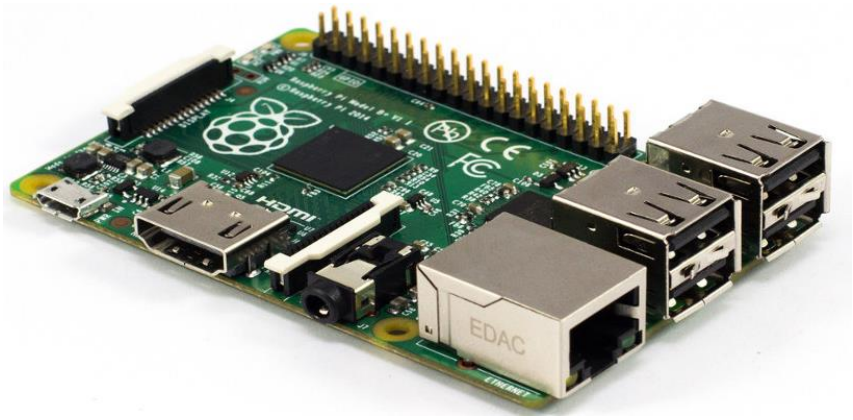


Figura 4.19. Raspberry Pi Modelo B+

Como Arduino, Raspberry Pi es un proyecto open hardware y open source, pero a diferencia de éste se trata de un dispositivo que cuenta con un microprocesador en vez de un microcontrolador. Las características del último modelo lanzado (Raspberry Pi Modelo B+) son las siguientes:

- Procesador: Broadcom BCM2835 SoC fullHD (igual que en modelo B)
- RAM: 512MB SDRAM 400 MHz (igual que el modelo B)
- Almacenamiento: ranura microSD (en el B era SD)
- USB: 4 puertos USB 2.0 (en el modelo B solo había 2)
- Pines GPIO: 40 (en el modelo B, 26)
- Consumo: 600mA hasta 1.8A a 5V (en modelo B, 750mA hasta 1.2ª a 5V)
- Precio: 35$ (26€ aprox.)

Como se puede observar se trata de una buena alternativa a Arduino, con una serie de características que lo hacen idóneo para determinadas situaciones, su precio ajustado unido a su conectividad a Internet vía Ethernet, suponen unos añadidos muy valiosos a la hora de su elección.

4.2.4. TI LaunchPad

Texas Instruments tiene en mercado diferentes tarjetas de desarrollo, las cuales divide en cuatro: MSP430 LaunchPad, C2000 LaunchPad, Connected LaunchPad y Hercules LaunchPad.

Como es lógico cada familia tendrá una serie de características, pero aquí se describirá la más expandida la LaunchPad MSP430.

MSP430

La familia MSP430 [12] de Texas Instruments son procesadores de bajo consumo que ofrecen diferentes conjuntos de periféricos específicos para diversas aplicaciones. El dispositivo cuenta con una CPU RISC de 16bits y los registros son de 16 bits. El oscilador controlado digitalmente (DCO) permite despertar desde los modos de reposo al modo activo en menos de 1ms.

Texas Instruments también ha comercializado una tarjeta de desarrollo denominada LaunchPad:

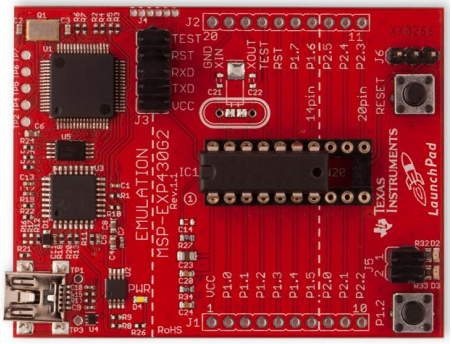


Figura 4.20. LaunchPad MSP430

La principal diferencia en LaunchPad MSP430 y Arduino es el costo. Mientras que un Arduino Uno tiene un precio de 24€, el LaunchPad MSP430 solamente 10€ (cable USB incluido).

Otras diferencias con respecto a Arduino son: el bajo consumo y el conversor ADC de 10/12/16 bits.

Aunque el chip de esta tarjeta de desarrollo (MSP430G2553) solo tiene 14 pines I/O y 16 KB de memoria de programa, frente a los que ofrece el ATmega328 (20 pines I/O y 32KB de memoria de programa) esta placa es una buena alternativa a Arduino.

Por otro lado, se disponen también de shields pero en este caso se denominan BoosterPacks. Además, soporta diferente software para realizar la programación aunque destaca el software multiplataforma denominado Energia que se define como una versión modificada del IDE de Arduino para esta placa.

4.3. Conclusiones.

Como se ha podido observar en este capítulo existe una amplia diversidad de microcontroladores que se pueden utilizar en el proyecto. Se han analizado los dispositivos más relevantes, pero hay muchas más alternativas de las descritas anteriormente.

Sin embargo, en el caso de este proyecto se ha optado por crearlo mediante el uso de los microcontroladores que ofrece Arduino, más adelante se detallará cuáles y cuantos se han seleccionado así como que función tendrá cada uno.

El principal motivo de esta elección son las grandísimas facilidades que ofrece Arduino, con una gran capacidad de expansión mediante shields y una relación calidad/precio muy alta.

El objetivo, a priori, de este proyecto es disponer en cada nodo sensor un dispositivo que recopile los datos, los almacene y los mande a otro dispositivo que sea el encargado de volcarlos a Internet, este último será posiblemente Arduino Yún debido a su ajustado precio, pero los demás dependerán de la cantidad de sensores que se implementen finalmente.

Capítulo 5.

Arquitectura Hardware

En este capítulo se explicarán los dispositivos que componen la parte relativa al hardware del sistema.

La distribución elegida a la hora de realizar la red de sensores, ha sido la de Maestro-Esclavo, por lo que se dispone de una serie de sensores distribuidos, en diferentes nodos que pasarán la información a un nodo Maestro cuando éste se la solicite. Éste tendrá que procesar toda la información que procede de los esclavos y tratarla adecuadamente. Esta parte se detallará más adelante, en el capítulo referente al software.

La comunicación entre nodos se realizará de manera inalámbrica mediante radiofrecuencia, en concreto se ha optado por utilizar módulos de radiofrecuencia nRF24l01, el cual se describirá en este capítulo.

Por otro lado, la información que recogen los sensores de los nodos esclavos necesita ser procesada y enviada mediante un microcontrolador, en concreto en el sistema los nodos secundarios estarán compuestos por Arduinos Uno.

Por último, para el nodo central se ha escogido un Arduino Yún, ya que ofrece una amplia conectividad, en concreto su función será recabar los datos de los demás nodos, guardar esta información en su tarjeta microSD y subirla a Internet.

A continuación se detallarán las características de los componentes seleccionados.

5.1. Transceptor nRF24l01.

5.1.1. Descripción.

Los transceptores nRF24l01 son unos módulos de radiofrecuencia a 2.4GHz que se basan en el chip nRF24l01+ de la compañía Nordic Semiconductor. Este chip integra además del transceptor, un sintetizador de radiofrecuencia y toda la lógica de banda base incluyendo un acelerador de protocolo por hardware, denominado Enhanced ShockBurst (ESB), con una interfaz SPI de alta velocidad para el controlador de la aplicación.

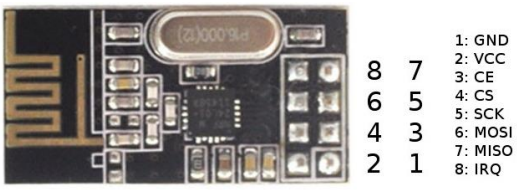


Figura 5.1. Transceptor nRF24l01

El rango de cobertura que ofrece depende de la situación de los transceptores, éstos tienen mucho más alcance cuando están en línea de visión que con obstáculos entre ellos. La distancia normal del módulo de baja potencia es de 50 metros, pero este valor es para espacio abierto, con obstáculos este valor es mucho más bajo.

Dentro de esta familia de transceptores se encuentran diferentes tipos unos con la antena integrada en el chip y otros con antena externa, unos de baja potencia y otros con mayor potencia, obviamente según el modelo escogido se tendrá un alcance u otro. En nuestro caso, dado que la distancia es corta con el modelo más básico basta.

Las características más destacadas del modelo seleccionado son las siguientes:

- Banda ISM de 2.4GHz
- Bajo consumo
- 11.3mA TX a 0dBm de potencia de salida
- 12.3mA RX a una velocidad de 2Mbps
- 900nA apagado
- 22uA en espera
- Regulador de voltaje incorporado en la placa
- Voltaje de alimentación de 1.9 a 3.6V
- Manejo de paquetes automático
- Distancia de alcance 20~50metros
- Dimensiones 33x15 mm

5.1.2. Conexionado.

Este módulo se comunica con Arduino mediante el protocolo de comunicaciones SPI. Como se muestra en la figura 5.1, este módulo cuenta con 8 pines. La conexión de éste al microcontrolador correspondiente se describe en la siguiente tabla:

Señal	**Módulo RF**	**ATmega328**	**ATmega32u4**
GND	1	GND	GND
VCC	2	3.3V ó 5V (socket)	3.3V ó 5V (socket)
CE	3	9 (variable)	9 (variable)
CSN	4	10 (variable)	10 (variable)
SCK	5	13	*SCK (ICSP)*
MOSI	6	11	*MOSI (ICSP)*
MISO	7	12	*MISO (ICSP)*
IRQ	8	NO	NO

Tabla 5.1. Conexionado nRF24L01

Como se observa para el caso del ATmega32u4 (Arduino Yún) se debe realizar la conexión de los pines SCK, MOSI y MISO con los correspondientes del conector ICSP de la placa, a continuación se muestra en detalle ese conector:

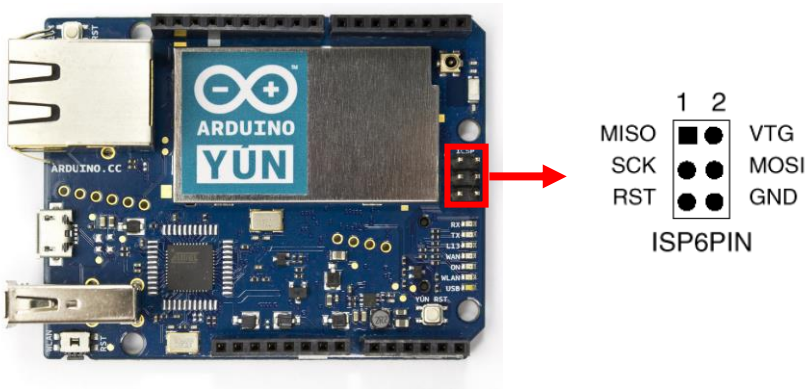


Figura 5.2. Conector ICSP

Por otro lado, es importante señalar que en estas unidades la alimentación (VCC) debe ir conectada a 3.3V. Pero, en el caso de este proyecto se usará un socket como el que aparece en la figura 5.3, que convertirá 5V a 3.3V y además permite conectar el módulo de forma más sencilla:

Figura 5.3. Socket módulo nRF24L01

Por último, los pines CE y CSN se pueden modificar dentro del código por ello son variables y en caso de no disponer de esos pines se pueden elegir otros, modificando el código adecuadamente.

5.2. Reloj en tiempo real (SPARKFUN).

5.2.1. Descripción.

Se trata de una placa de diseño compacto, que contiene un reloj en tiempo real (RTC, en sus siglas en Inglés) DS1307. El módulo viene completamente montado y pro-programado con la hora actual e incluye una pila de botón CR1225 de litio que permitirá ejecutar el RTC durante un mínimo de 9 años (típicamente son 17 años).

Figura 5.4. Real Time Clock (RTC)

El DS1307 es un chip con un reloj de tiempo real de baja potencia con 56 bytes de memoria RAM no volátil, reloj de código BCD y calendario. Se accede mediante protocolo I2C. La última revisión del módulo añade resistencias I2C y una almohadilla de la batería más grande para solucionar problemas de cortocircuito en la placa.

Las características más relevantes, de forma resumida, son las siguientes:

- RTC DS1307 I2C
- Interfaz I2C
- Compensación de año bisiesto.
- Calendario preciso hasta el año 2100.
- Respaldo de batería incluido.
- Pin de 1Hz de salida.
- 56 Bytes de memoria RAM disponible.
- Alimentación externa: 5V
- Dimensiones: 20 x 20 mm

5.2.2. Conexionado.

Este módulo se comunica con Arduino mediante el protocolo de comunicaciones I2C. La conexión de los distintos pines al microcontrolador correspondiente es la siguiente:

Señal	ATmega328	ATmega32u4
GND	GND	GND
VCC	5V	5V
SDA	A4 o SDA	D2 o SDA
SCL	A5 o SCL	D3 o SCL
SQW	NO	NO

Tabla 5.2. Conexionado RTC

Señalar que se puede realizar la conexión I2C mediante los pines digitales D2 y D3 o los pines dedicados a ello en la placa, como se muestra en la figura 5.5.

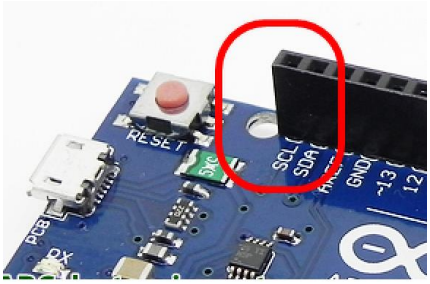


Figura 5.5. Pines SCL/SDA

En este caso, el módulo RTC irá colocado solamente en el nodo central, es decir, en el Arduino Yún. Por tanto, se realizará la conexión que aparece en la segunda columna, en concreto se ha optado por conectarlo de la siguiente forma:

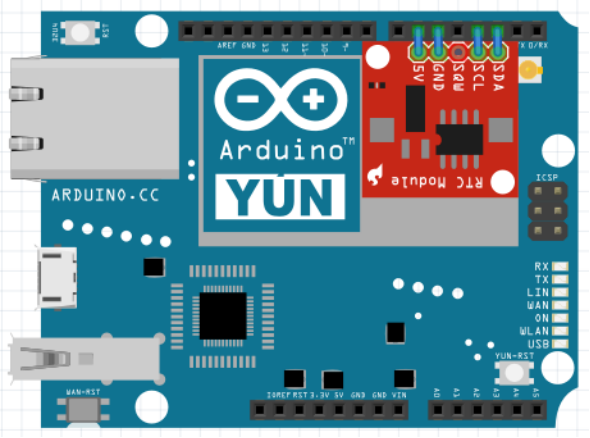


Figura 5.6. Conexión RTC-Arduino YUN

Donde los pines D5 y D6, son GND y 5V, respectivamente, configurando previamente en el código éstos pines como “outputs” (salidas) y en con valor “0” y “1”.

5.3. Sensor de humedad y temperatura SHT10.

5.3.1. Descripción.

Este dispositivo, desarrollado por Adafruit, incluye un sensor de humedad y temperatura, fabricado por Sensirion, dentro de un encapsulado mallado metálico. La carcasa es resistente a la intemperie y evita que el agua se filtre y dañe el sensor, pero permite que el aire pase y gracias a esto puede medir la humedad del suelo.

Figura 5.7. Sensor SHT10 (Humedad/Temperatura)

Está diseñado para ser sumergible en el agua, pero no para un largo periodo (no más de 1 hora seguida) por lo que si se prevée que puede estar sumergido durante un largo periodo de tiempo se debe buscar otra alternativa.

Los SHT1x (SHT10, SHT11 y SHT15) son una familia de sensores, de Sensirion, de humedad y temperatura. Los SHT1x se calibran individualmente de fábrica. Los coeficientes de calibración se programan en una memoria OTP (One Time Programmable) en el chip, estos coeficientes se utilizan para calibrar internamente las señales de los sensores. La interfaz de serial de 2hilos y la regulación de tensión interna permite la integración del sistema de una forma fácil y rápida. Las características más importantes, del modelo que se ha seleccionado (SHT10), son las siguientes:

- Precisión de Humedad: ±4.5 [%RH]
- Precisión de Temperatura: ±0.5ºC @25ºC
- Tensión de alimentación: 3-5 VDC
- Rango de Temperaturas: -40 a 120ºC
- Rango de Humedades: 0-100 %RH
- Consumo: 80uW (12 medidas, 1 vez/s)
- Tiempo de respuesta: 8s (Humedad); 5-30s (Temperatura).
- Dimensiones del encapsulado: 14mm diametro, 50mm de largo.
- Longitud del cable: 1 metro
- Otros: Completamente calibrado, salida digital, excelente estabilidad.

5.3.2. Conexionado.

La conexión de este dispositivo al microcontrolador se hace a través de las entradas digitales de éste. El sensor cuenta con cuatro cables VCC, GND, DATA y CLOCK, como se aprecia en la figura 5.8.

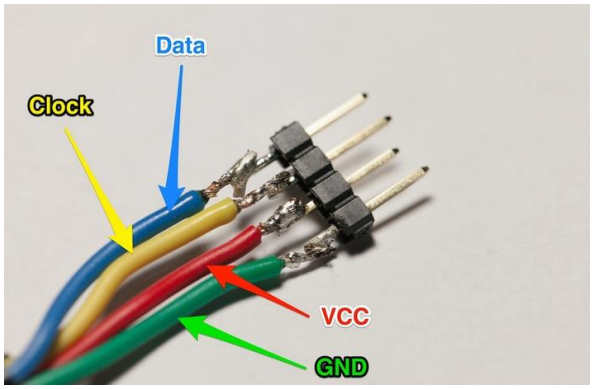


Figura 5.8. Cables SHT10

Por lo que:

Señal	**Arduino**
GND	GND
VCC	5V
DATA	Pin digital (D0-D13)
CLOCK	Pin digital (D0-D13)

Tabla 5.3. Conexionado SHT10

El fabricante recomienda colocar una resistencia de pull-up de 10K entre VCC y DATA, por lo que hay que aplicar este pequeño acondicionamiento para realizar las medidas correctamente.

5.4. Sensor de humedad y temperatura DHT22.

5.4.1. Descripción.

Se trata de un pequeño dispositivo que permite medir la humedad y la temperatura. En concreto en este proyecto se usarán para determinar la humedad y temperatura ambiente.

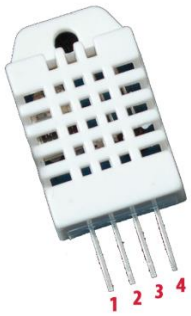


Figura 5.9. Patillaje (pinout) DHT22

Al igual que el sensor anterior, la señal de salida es digital, por lo que la conexión se realizará con los pines digitales, y además incorporan un pequeño microcontrolador interno para hacer el tratamiento de la señal. El funcionamiento es muy sencillo, los DHT22 se componen de un sensor capacitivo para medir la humedad y de un termistor, ambos sensores están calibrados por lo que no se debe ajustar ningún parámetro.

Las características de éste sensor son las siguientes:

- Precisión de Humedad: ±2 [%RH]
- Precisión de Temperatura: ±0.5ºC @25ºC
- Tensión de alimentación: 3.3 - 6 VDC
- Rango de Temperaturas: -40 a 80ºC
- Rango de Humedades: 0-100 %RH
- Tiempo de respuesta: 2s
- Dimensiones del encapsulado: 14x18x5.5mm

5.4.2. Conexionado.

Se debe colocar una resistencia de pull-up de 10K entre los pines 1 y 2 (Vcc y Data), el resto de conexiones se realizarán como indica la siguiente tabla:

DHT22	**Arduino**
1.- VCC	5V
2.- Datos	Pin digital (D0-D13)
3.- No conectado	-
4.- GND	GND

Tabla 5.4. Conexionado DHT22

5.5. Conversor ADC 12bits ADS1015.

5.5.1. Descripción.

Este dispositivo proporciona una mayor resolución que la obtenida mediante el conversor ADC de 10bits de Arduino, esto será de gran utilidad en algunos sensores que se describirán más adelante.

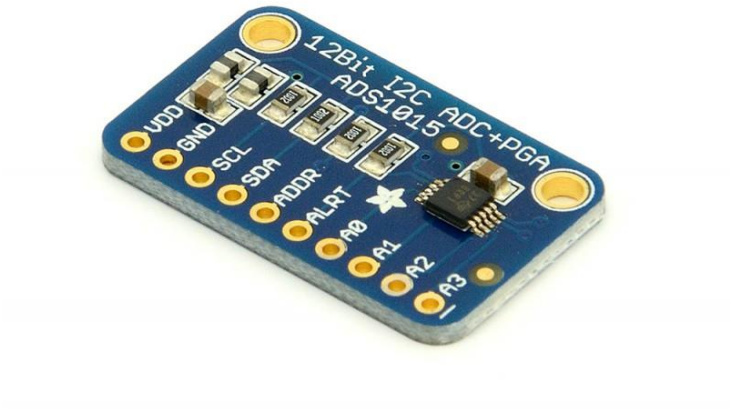


Figura 5.10. ADS1015 - ADC 12bits

El ADS1015 proporciona una resolución de 12 bits con 3300 muestras/segundo vía I2C. El chip puede ser configurado como 4 canales “single-ended” o como 2 canales diferenciales. Además incorpora un amplificador de ganancia programable, hasta x16, para elevar las pequeñas señales a todo el rango.

El resto de sus características son:

- Tensión de alimentación: 2V – 5V
- Consumo: 150uA (Modo continuo); Auto-Apagado (Modo Single-Shot)
- Tasa de datos: 128SPS – 3.3kSPS
- 4 entradas single-ended o 2 entradas diferenciales
- Comparador programable
- Usa direcciones I2C de 7bit entre 0x48 – 0x4B

5.5.2. Conexionado.

El ADS1015 irá conectado mediante I2C a arduino por tanto la conexión es parecida a la que tenemos para el módulo RTC, pero en este caso el dispositivo se colocará en un Arduino Uno:

Señal	**ATmega328**	**ATmega32u4**
VDD	5V	5V
GND	GND	GND
SCL	A5 o SCL	D3 o SCL
SCA	A4 o SDA	D2 o SDA
ADDR	NO	NO
ALRT	NO	NO
A0, A1, A2 y A3	Aquí se conectarán los sensores con resolución de 12bits. Se tienen dos modos: single-ended y diferencial	

Tabla 5.5. Conexionado ADS1015.

5.6. Sensores de irradiancia S1087 y S1087-01.

5.6.1. Descripción.

Las series S1087 [13] son fotodiodos encapsulados que ofrecen una salida proporcional a luz que les llega. El encapsulado cerámico es impermeable a la luz, por lo que no hay luz dispersa que pueda llegar a la zona activa de la parte trasera. Esto permite mediciones ópticas fiables en un amplio rango.

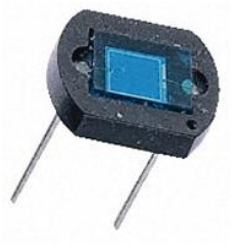


Figura 5.11. Sensor S1087

La diferencia entre los dos sensores utilizados es el espectro que abarcan en el caso del modelo S1087 desde 320 a 730nm, mientras que el S1087-01 tiene un rango más amplio desde 320 a 1100nm.

El primero será utilizado para obtener los datos correspondientes a la fotosíntesis ya que las radiaciones con longitudes de onda menores de 400nm (como la luz ultravioleta) y mayores de 700 (como las infrarrojas) pueden tener diversos efectos biológicos, pero no pueden ser aprovechadas para la fotosíntesis.

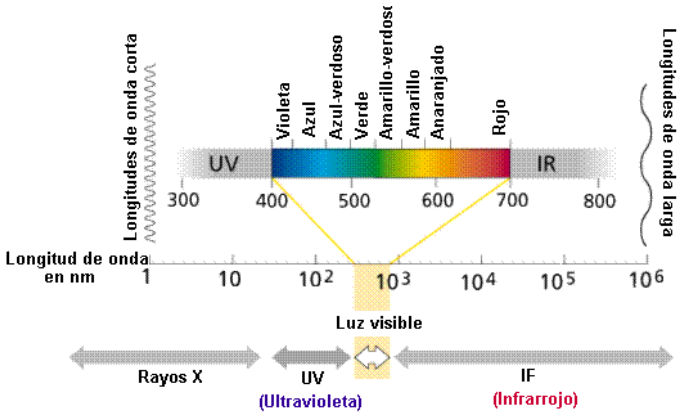


Figura 5.12. Espectro de la radiación visible

En realidad, las encargadas de recoger la energía de la luz son las clorofilas, una familia de pigmentos de color verde, que tienen dos picos de absorción en el espectro visible, uno en el entorno de la luz azul (400-500nm) y otro en la zona roja (600-700nm), sin embargo reflejan la parte media del espectro, la correspondiente al color verde. Esta es la razón por la que las clorofilas tienen color verde, en la figura 5.13 se muestra claramente:

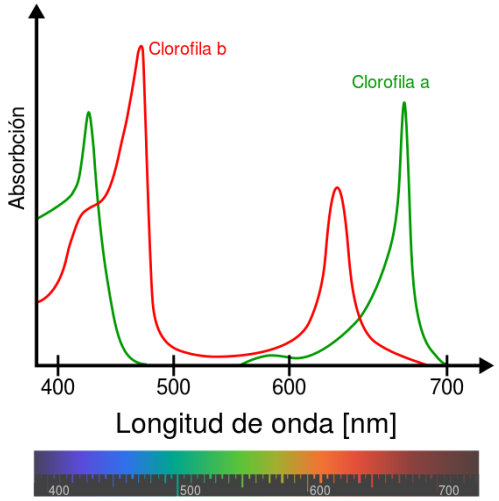


Figura 5.13. Absorción energía clorofilas.

Por otro lado, el S1087-01 se utilizará para medir la radiación total, algo que servirá, entre otras cosas, a determinar si es viable, o no, colocar placas solares en un futuro.

A continuación se detallarán las características de ambos modelos [14]:

Modelo	Dimensiones (mm)	Área efectiva (mm^2)	V.retorno Vr (V)	R.Temp de operación (ºC)	R. Temp. almacenamiento (ºC)
S1087	1,3 x 1,3	1,6	10	-10 a 60	-20 a 70
S1087-01	1,3 x 1,3	1,6	10	-10 a 60	-20 a 70

Tabla 5.6. Características principales de los sensores S1087 y S1087-01.

Modelo	Longitud de onda λ (nm)	Sensibilidad longitud de onda pico λp (nm)	Isc a 100lux (uA)
S1087	320 a 730	560	0,16
S1087-01	320 a 1100	960	1,4

Tabla 5.7. Características eléctricas de los sensores S1087 y S1087-01.

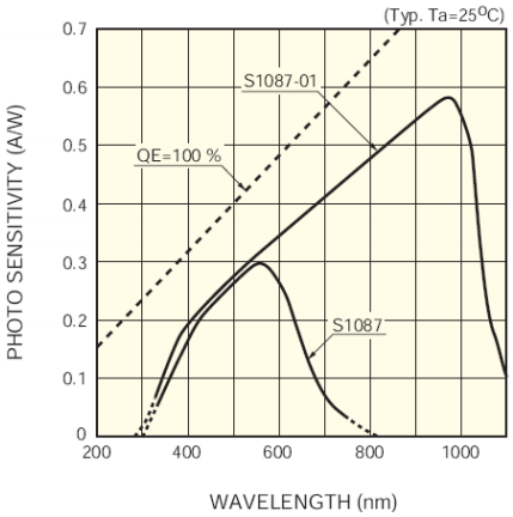


Figura 5.14. Sensibilidad (A/W) - Longitud de onda (nm)

5.6.2. Conexionado.

La conexión no se realizará directamente a arduino ya que se utilizará el conversor ADC 12 bit descrito en el apartado anterior, esto permitirá tener una mayor resolución y obtener resultados más precisos.

Por tanto, se implementará el siguiente circuito básico:

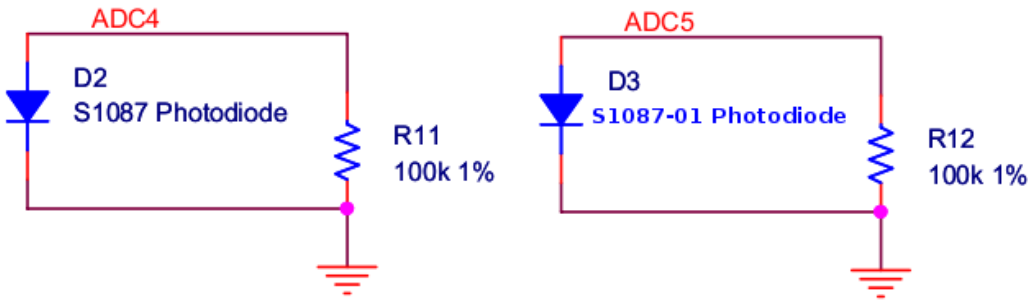


Figura 5.15. Conexión S1087 - ADC ADS1015

Más adelante en la parte referente al software se explicará el código correspondiente a estos sensores.

5.7. Sensor de temperatura LM35.

5.7.1. Descripción.

El LM35 es un sensor de temperatura con una precisión calibrada de 1ºC. Este elemento lo se usará para realizar un seguimiento de la temperatura alcanza en cada nodo, sobre todo de los expuestos al sol, para evitar problemas de sobrecalentamiento.

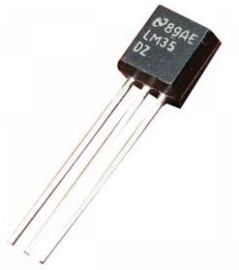


Figura 5.16. Sensor LM35

Las características más importantes de este sensor son:

- Tensión de alimentación: 4V – 30V
- Rango de temperaturas: -55ºC a 150ºC
- Consumo: 60uA
- Salida lineal proporcional a la temperatura (10mV/ºC)
- Tiene una precisión garantizada de 0,5ºC a 25ºC
- Bajo auto-calentamiento (0,08 ºC en aire estático)
- Baja impedancia de salida, 0,1W para cargas de 1mA
- Bajo coste

5.7.2. Conexionado.

La conexión de este elemento es muy sencilla simplemente se tienen tres patillas: Vs, Vout y GND. Como se muestra en la figura 5.17:

Figura 5.17. Patillas LM35

Donde Vs se conectará a los 5V de Arduino, GND con GND de Arduino y Vout con cualquiera de las entradas análogas de Arduino.

5.8. Relé 5VDC – 250VAC 10A.

5.8.1. Descripción.

Este dispositivo se utilizará para controlar la alimentación de los nodos secundarios. Cada 15 minutos se permitirá la alimentación de los nodos secundarios que, tras realizar las medidas oportunas, serán desconectados nuevamente.

Figura 5.18. Relé

Este modelo posee las siguientes características:

- Necesita 15-20mA para excitar el driver.
- Contactos independientes para proteger y aislar el circuito.
- Dimensiones: 43x27mm

5.8.2. Conexionado.

La conexión de este dispositivo es muy sencilla, por un lado, simplemente se debe configurar una patilla digital como salida y activarla o desactivarla cuando convenga.

Pin	**Arduino**
GND	GND
VCC	5V
IN	D13 (variable)

Tabla 5.4. Conexionado Relé – Parte 5V

Por el otro lado, se conectará el cable que alimentará a los nodos secundarios de la siguiente forma:

Pin	**Cable**
COM	Negativo
NA	Positivo

Tabla 5.5. Conexionado Relé – Parte 220V

Capítulo 6.

Arquitectura Software

En este capítulo se describirá la parte referente al software del presente proyecto, en primer lugar se realizará un pequeño análisis de las diferentes plataformas para IoTs (Internet of Things) que se pueden utilizar, para entender con mayor claridad nuestra elección.

Luego se explicará el funcionamiento global del sistema, mostrando una imagen descriptiva de la distribución final y detallando las funciones que realizará cada nodo así como la información que enviará al nodo principal.

Tras esto, se describirán las distintas vías que se pueden seguir para obtener datos del sistema, y por último, se mostrarán que procedimientos se pueden seguir para analizar los resultados del sistema.

6.1. Plataformas IoTs.

Existe una inmensa cantidad de plataformas dedicadas al Internet de las Cosas, en este apartado se analizarán las más comunes y/o conocidas, se realizará una breve descripción de cada una de ellas haciendo hincapié, sobretodo, en los límites que tiene.

6.1.1. Xively.

Xively [15], anteriormente conocida como Cosm (antes Pachube), es un servicio propiedad de la empresa LogMeIn. Xively se define como una “Plataforma como un Servicio” (PaaS) para el Internet de las Cosas.

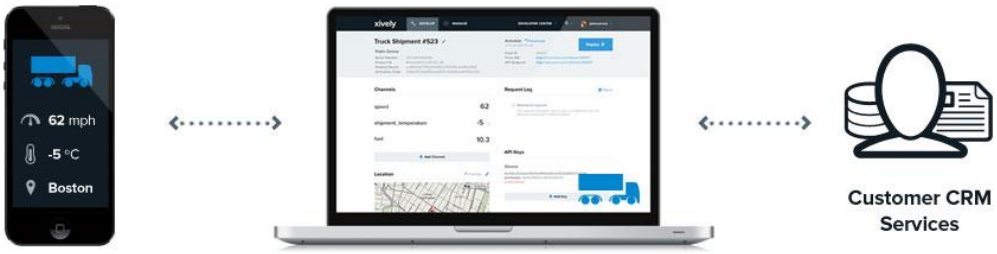


Figura 6.1. Xively

Posee un servicio de suscripción enfocado a las empresas que deseen utilizar la plataforma como un back-end para sus productos conectados a Internet. El coste es de 999 € por una licencia anual y además se debe pagar entre 1 y 3 € por dispositivo conectado, dependiendo de la cantidad de datos que produzca el dispositivo.

La principal ventaja de este servicio es la facilidad de uso y su rápida integración en cualquier sistema que se desee monitorizar.

Por otro lado, ofrece un servicio básico de manera gratuita pero que solamente almacena los datos de un mes. Con un límite de 25 llamadas a la API por minuto.

6.1.2. Plotly.

Se trata de una plataforma recién nacida, enfocada a la representación de manera gráfica de los datos almacenados. Permite cargar ficheros en formato Excel, CSV, TSV, Matlab y muchos más.

Tiene implementados una serie de algoritmos y herramientas como los que se encuentran en Excel pero de manera totalmente gratuita. Las gráficas creadas son totalmente interactivas y muy variadas.

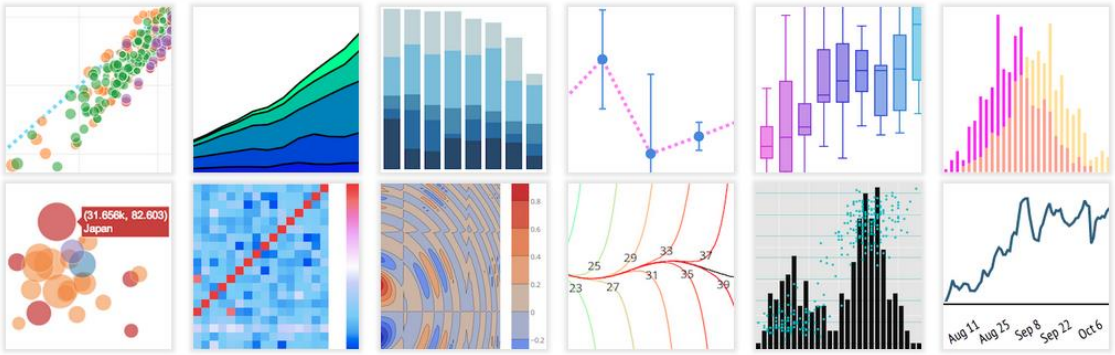


Figura 6.2. Plotly

Además ofrece una serie de librerías para tomar los datos directamente desde Arduino y otras aplicaciones como MATLAB, R, Julia…

La mayor virtud de esta plataforma radica en su sencillez y su aspecto elaborado, esto dota a los proyectos de un valor añadido.

En cuanto a las limitaciones de esta plataforma vienen determinadas en función del plan escogido como se muestra en la figura 6.3:

	Basic	Premium	Organizational
Price	Free	$12 / month (free with 5 referrals)	$10 / month / User
Public files	Unlimited	Unlimited	Unlimited
Private files Only you and your collaborators can access	20	Unlimited	Unlimited
Account management			
Centralized billing	✖	✖	✔
Add or remove users	✖	✖	✔
	Sign up	Subscribe	Subscribe

Figura 6.3. Planes de suscripción Plotly

Como se observa existen tres planes diferentes, si se observa el plan gratuito, éste ofrece un número ilimitado de ficheros públicos pero limita los ficheros privados. Aun así las limitaciones de esta plataforma son mucho menores que las de Xively.

6.1.3. Carriots.

Carriots es una plataforma española, se trata de una spin off de Wairbut, empresa española de ingeniería y desarrollo.

Figura 6.4. Funcionamiento Carriots

Como se muestra, su funcionamiento es similar al de las demás plataformas además ofrece distintos planes de suscripción muy parecidos a los que ofrece Xively, como se muestra en la figura 6.5.

GRATIS	EMPRESARIAL	NUBE PRIVADA
Gratis (NO HACE FALTA TARJETA)	2 € AL MES POR DISPOSITIVO*	Contacte con nosotros SI QUIERE QUE CARRIOTS SE EJECUTE EN SU NUBE PRIVADA
Número máx. Dispositivos **10**	Número máx. Dispositivos **Ilimitado**	Número máx. Dispositivos **Ilimitado**
API KEYS **2**	API KEYS **Ilimitado**	API KEYS **Ilimitado**
Max. Tramas aceptadas **15000 tramas / día** **500 tramas / minuto**	Max. Tramas aceptadas **Ilimitado**	Max. Tramas aceptadas **Ilimitado**
Max. Datos almacenados **1 año online** **Ilimitado offline**	Max. Datos almacenados **1 año online** **Ilimitado offline**	Max. Datos almacenados **Ilimitado**
SMS API **5 SMS / día** **1 SMS / minuto**	SMS API **Gratis 5 SMS / día** **>6 SMS 0.1€* / unit.**	SMS API **Contacte con nosotros**
Email API **100 Email / día** **10 Email / minuto**	Email API **Gratis 100 Email / día** **>100 0.50€* / millar**	Email API **Contacte con nosotros**
SDK Http Request (Salida) **15000 Req. / día**	SDK Http Request (Salida) **45000 Req. / día**	SDK Http Request (Salida) **Ilimitado**
Soporte Básico: Atención vía Email con nuestro máximo esfuerzo	**Soporte Empresarial:** Atención telefónica - **Contacte con nosotros** Atención via Email en menos de 24 horas	**Soporte Premium:** Diferentes planes disponibles **Contacte con nosotros**

Figura 6.5. Planes de suscripción Carriots

Se aprecia como sus características son similares a las anteriores plataformas pero ofrece dos funciones muy interesantes: SMS API y Email API. Estas funciones son realmente útiles para, por ejemplo, establecer alertas cuando un sensor alcance un límite preestablecido, o para enviar los datos tomados mediante correo electrónico…

6.1.4. ThingSpeak.

ThingSpeak se caracteriza porque se trata de una plataforma open source, esto tiene una serie de ventajas, como, por ejemplo, que es posible usar el código compartido por ThingSpeak para implementarlo en nuestra propia página web.

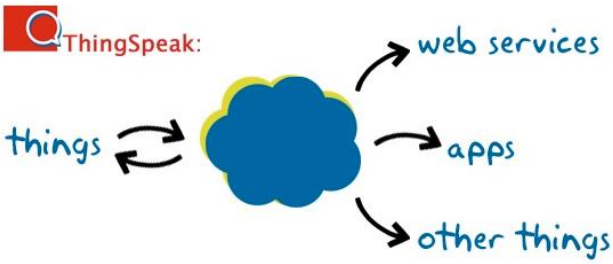


Figura 6.6. ThingSpeak

Ofrece una serie de tutoriales y documentación sobre como conectar distintos objetos y servicios. Además dispone de una serie de aplicaciones integradas como:

- ThingTweet: Conecta una cuenta de Twitter para enviar mensajes.
- ThingHTTP: Para enlazar con otros Web services y compartir los datos.
- TweetControl: Serie de comandos recibidos vía twitter que realizan una acción.
- React: Realiza acciones cuando se cumplen unas condiciones preestablecidas.
- Talkback: Permite a los dispositivos ejecutar comandos en cola.

ThingSpeak permite descargar los datos de los dispositivos conectados en formatos: XML, CSV o JSON.

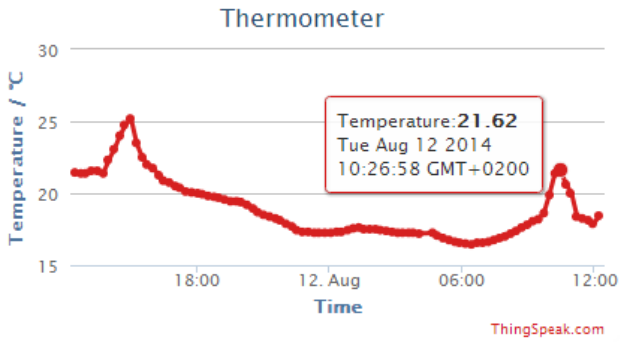


Figura 6.7. Gráfica generada con ThingSpeak

Como se observa en la figura 6.7, las gráficas son interactivas y permiten saber el valor exacto de cada punto. Estas gráficas están disponibles en los distintos canales de nuestra cuenta, los cuales se pueden establecer como públicos o privados, en cada canal se establecen hasta 8 gráficas, y además se pueden añadir secciones de video, ubicación y plugins.

Por otro lado, como se ha comentado anteriormente esta plataforma es open-source y no posee las limitaciones que tenían las anteriores alternativas, sin embargo, tampoco ofrece tantas características como las anteriores pero aun así es sin duda la mejor alternativa para este proyecto.

6.2. Distribución del Sistema.

El sistema planteado, como se ha comentado anteriormente, dispone de una serie de nodos secundarios que recogen datos de diferentes sensores, tras obtener estos datos y procesarlos, los envían al nodo principal que se encargará de organizar la información procedente de los demás nodos, guardarla adecuadamente en la tarjeta microSD y subirla a una plataforma de IoTs.

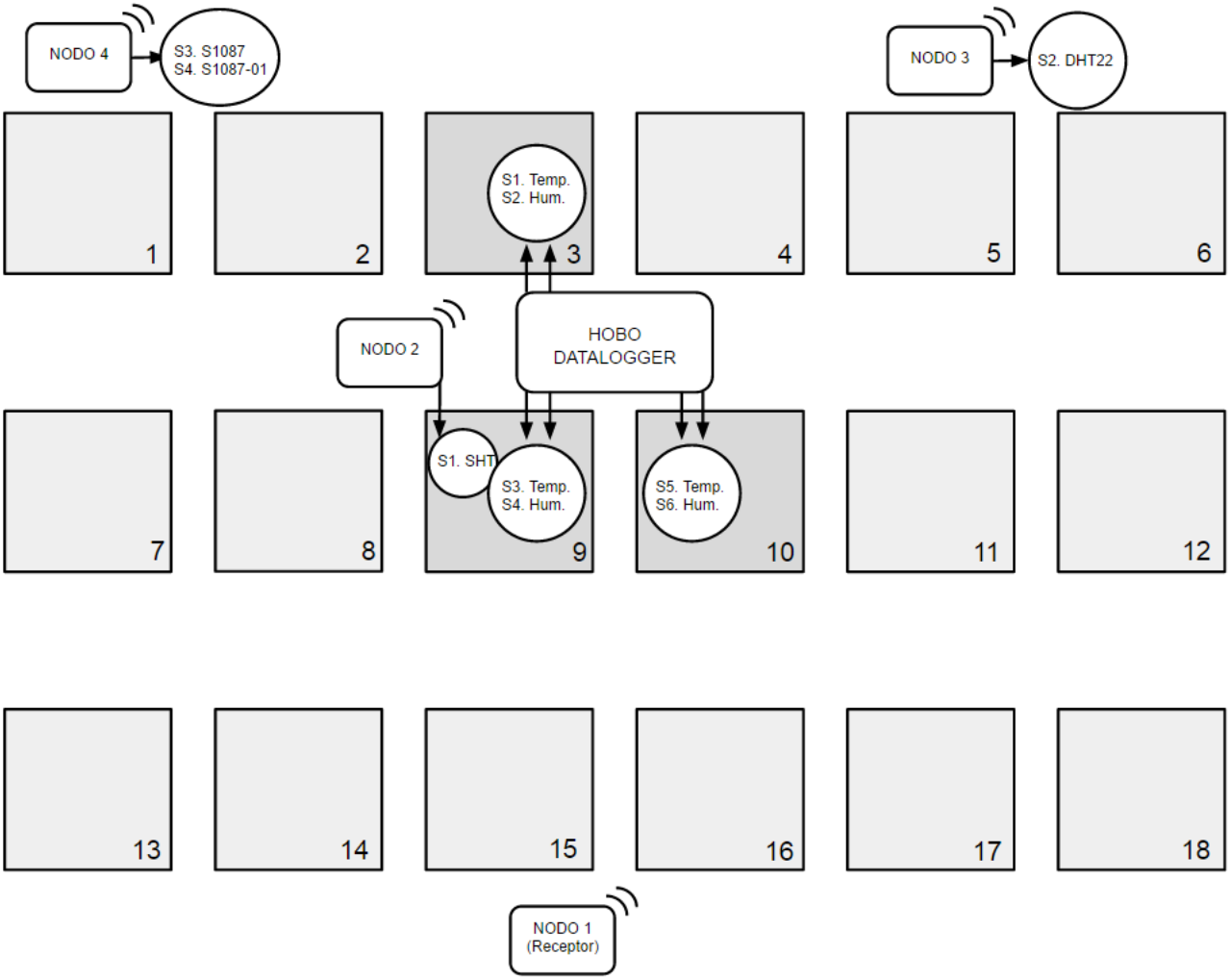


Figura 6.8. Distribución del sistema (final).

Se muestra como tras la inclusión de los nuevos elementos se tiene un sistema más completo que el inicial, y que aún se puede ampliar más. También, se puede ver como en la caja número 9 se tienen los sensores del sistema inicial y el sensor nuevo (SHT), esto permitirá comparar los resultados y comprobar si los resultados obtenidos por el nuevo sensor son veraces.

Como se observa en la representación anterior, se tendrán dos tipos de nodos, el central, y los secundarios. El funcionamiento general se puede describir de la siguiente forma:

1. El nodo central (nodo 1) se configura como emisor y los demás como receptores.
2. El nodo central envía una señal al nodo 2.
3. Se invierten las configuraciones, el central pasa a receptor y el nodo 2 a emisor.
4. El nodo central recibe los datos y los procesa.
5. Se repite el proceso tantas veces como nodos haya.

Por tanto, serán necesarios dos tipos de código, principalmente, el código correspondiente al nodo central y el código correspondiente a cada uno de los secundarios, donde solo variará la lectura que realice cada sensor y por tanto el tipo de datos que enviará. A continuación se detallarán el funcionamiento exacto de cada nodo.

6.2.1. Nodo 1 – Nodo central.

En primer lugar Arduino Yún activará la señal del relé para así alimentar a los nodos secundarios.

Tras alimentarlos, y esperar durante unos segundos para que los nodos se inicialicen, Arduino Yún se encargará de realizar una petición de los datos a cada nodo secundario, comprobará si la respuesta es adecuada y guardará los datos recibidos en la tarjeta microSD, junto con la fecha actual obtenida del módulo RTC.

Tras recibir los datos pertinentes de todos los nodos se realizará la subida de datos a ThingSpeak mediante una función específica que utiliza la parte Linux de Arduino Yún.

Por último, desactivará el relé, y esperará durante 15 minutos para volver a repetir el proceso.

6.2.2. Nodos 2, 3 y 4.

El comportamiento de estos es muy similar, primero, realizan las lecturas de los sensores, después, comprueban que los datos obtenidos por los sensores se encuentren en el rango de los mismos. Si no se encuentra en el rango del sensor se sustituirá el valor por 0 para así identificar el problema más fácilmente en el fichero donde se guarden.

Tras esto pasarán a modo “escucha”, es decir, esperarán a que el nodo principal le pida el dato para enviarlo.

En concreto los nodos serán:

- Nodo 2 - Sensor de Humedad/Temperatura del Suelo.

  En este nodo se colocará el sensor SHT10 descrito anteriormente, este sensor es el más importante, pues realiza el seguimiento de la humedad/temperatura de la plantación. Algo que mejorará la optimización del riego y a analizar su evolución.

- Nodo 3 - Sensor de Humedad/Temperatura Ambiente.

  Aquí se colocará un sensor DHT22, que tiene unas especificaciones buenas teniendo en cuenta su bajo coste.

  Esta variable permitirá analizar el entorno en el que se encuentra la plantación monitorizada.

- Nodo 4 - Sensores de radiación.

  En este nodo se colocarán los sensores S1087 y S1087-01 analizados en el capítulo anterior.

  Por un lado, el sensor S1087-01 obtendrá los datos de la radiación total, estos datos podrán ser utilizados, por ejemplo, para saber cuántas células fotovoltaicas serían necesarias para alimentar al sistema.

  Por otro lado, el sensor S1087 obtendrá los datos de radiación correspondientes al espectro de la fotosíntesis (luz visible), unos datos necesarios para comprender el comportamiento de la plantación.

6.3. Obtención de datos.

Para obtener los datos de manera inalámbrica se tienen diferentes alternativas: acceder a los datos almacenados en ThingSpeak, acceder a la dirección que crea Arduino Yún desde el navegador o usar un programa que comunique con Arduino Yún.

Aunque si no existe posibilidad de establecer una comunicación inalámbrica, también se puede acceder a la tarjeta microSD físicamente. A continuación se detallarán cada una de estas posibilidades.

6.3.1. Vía ThingSpeak.

Este método es el más sencillo de todos pues consiste en entrar a la web de esta plataforma, acceder a la página del canal, presionar sobre “developer info”, y aparecerá la siguiente ventana:

Figura 6.9. Descargar datos ThingSpeak

En esta ventana se encuentran todos los datos almacenados en el canal, que se pueden descargar en diversos formatos. Tras la descargar se puede importar el fichero a una hoja de cálculo y realizar las gráficas pertinentes.

Pero esta alternativa también tiene ciertas desventajas, por ejemplo, en caso de que se haya perdido la conexión WiFi, el dispositivo lo seguirá almacenando en la tarjeta microSD pero le será imposible enviarlo a ThingSpeak.

6.3.2. Vía navegador web.

Este método aprovecha una de las ventajas de Arduino Yún, se debe resaltar que este método solo funcionará cuando se esté conectado a la misma red que el dispositivo.

Básicamente se trata de acceder, en el navegador, a la dirección:

http://arduino.local/sd/

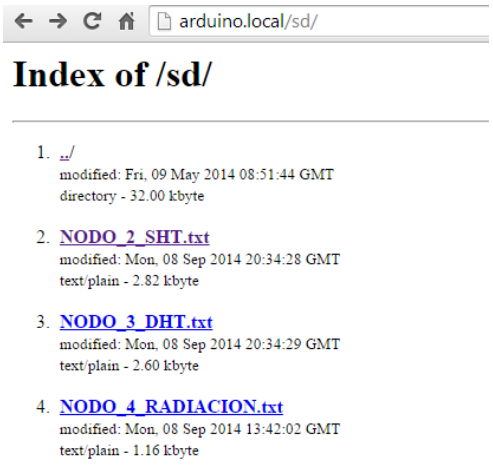


Figura 6.10. Acceso a Yún vía web

Como se observa aparece una página web, muy básica, donde aparecen los ficheros disponibles en la tarjeta microSD. Si se selecciona cualquiera de estos ficheros se podrá ver los datos guardados:

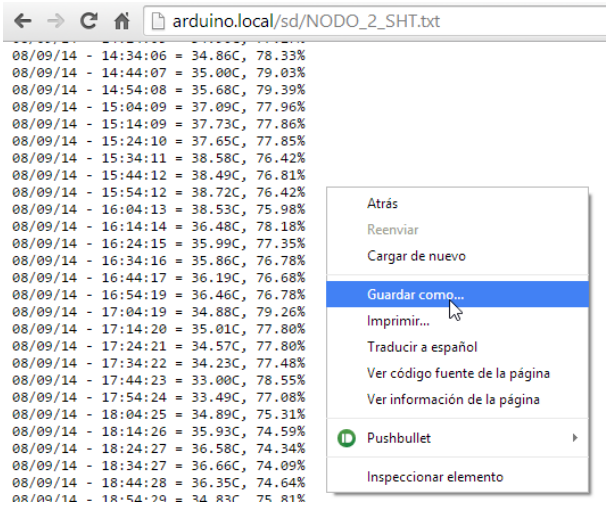


Figura 6.11. Guardar datos vía web

Como la anterior este método no funciona si no existe una conexión a la misma red WiFi que Arduino Yún.

6.3.3. Vía cliente FTP.

Este método es posible, gracias a la parte Linux que posee Arduino Yún se pueden utilizar diversos programas que se conectan a su sistema de ficheros de forma remota, como por ejemplo, Filezilla, WinSCP o SSHfs Manager. Para ello antes se realizarán una serie de procedimientos que se detallan a continuación:

1) Acceder a la terminal de Yún de forma remota (SSH), desde Windows se puede acceder mediante el programa “puTTY”

2) Introducir en el terminal los siguientes comandos:

Comando	**Comentario**

`ssh -l root X`
 	
`vi /etc/dropbear/authorized_keys`
 	
`chmod 0600 /etc/dropbear/authorized_keys`
 	
`opkg update`
 	
`opkg install openssh-sftp-server`
	X es la IP de acceso a Yún

Figura 6.12. Comandos servidor FTP

3) Por último, se aceptan todas las peticiones que aparecen

Como se ha comentado anteriormente, existen multitud de programas con los que se podrían establecer conexión a Arduino Yún pero el más adecuado para este caso es SSHfs Manager.

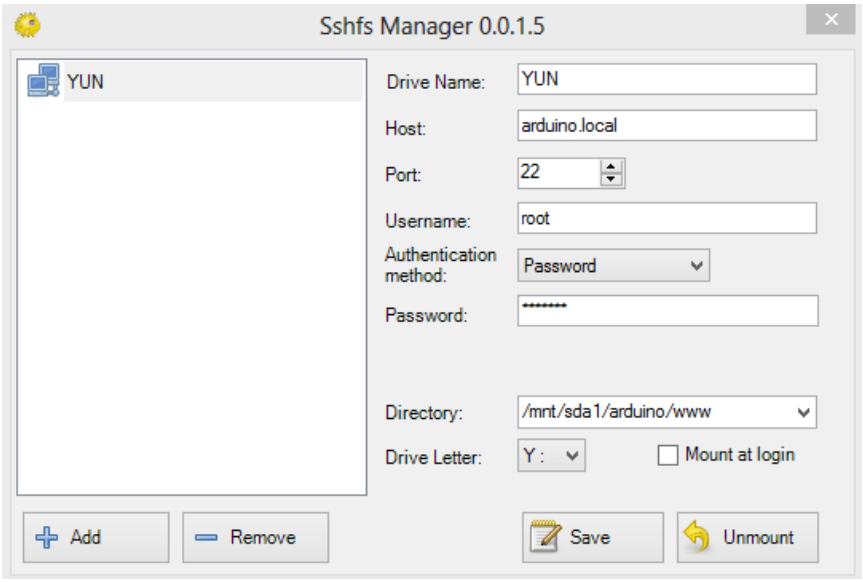


Figura 6.13. Configuración Sshfs Manager

La configuración es muy sencilla, lo que se hará simplemente será establecer la conexión con la memoria microSD y el programa cargará esta dirección como un dispositivo externo del sistema, como si se tratara de un USB, con las ventajas que esto supone.

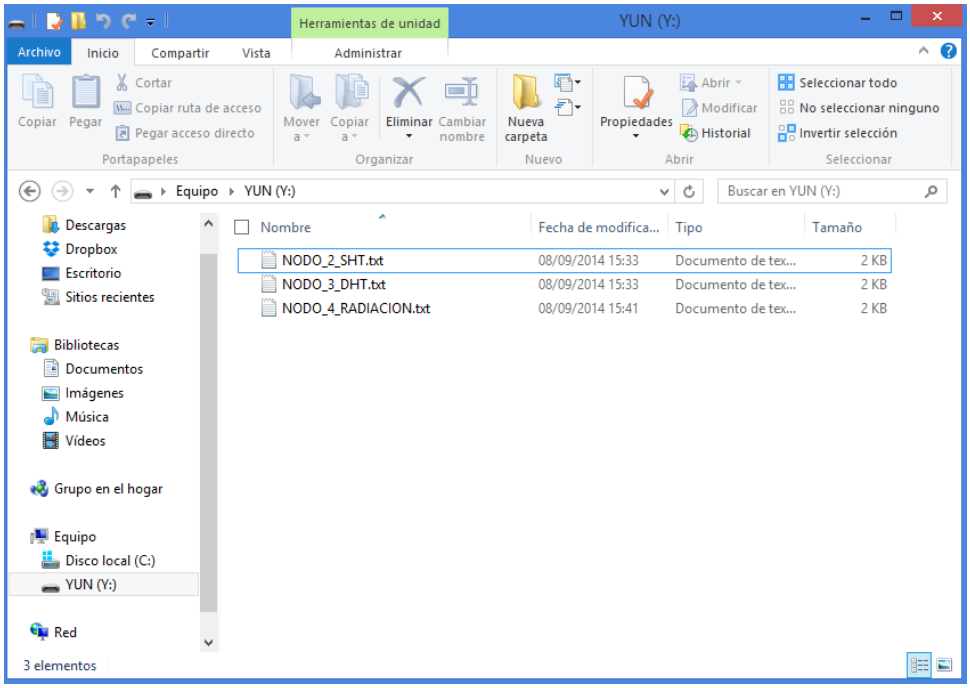


Figura 6.14. Acceso a Yún con Sshfs Manager

Una vez configurado correctamente este método es sin duda una de las mejores opciones para obtener los datos, puesto que su uso es idéntico al de manejar una memoria USB con la ventaja de la comunicación inalámbrica.

6.3.4. Vía tarjeta microSD.

Este método es similar al anterior, salvo que para llevarlo a cabo se debe extraer la tarjeta microSD de Arduino Yún y conectarla mediante un adaptador al PC. Este es método menos recomendable pues supone tener que desmontar los elementos de protección que se colocan en la instalación.

Figura 6.15. Adaptadores microSD

Cuando se vaya a extraer la tarjeta microSD, antes, se debe desconectar el dispositivo de la corriente para impedir que se pierdan datos.

6.4. Análisis y procesamiento de datos.

6.4.1. ThingSpeak.

Como ya se ha comentado anteriormente, los datos de este sistema se encuentran alojados en ThingSpeak, esta plataforma dispone de diversas herramientas y en ella se pueden representar los datos.

Por lo que, una forma de analizar los datos sería visualizar éstos en la página web, además las gráficas son interactivas como se muestra en la figura 6.16:

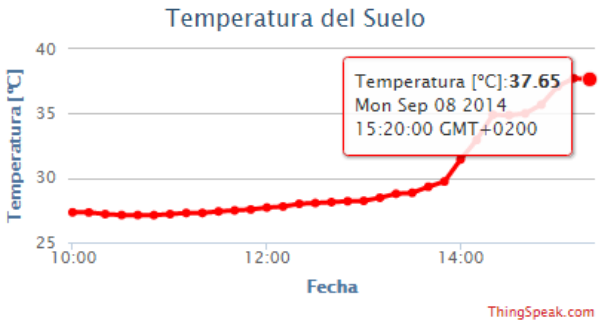


Figura 6.16. Gráfica "Temperatura del Suelo" ThingSpeak

Estas gráficas son totalmente personalizables, tan solo se debe presionar el botón de opciones y configurarla adecuadamente.

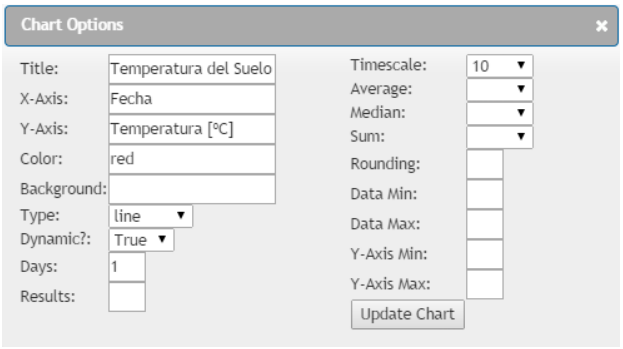


Figura 6.17. Opciones para gráficas en ThingSpeak

Esto posibilita una herramienta útil y accesible para el correcto análisis de este sistema.

6.4.2. Hoja de cálculo.

Otra forma de analizar los datos es mediante el uso de algún programa de hoja de cálculo, como, por ejemplo, Excel, Google SpreadSheets, Calc (Libreoffice)…

Por ejemplo, con Excel se puede crear una tabla dinámica, la cual tiene unas funciones muy interesantes:

- Permite realizar un filtro de los datos:

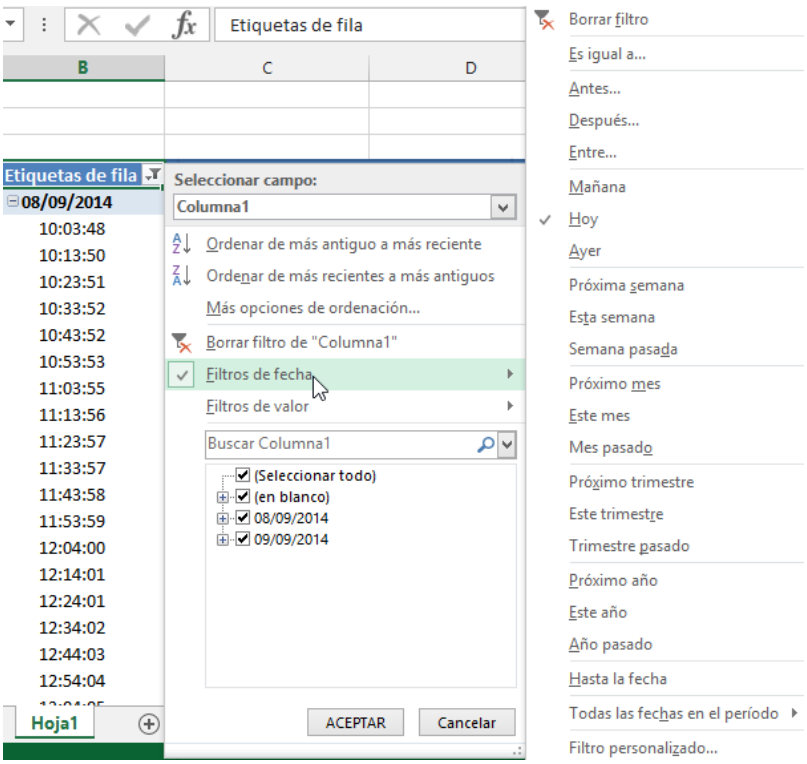


Figura 6.18. Filtro de datos en Excel

- Actualizar los datos desde el fichero de forma periódica:

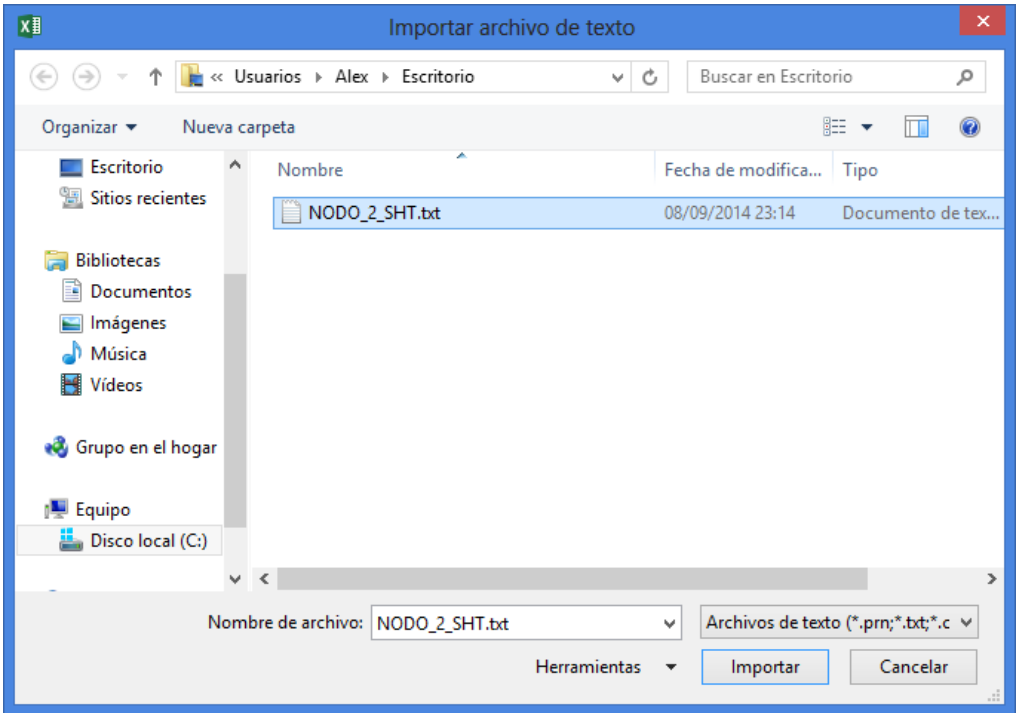


Figura 6.19. Importar archivo de texto a Excel

Una vez creada la tabla dinámica, se puede crear una gráfica dinámica que dependa de los datos de la tabla anterior, así se consigue que cuando se actualicen los datos de la tabla, la gráfica haga lo propio.

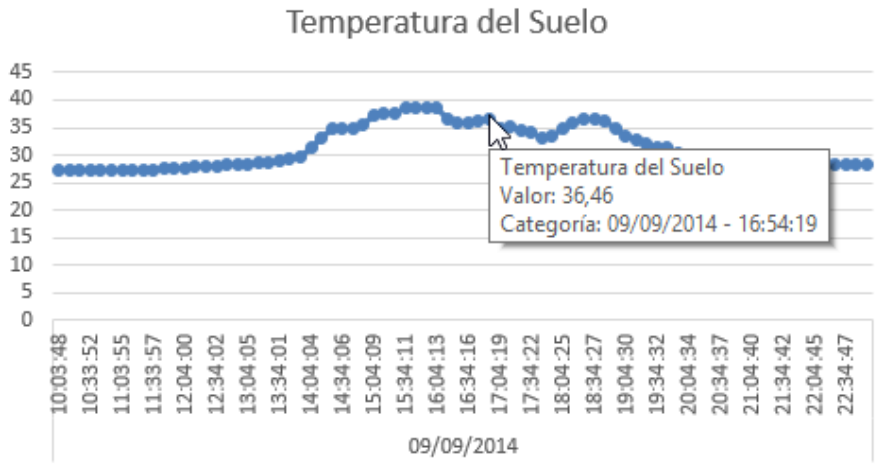


Figura 6.20. Gráfica dinámica Temperatura del Suelo

Este método es muy completo ya que pone a nuestra disposición todas las herramientas de Excel, algo que para las personas que utilizarán los datos obtenidos les resultará más sencillo que adaptarse a otros programas.

Capítulo 7.

Resultados, conclusiones y trabajos futuros

En este capítulo se analizará por completo el sistema final para comprobar su robustez y que su comportamiento es el adecuado.

Se intentará comprometer el sistema, como, por ejemplo, desconectando sensores, para ver su comportamiento ante errores.

El objetivo es mostrar que la solución propuesta es fiable y rentable, robusta y sencilla. Esto es sin duda un aspecto muy importante pues aunque el contenido sensorial no es muy extenso, debido al ajustado presupuesto, se mostrará una solución económica con resultados veraces la cual puede ser expandida en un futuro.

También se analizará si se han cumplido los objetivos propuestos al inicio, así como los comentarios pertinentes.

Y por último se hablará sobre el trabajo futuro que se puede realizar, tomando la base desarrollada en este proyecto, para mejorar las funcionalidades que posee el sistema.

7.1. Resultados.

7.1.1. Rango de cobertura.

El rango es muy dependiente de la situación de los transceptores que tienen mucho más alcance cuando están en la línea de visión, al aire libre que en interior, con obstáculos como paredes y otros materiales. La distancia normal que indica el fabricante para el módulo utilizado es de 50 metros, pero obviamente este dato es para el caso más óptimo.

En el caso de este proyecto, tras llevar a cabo una serie de pruebas, se comprueba que el alcance sin obstáculos se sitúa en torno a los 40-45 metros, pero cuando se interponen elementos el alcance baja hasta los 15-20 metros. Aun así, este alcance es más que suficiente para el correcto funcionamiento del sistema.

Sin embargo, para mejorar el rango, existen módulos en los que se añaden amplificadores de potencia al transmisor y preamplificadores al receptor para conseguir distancias más largas, que pueden llegar hasta 1 km. Estos módulos utilizan una antena externa que puede ser una antena simple que esté directamente conectada o un cable conectado a una antena con más ganancia.

Figura 7.1. Módulo nRF24l01 con amplificador.

En la figura 7.1 se muestra la versión con amplificador de potencia de transmisión y preamplificador para recepción. Los pines se conectan con Arduino de la misma forma en todas las versiones de transceptor y se utiliza el mismo software.

Por tanto, una solución sencilla para aumentar el rango sería la implementación de uno de estos módulos con amplificador en el nodo principal, con los demás nodos usando el módulo normal.

7.1.2. Fiabilidad frente a errores.

En este apartado se intentará demostrar el correcto funcionamiento del sistema, provocando fallos intencionadamente, similares a los que podrían ocurrir realmente.

Sensor defectuoso/desconectado

Este error se producirá cuando un nodo secundario se encuentre correctamente alimentado, pero el sensor que se conecta a él tiene algún problema, como por ejemplo, de alimentación.

Este problema podría provocar que el nodo principal guardara datos erróneos o irrelevantes, algo que podría comprometer el análisis posterior de los datos.

La solución consiste en que cada nodo compruebe todos los datos que envía, comprobando que se encuentre dentro del rango lógico, si sobrepasa el rango se establecerán los valores como 0, y estos valores serán fácilmente reconocibles al analizar los datos.

Módulo RF defectuoso/desconectado

Este error ocurre cuando hay algún problema de sincronización, o también cuando el módulo de radiofrecuencia no funciona adecuadamente.

Para solucionar el problema de la sincronización, tras llevar a cabo varias pruebas, se ha llegado a la conclusión de que para evitarlo se debe reintentar la comunicación una serie de veces. En concreto, se solicitará al nodo correspondiente que envíe los datos 1 vez, si esta falla, se repetirá el proceso hasta 5 veces. Lo normal será que con 2 intentos baste, aun así se dará un margen mayor.

Si tras 5 intentos de conexión se sigue produciendo error, se escribirá en un fichero de texto el nodo que ha fallado para tener un control de los errores.

Perdida de conexión WiFi

Este problema puede deberse a dos motivos, el primero sería que el Arduino Yún se desconfigure, el segundo sería que no exista la red WiFi almacenada por Arduino Yún.

Esta desconexión de Internet provocaría la imposibilidad de subir los datos a ThingSpeak, pero los datos seguirían almacenándose en la tarjeta microSD.

La solución a este problema consiste en reconfigurar la configuración inalámbrica del Yún tal y como se especifica en el ANEJO I.

Tarjeta microSD defectuosa/extraída

En caso de que la tarjeta microSD sea extraída mientras Arduino Yún está en funcionamiento, esto provocará que los datos tomados durante el periodo de extracción no puedan almacenarse. Pero si se subirán a ThingSpeak (si hay conexión a Internet).

7.1.3. Prueba de funcionamiento.

En este apartado se realizará un análisis de los datos obtenidos por el sistema durante un período de tiempo.

Teniendo en cuenta los datos obtenidos del sensor SHT10 del nodo 2, las gráficas que se obtienen tras un día de uso cotidiano son:

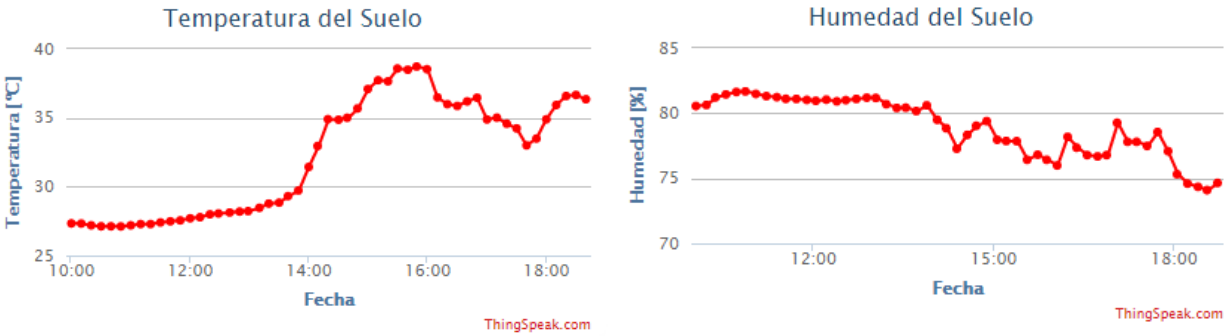


Figura 7.2. Gráficas Temperatura-Humedad del Suelo

Los resultados son muy buenos pues son de un día nublado, pero se muestran variaciones muy bruscas en algunos puntos, esto se debe a que en el sistema actual las medidas son tomadas 1 sola vez.

Teniendo en cuenta que la precisión varia en ±4.5% para la humedad y ±0.5ºC para la temperatura, para solucionar este error se ha modificado ligeramente el código para que se tomen 5 medidas y se haga una media de éstas. Así evitaremos saltos tan bruscos como los que aparecen en las gráficas anteriores.

Esta solución es aplicable a todos los nodos, que leerán 5 veces los datos de sus sensores para obtener datos más fiables.

Los resultados obtenidos son totalmente verídicos y fiables, con una gran robustez frente a errores, durante el período de tiempo simulado no se producido ningún error ni pérdida de datos.

7.2. Conclusiones.

Este trabajo supone el esfuerzo y dedicación de varios meses, donde la búsqueda de un método de comunicación inalámbrica fiable, robusto y barato ha sido la parte más compleja. Lograr que los nodos secundarios se comuniquen adecuadamente con el nodo receptor ha sido la gran prioridad del trabajo.

El objetivo principal de este proyecto era el desarrollo de un sistema electrónico de bajo coste que permitiera la monitorización de variables ambientales en huertos urbanos, esta parte se ha desarrollado mediante el uso de Arduinos Uno y una serie de sensores descritos anteriormente.

Por otro lado, se propusieron otros objetivos tales como:

- Almacenamiento en microSD de los datos con la hora precisa.
- Posibilidad de acceso remoto vía WiFi
- Representación de los datos en una plataforma IoT

Estas funcionalidades son las que realiza Arduino Yún, el microcontrolador que se ha seleccionado como parte del nodo principal. Por un lado, Yún es el receptor de los datos que leen los demás nodos distribuidos por el sistema y los guarda en la tarjeta microSD junto con la hora leída de una RTC. Por otro lado, Yún es el encargado de establecer las comunicaciones con Internet, tanto para subir los datos a una plataforma IoT como para permitir el acceso remoto a su tarjeta microSD para guardar los datos.

Además una funcionalidad que se ha implementado en el sistema, ha sido el control de la alimentación de los nodos secundarios por parte del nodo central mediante el uso de un relé.

Por tanto, se puede concluir que se ha realizado un trabajo acorde a lo esperado, que pone las bases para la realización de un sistema más completo en el futuro y que funciona correctamente

7.3. Trabajos futuros.

Una de las ventajas de realizar el proyecto basado en Arduino es la gran escalabilidad que posee el sistema, pues permite la incorporación de nuevos elementos realizando pequeñas variaciones en código.

Algunos de los proyectos que pueden llevarse a cabo con el sistema desarrollado son los siguientes:

- Añadir placas solares para alimentar el sistema.
- Creación de una página web (propia) que muestre las gráficas del sistema.
- Creación de una PCB eliminando los elementos no necesarios de Arduino.
- Añadir actuadores para tener control absoluto de forma remota.
- Añadir más sensores.
- Etc.

ANEJO I.

Configuración de Arduino Yún

I.1. Configuración inicial.

La primera vez que se inicia Arduino Yún se deberán seguir una serie de pasos para su correcta configuración

Al conectar el Arduino Yún a la alimentación, éste creará una red inalámbrica Ad-Hoc con un nombre aleatorio del tipo "ArduinoYun-XXXXXXXXX" a la que nos conectaremos desde cualquier dispositivo WiFi.

Tras esto, debemos acceder, a través del navegador, a la puerta de enlace predeterminada "192.168.240.1" o bien utilizar "arduino.local". Tras introducir esta ip, nos encontraremos con una página de login LuCi, como se muestra en la siguiente captura:

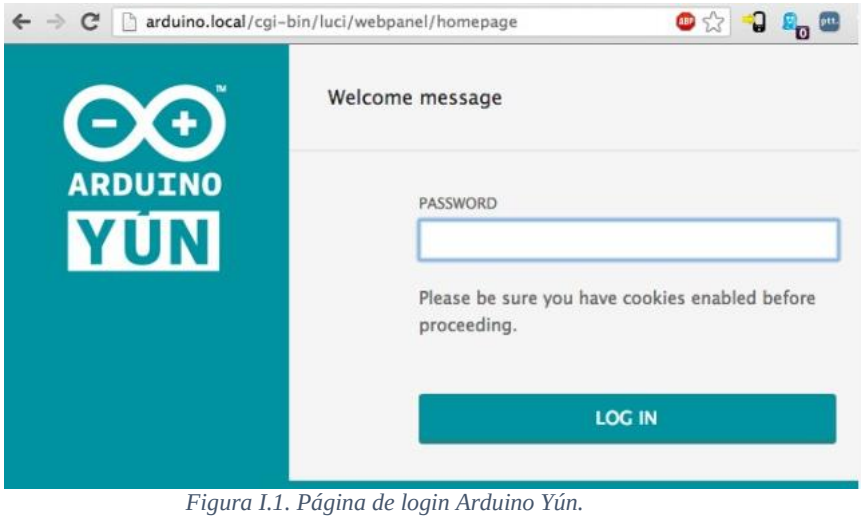


Figura I.1. Página de login Arduino Yún.

La clave por defecto es: "arduino".

Tras introducirla nos encontraremos con la información relacionada con las interfaces y con un botón de configuración:

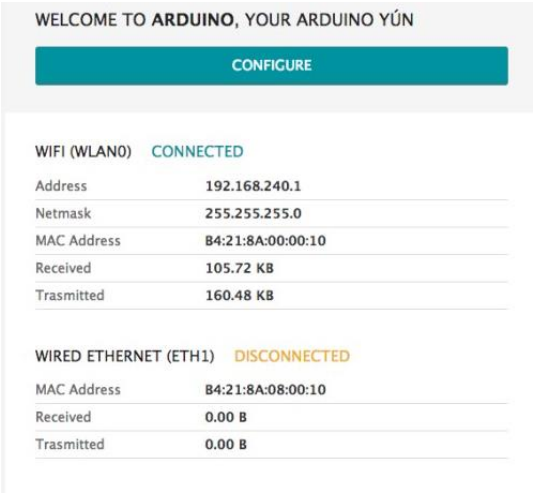


Figura I.2. Información Arduino Yún

Cuando presionemos sobre el botón “CONFIGURE”, nos aparecerá otra página donde podremos cambiar el nombre de la placa (por defecto, arduino), la contraseña, la zona horaria y lo más importante, la red WiFi a la que conectaremos la placa.

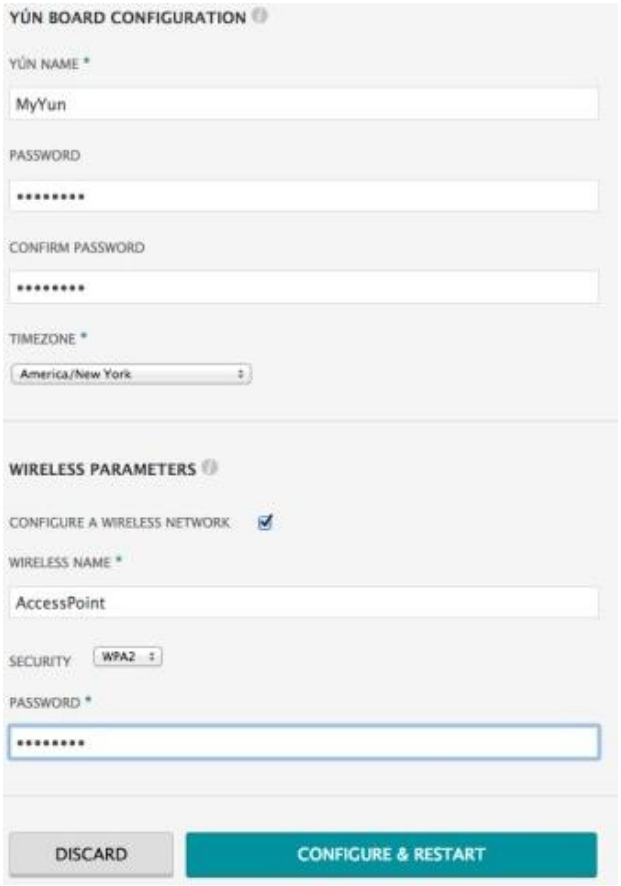


Figura I.3. Configuración Arduino Yún

En el apartado “API Rest” seleccionaremos “with password”, ya que en nuestro proyecto no hemos usado este protocolo.

Por último, se debe presionar el botón “Configure & Restart” para que los cambios se apliquen.

I.2. Cambiar configuración.

Si tenemos un Arduino Yún configurado, pero queremos cambiar, por ejemplo, la red WiFi a la que se conectará, tenemos dos formas de realizarlo.

La primera consiste en proceder de manera similar a la anterior, pero esta vez debemos estar conectados a la misma red que Arduino Yún teclear en el navegado la puerta de enlace a Yún, ésta la podremos introducir de forma numérica (tendremos que acceder a nuestro Router para ver la IP) o bien podremos acceder vía host introduciendo: “Nombre_arduino”.local, por defecto el nombre de arduino es, valga la redundancia, “arduino”, por tanto sería arduino.local.

Tras esto nos encontraremos con la página de login indicada anteriormente, y podremos acceder a la configuración.

La segunda forma consiste en resetear la configuración de Arduino Yún, para ello la placa posee un botón específico el cual debemos presionar durante 5 segundos:

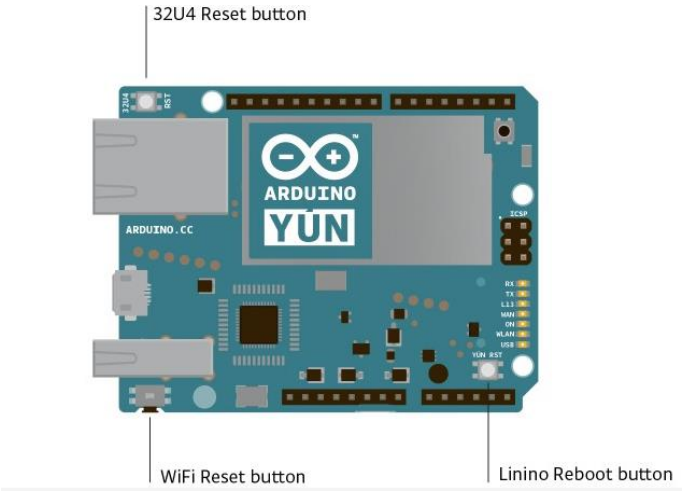


Figura I.4. Botón Reset WiFi Arduino Yún

Al presionar este botón deberemos reconfigurar el Arduino Yún siguiendo los pasos descritos en el punto I.1.

ANEJO II.

Códigos del Sistema

II.1. Nodo 1 – Nodo central.

```
#include <SPI.h>
#include <Wire.h>
#include <Console.h>
#include <Process.h>
#include <FileIO.h>
#include <nRF24L01.h>
#include <RF24.h>
#define DS1307_I2C_ADDRESS 0x68  //  Dirección I2C para el RTC
#define rele 13
#define alimentacion 6
#define gnd 5

//
// Variables globales
//

String FECHA_ACTUAL;
byte seg, min, hora, diaSemana, diaMes, mes, ano;
int t, intentos;

// Esta variable "error" contendrá los errores de todos los nodos.
int error[5] = {1, 1, 1, 1, 1};


struct N_SHT {
  float temp_suelo;
  float hum_suelo;
  float temp_nodo;
} N2;

struct N_DHT22 {
```

```
  float temp_amb;
  float hum_amb;
} N3;

struct N_RADIACION {
  float total;
  float foto;
  float temp_nodo;
} N4;



//
// Configuración Hardware
//

// Configuración del transceptor nRF24L01

RF24 radio(9, 10);

//
// Topologia
//


// Direcciones necesarias para la comunicación "ping-pong" o "Emisor-Receptor"
// Cada nodo secundario necesita una vía de comunicación independiente tanto para escribir como para
escuchar.
// El nodo MAESTRO lee cada una de estas direcciones.

const uint64_t talking_pipes[5] = { 0xF0F0F0F0D2LL, 0xF0F0F0F0C3LL, 0xF0F0F0F0B4LL, 0xF0F0F0F0A5LL,
0xF0F0F0F096LL };
const uint64_t listening_pipes[5] = { 0x3A3A3A3AD2LL, 0x3A3A3A3AC3LL, 0x3A3A3A3AB4LL,
0x3A3A3A3AA5LL, 0x3A3A3A3A96LL };


void setup(void)
{

  // Configuración de los pines digitales
  pinMode(alimentacion, OUTPUT);
  pinMode(gnd, OUTPUT);
  pinMode(rele, OUTPUT);
  digitalWrite(gnd, LOW);
  digitalWrite(rele, LOW);
  digitalWrite(alimentacion, HIGH);

  // Inicialización de procesos
  Wire.begin();
  Bridge.begin();
  Console.begin();
  FileSystem.begin();
  Serial.begin(9600);


  //
  // Inicio de la comunicación RF
  //

  radio.begin();

  //
```

```
  // Apertura de vías de comunicación con los nodos.
  //

  radio.openReadingPipe(1, talking_pipes[0]);
  radio.openReadingPipe(2, talking_pipes[1]);
  radio.openReadingPipe(3, talking_pipes[2]);
  radio.openReadingPipe(4, talking_pipes[3]);
  radio.openReadingPipe(5, talking_pipes[4]);


  // Se comienza a escuchar.
  // NOTA: Si no se establece en modo escucha, se configurará automáticamente en modo escritura.

  radio.startListening();

}

void loop(void)
{
  inicializacion();

  uint8_t i;
  for (i = 2; i < 5; i++) {
    intentos=0;
    while (error[i - 1] == 1 && intentos < 5) {
      pedir_datos(i);
      leer_datos(i);
    }
    escribir_fichero(i);
  }

  procesoThingSpeak();

  // Se desactiva la alimentación de los nodos secundarios.
  digitalWrite(rele, LOW);

  // El nodo central espera durante un periodo de tiempo.

  for (t = 0; t < 473; t++) { // 10 minutos = 600 segundos, quitando los 120 segundos
    delay(1000);              // de espera (t=480s), y el tiempo medio del proceso... t=473s
  }

}
void inicializacion() {

  // En primer lugar se activa el rele y, por tanto, la alimentación de los nodos secundarios.
  digitalWrite(rele, HIGH);

  // Esperamos dos minutos para que a los nodos les de tiempo a leer los sensores.
  delay(30000);
  for (t = 0; t < 120; t++) {
    delay(1000);
    }

  // Se inicializan los errores a 1
  error[0] = 1;
  error[1] = 1;
  error[2] = 1;
  error[3] = 1;

  // Se ponen todas las variables a 0
  N2.hum_suelo = 0;
```

```
  N2.temp_suelo = 0;
  N2.temp_nodo = 0;

  N3.hum_amb = 0;
  N3.temp_amb = 0;


  N4.total = 0;
  N4.foto = 0;
  N4.temp_nodo = 0;

}
void pedir_datos(uint8_t nodo) {

  byte peticion = 1;

  // Para de escuchar, para poder enviar...
  radio.stopListening();

  // Se abre una vía de comunicación con el nodo correspondiente
  radio.openWritingPipe(listening_pipes[nodo - 2]);
  bool done = false;

  done = radio.write( &peticion, sizeof(byte) );

  Serial.print("Datos pedidos al nodo ");
  Serial.println(nodo);

}

void leer_datos(uint8_t nodo) {
  byte cnt = 0;
  radio.startListening();

  // Se espera a que la radio este disponible durante 10seg
  while (!radio.available() && cnt <= 5)
  {
    cnt++;
    delay(1000);
  }

  // Si está disponible en menos de 10seg se lee
  if (cnt < 5) {
    bool done = false;
    while (!done) {

      if (nodo == 2) {
        done = radio.read( &N2, sizeof(struct N_SHT) );
      }
      else if (nodo == 3) {
        done = radio.read( &N3, sizeof(struct N_DHT22) );
      }
      else if (nodo == 4) {
        done = radio.read( &N4, sizeof(struct N_RADIACION) );
      }
    }

    error[nodo - 1] = 0;

  }
```

```
  // Si no está disponible se marca como erróneo
  else {
    error[nodo - 1] = 1;
  }
  intentos++;
}

void escribir_fichero(uint8_t nodo) {

  Serial.println("-----------------------------------------");

  if (error[nodo - 1] == 0) {

    switch (nodo) {

      case 2: {
          Serial.println();
          Serial.print("Temperatura (Suelo):  ");
          Serial.print(N2.temp_suelo);
          Serial.print(" || Humedad (Suelo):  ");
          Serial.print(N2.hum_suelo);

          // La temperatura del nodo, se muestra por pantalla y se sube a ThingSpeak
          // pero no se pone en el fichero.

          Serial.print(" || Temperatura (Nodo):  ");
          Serial.println(N2.temp_nodo);

          String dataString;

          // Se captura la fecha
          FECHA_ACTUAL = date();
          dataString += FECHA_ACTUAL;
          dataString += " = ";
          dataString += String(N2.temp_suelo);
          dataString += ",";
          dataString += String(N2.hum_suelo);
          // Se abre el fichero. Solo puede estar abierto 1 fichero a la vez.
          File dataFile = FileSystem.open("/mnt/sd/arduino/www/NODO_2_SHT.txt", FILE_APPEND);
          // Si el fichero está disponible, se escribe...
          if (dataFile) {
            dataFile.println(dataString);
            dataFile.close();
            // Se imprime por puerto serie.
            Serial.println(dataString);
            Serial.println();
            Serial.println();

          }
          // Si el fichero no se abre...
          else {
            Serial.println("Error al abrir el fichero NODO_2_SHT.txt");
          }
          return;
        }

      case 3: {
          Serial.println();
          Serial.print("Temperatura(Ambiente):  ");
          Serial.print(N3.temp_amb);
          Serial.print(" || Humedad (Ambiente):  ");
          Serial.println(N3.hum_amb);
```

```
    String dataString;
    dataString += FECHA_ACTUAL;
    dataString += " = ";
    dataString += String(N3.temp_amb);
    dataString += ",";
    dataString += String(N3.hum_amb);
    // Abrimos el fichero. Solo puede estar abierto 1 fichero a la vez.
    File dataFile = FileSystem.open("/mnt/sd/arduino/www/NODO_3_DHT.txt", FILE_APPEND);
    // Si el fichero está disponible, se escribe...
    if (dataFile) {
      dataFile.println(dataString);
      dataFile.close();
      // Se imprime por puerto serie
      Serial.println(dataString);
      Serial.println();
      Serial.println();
    }
    // Si el fichero no se abre...
    else {
      Serial.println("Error al abrir el fichero NODO_3_DHT.txt");
    }
    return;
  }

case 4: {
    Serial.println();
    Serial.print("Radiacion Total:  ");
    Serial.print(N4.total);
    Serial.print(" || Radiacion Fotosintesis:  ");
    Serial.println(N4.foto);

    // Esta temperatura si interesa guardarla en fichero.
    Serial.print(" || Temperatura (Nodo):  ");
    Serial.println(N4.temp_nodo);

    String dataString;
    dataString += FECHA_ACTUAL;
    dataString += " = ";
    dataString += String(N4.total);
    dataString += ",";
    dataString += String(N4.foto);
    dataString += ",";
    dataString += String(N4.temp_nodo);

    // Abrimos el fichero. Solo puede estar abierto 1 fichero a la vez.
    File dataFile = FileSystem.open("/mnt/sd/arduino/www/NODO_4_RADIACION.txt", FILE_APPEND);
    // Si el fichero está disponible, se escribe...
    if (dataFile) {
      dataFile.println(dataString);
      dataFile.close();
      // Se imprime por puerto serie
      Serial.println(dataString);
      Serial.println();
      Serial.println();
    }
    // Si el fichero no se abre...
    else {
      Serial.println("Error al abrir el fichero NODO_4_RADIACION.txt");
    }
```

```
          return;
        }
      }
    }

    else {

      String dataString;
      dataString += FECHA_ACTUAL;
      dataString += " - Error en el nodo ";
      dataString += String(nodo);

      // Abrimos el fichero. Solo puede estar abierto 1 fichero a la vez.
      File dataFile = FileSystem.open("/mnt/sd/arduino/www/errores.txt", FILE_APPEND);
      // Si el fichero está disponible, se escribe...
      if (dataFile) {
        dataFile.println(dataString);
        dataFile.close();
        // Se imprime por puerto serie
        Serial.println(dataString);
        Serial.println();
      }
      // Si el fichero no se abre...
      else {
        Serial.println("Error al abrir el fichero errores.txt");
      }

    }
}

////-----RTC-------////

// -- Conversor BCD ==> DECIMAL --
byte bcdToDec(byte val)
{
  return ( (val / 16 * 10) + (val % 16) );
}

// -- Imprimir en formato 2 digitos --
String fprint (int dato)
{
  String retorno = String(dato);
  if (retorno.length() < 2)
    retorno = "0" + retorno;
  return retorno;
}

String date() {

  // -- Proceso de lectura --
  Wire.beginTransmission(DS1307_I2C_ADDRESS); // Abre I2C en modo escritura
  Wire.write((byte)0x00);
  Wire.endTransmission();                     // Se termina la escritura
  Wire.requestFrom(DS1307_I2C_ADDRESS, 7);    // Abre I2C en modo lectura
  seg     = bcdToDec(Wire.read() & 0x7f);     // Se leen 7 bytes de datos
  min     = bcdToDec(Wire.read());
  hora       = bcdToDec(Wire.read() & 0x3f);
  diaSemana  = bcdToDec(Wire.read());   //No se necesita pero se lee también
  diaMes = bcdToDec(Wire.read());
  mes      = bcdToDec(Wire.read());
  ano      = bcdToDec(Wire.read());
```

```
  String FECHA;
  FECHA += fprint(diaMes);
  FECHA += "/";
  FECHA += fprint(mes);
  FECHA += "/";
  FECHA += fprint(ano);
  FECHA += " - ";
  FECHA += fprint(hora);
  FECHA += ":";
  FECHA += fprint(min);
  FECHA += ":";
  FECHA += fprint(seg);

  return FECHA;
}



void procesoThingSpeak ()
{

  String key = "MVVIE53JKE4AYH1Y"; // Esta key se obtiene de ThingSpeak

  Process p;
  String cmd = "curl --data \"key=" + key;

  if (error[1] == 0) {

    cmd = cmd + "&field1" +  + "=" + N2.temp_suelo;
    cmd = cmd + "&field2" +  + "=" + N2.hum_suelo;
    cmd = cmd + "&field3" +  + "=" + N2.temp_nodo;
  }

  if (error[2] == 0) {

    cmd = cmd + "&field4" +  + "=" + N3.temp_amb;
    cmd = cmd + "&field5" +  + "=" + N3.hum_amb;
  }

  if (error[3] == 0) {

    cmd = cmd + "&field6" +  + "=" + N4.foto;
    cmd = cmd + "&field7" +  + "=" + N4.total;
    cmd = cmd + "&field8" +  + "=" + N4.temp_nodo;


  }

  cmd = cmd + "\" http://api.thingspeak.com/update";
  p.runShellCommand(cmd);
  Console.println(cmd);
  p.close();

}
```

II.2. Nodo 2 – Nodo sensor SHT10.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <SHT1x.h>
#define alimentacion 2
#define NODO 2

//
// Configuración de Sensores
//

// Pines necesarios para el sensor SHT10
#define dataPin  5 //AZUL
#define clockPin 6 //AMARILLO

SHT1x sht1x(dataPin, clockPin);


// Pin (analógico) necesario para el LM35
#define LM35 A0

struct N_SHT {
  float temp_suelo;
  float hum_suelo;
  float temp_nodo;
} N2;

// Variables detección errores
boolean error_medida1 = false;
boolean error_medida2 = false;
boolean error_medida3 = false;

//
// Configuración RF
//

// Configuración del transceptor nRF24L01

RF24 radio(9, 10);

//
// Topologia
//

// Direcciones necesarias para la comunicación "ping-pong" o "Emisor-Receptor"
// Cada nodo secundario necesita una vía de comunicación independiente tanto para escribir como
para escuchar.
// El nodo MAESTRO lee cada una de estas direcciones.

const uint64_t talking_pipes[5] = { 0xF0F0F0F0D2LL, 0xF0F0F0F0C3LL, 0xF0F0F0F0B4LL,
0xF0F0F0F0A5LL, 0xF0F0F0F096LL };
const uint64_t listening_pipes[5] = { 0x3A3A3A3AD2LL, 0x3A3A3A3AC3LL, 0x3A3A3A3AB4LL,
0x3A3A3A3AA5LL, 0x3A3A3A3A96LL };



// Se crea una variable donde se guardará el valor de la dirección correspondiente

uint8_t direccion_nodo = NODO;
```

```
void setup(void)
{
  Serial.begin(9600);
  pinMode(alimentacion, OUTPUT);
  digitalWrite(alimentacion, HIGH);

  //
  // Inicio de la comunicación RF
  //

  radio.begin();

  //
  // Apertura de vías de comunicación con el nodo MAESTRO
  //

  // Se escribe y se lee en su dirección correspondiente.
  radio.openWritingPipe(talking_pipes[direccion_nodo - 2]);
  radio.openReadingPipe(1, listening_pipes[direccion_nodo - 2]);

}

void loop(void)
{
  N2.temp_suelo = 0;
  N2.hum_suelo = 0;
  N2.temp_nodo = 0;

  int i;
  // Primero se leen los datos del sensor (5 veces)
  for (i = 0; i < 5; i++) {
    leer();
    delay(2000);
  }

  // Segundo, se comprueban los datos
  comprobar();

  // Tercero, se escucha al nodo principal
  escuchar();

  // Por último, se envían los datos
  envio_datos();
}

void leer() {
  int lectura_lm35;

  N2.hum_suelo = N2.hum_suelo + sht1x.readHumidity();
  N2.temp_suelo = N2.temp_suelo + sht1x.readTemperatureC();

  lectura_lm35 = analogRead(LM35);
  lectura_lm35 = (5.0 * lectura_lm35 * 100.0) / 1024.0;
  N2.temp_nodo = N2.temp_nodo + lectura_lm35;
}

void comprobar() {

  // Se divide entre 5 porque se hace la media de las 5 medidas realizadas
  N2.hum_suelo = N2.hum_suelo / 5.0;
  N2.temp_suelo = N2.temp_suelo / 5.0;
  N2.temp_nodo = N2.temp_nodo / 5.0;
```

```
  // Se verifica si los datos son correctos
  error_medida1 = N2.temp_suelo > 120 && N2.temp_suelo < -40  ? true : false;
  N2.temp_suelo = error_medida1 == true ? 0 : N2.temp_suelo;

  error_medida2 = N2.hum_suelo > 100 && N2.hum_suelo < 0  ? true : false;
  N2.hum_suelo = error_medida2 == true ? 0 : N2.hum_suelo;

  error_medida3 = N2.temp_nodo > 150 && N2.temp_nodo < -55  ? true : false;
  N2.temp_nodo = error_medida3 == true ? 0 : N2.temp_nodo;
}
void escuchar() {

  byte peticion;
  bool done = false;

  //
  // Comenzamos a escuchar
  //

  radio.startListening();

  // Tras la lectura, se espera a que llegue una petición de datos del maestro.
  while (!radio.available()) {
    delay(100);
  }

  while (!done)
  {
    // Se lee hasta que se complete la transferencia.
    done = radio.read( &peticion, sizeof(byte) );
  }
  if (peticion == 1) {

    return;

  }

  else {
    //Si no se trata de la petición esperada, se vuelve a escuchar...
    escuchar();
  }

}
void envio_datos() {

  radio.stopListening();

  Serial.print("Humedad del Suelo:  ");
  Serial.print(N2.hum_suelo);

  Serial.print("Temperatura del Suelo:  ");
  Serial.print(N2.temp_suelo);

  Serial.print("Temperatura del Nodo:  ");
  Serial.println(N2.temp_nodo);

  bool done = false;
  while (!done) {
    done = radio.write( &N2, sizeof(struct N_SHT) );
  }
}
```

II.3. Nodo 3 – Nodo sensor DHT22.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <DHT.h>
#define alimentacion 2
#define NODO 3

//
// Configuración de Sensores
//

#define DHTPIN 5
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);


struct N_DHT22 {
  float temp_amb;
  float hum_amb;
} N3;

// Variables detección errores
boolean error_medida1 = false;
boolean error_medida2 = false;

//
// Configuración RF
//

// Configuración del transceptor nRF24L01

RF24 radio(9, 10);

//
// Topologia
//

// Direcciones necesarias para la comunicación "ping-pong" o "Emisor-Receptor"
// Cada nodo secundario necesita una vía de comunicación independiente tanto para escribir como
para escuchar.
// El nodo MAESTRO lee cada una de estas direcciones.

const uint64_t talking_pipes[5] = { 0xF0F0F0F0D2LL, 0xF0F0F0F0C3LL, 0xF0F0F0F0B4LL,
0xF0F0F0F0A5LL, 0xF0F0F0F096LL };
const uint64_t listening_pipes[5] = { 0x3A3A3A3AD2LL, 0x3A3A3A3AC3LL, 0x3A3A3A3AB4LL,
0x3A3A3A3AA5LL, 0x3A3A3A3A96LL };



// Se crea una variable donde se guardará el valor de la dirección correspondiente

uint8_t direccion_nodo = NODO;

void setup(void)
{
  Serial.begin(9600);

  pinMode(alimentacion, OUTPUT);
  digitalWrite(alimentacion, HIGH);
```

```
  //
  // Inicialización del sensor DHT22
  //
  dht.begin();

  //
  // Inicio de la comunicación RF
  //

  radio.begin();

  //
  // Se abren vías de comunicación con el nodo MAESTRO
  //

  // Se escribe y se lee en su dirección correspondiente.
  radio.openWritingPipe(talking_pipes[direccion_nodo - 2]);
  radio.openReadingPipe(1, listening_pipes[direccion_nodo - 2]);

}

void loop(void)
{
  int i;
  N3.temp_amb = 0;
  N3.hum_amb = 0;

  // Primero se leen los datos del sensor (5 veces)
  for (i = 0; i < 5; i++) {
    leer();
    delay(2000);
  }

  // Segundo, se comprueban los datos
  comprobar();

  // Tercero, se escucha al nodo principal
  escuchar();

  //Por último, se envían los datos al maestro
  envio_datos();

}


void leer() {

  N3.temp_amb = N3.temp_amb + dht.readTemperature();
  N3.hum_amb = N3.hum_amb + dht.readHumidity();
  if (isnan(N3.temp_amb) || isnan(N3.hum_amb)) {
    N3.temp_amb = 0;
    N3.hum_amb = 0;
    leer();
  }
}

void comprobar() {

  //Se divide entre 5 porque se hace la media de las 5 medidas realizadas
  N3.temp_amb = N3.temp_amb / 5.0;
  N3.hum_amb = N3.hum_amb / 5.0;
```

```
  error_medida1 = N3.temp_amb > 120 && N3.temp_amb < -40  ? true : false;
  N3.temp_amb = error_medida1 == true ? 0 : N3.temp_amb;

  error_medida2 = N3.hum_amb  > 100 && N3.hum_amb  < 0  ? true : false;
  N3.hum_amb  = error_medida2 == true ? 0 : N3.hum_amb ;

}
void escuchar() {


  byte peticion;

  bool done = false;

  //
  // Comenzamos a escuchar
  //

  radio.startListening();

  //Tras la lectura, esperamos que llegue una petición de datos del maestro.
  while (!radio.available()) {
    delay(100);
  }

  while (!done)
  {
    // Leemos hasta que se complete la transferencia.
    done = radio.read( &peticion, sizeof(byte) );
  }
  if (peticion == 1) {

    return;

  }

  else {

    //Si no se trata de la petición esperada, se vuelve a escuchar...
    escuchar();
  }
}


void envio_datos() {

  radio.stopListening();

  Serial.print("Humedad:  ");
  Serial.print(N3.hum_amb);

  Serial.print(" Temperatura ambiente:  ");
  Serial.print(N3.temp_amb);

  bool done = false;
  while (!done) {
    done = radio.write( &N3, sizeof(struct N_DHT22) );
  }
}
```

II.4. Nodo 4 – Nodo sensores radiación.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

#define alimentacion 2
#define NODO 4

//
// Configuración de Sensores
//

// Pin sondas S1087 (fotosíntesis) y S1087-01(total)
#define TOTAL A0
#define FOTO A1

// Pin (analógico) necesario para el LM35
#define LM35 A2

struct N_RADIACION {
  float total;
  float foto;
  float temp_nodo;
} N4;

// Variables detección errores
boolean error_medida1 = false;
boolean error_medida2 = false;
boolean error_medida3 = false;

//
// Configuración RF
//

// Configuración del transceptor nRF24L01

RF24 radio(9, 10);

//
// Topologia
//

// Direcciones necesarias para la comunicación "ping-pong" o "Emisor-Receptor"
// Cada nodo secundario necesita una vía de comunicación independiente tanto para escribir como
para escuchar.
// El nodo MAESTRO lee cada una de estas direcciones.

const uint64_t talking_pipes[5] = { 0xF0F0F0F0D2LL, 0xF0F0F0F0C3LL, 0xF0F0F0F0B4LL,
0xF0F0F0F0A5LL, 0xF0F0F0F096LL };
const uint64_t listening_pipes[5] = { 0x3A3A3A3AD2LL, 0x3A3A3A3AC3LL, 0x3A3A3A3AB4LL,
0x3A3A3A3AA5LL, 0x3A3A3A3A96LL };



// Se crea una variable donde se guardará el valor de la dirección correspondiente

uint8_t direccion_nodo = NODO;
```

```
void setup(void)
{
  Serial.begin(9600);
  pinMode(alimentacion, OUTPUT);
  digitalWrite(alimentacion, HIGH);


  //
  // Inicio de la comunicación RF
  //

  radio.begin();

  //
  // Se abren vías de comuniación con el nodo MAESTRO
  //

  // Se escribe y se lee en su dirección correpondiente.
  radio.openWritingPipe(talking_pipes[direccion_nodo - 2]);
  radio.openReadingPipe(1, listening_pipes[direccion_nodo - 2]);

}

void loop(void)
{

  N4.total = 0;
  N4.foto = 0;
  N4.temp_nodo = 0;

   int i;

  // Primero se leen los datos del sensor (5 veces)
  for (i = 0; i < 5; i++) {
    leer();
    delay(2000);
  }

  // Se divide entre 5 porque se hace la media de las 5 medidas realizadas
  N4.total = N4.total / 5.0;
  N4.foto = N4.foto / 5.0;
  N4.temp_nodo = N4.temp_nodo / 5.0;

  // Se escucha al nodo principal
  escuchar();

  // Por último, se envían los datos al maestro
  envio_datos();

}


void leer() {
  float Vsensor, I;

  Vsensor = analogRead(TOTAL);
  Vsensor = (Vsensor / 1024) * 5;
  I = Vsensor / 100000;
  N4.total = N4.total + 0.769 * (10 ^ 5) * I * 1000;

  Vsensor = analogRead(FOTO);
  Vsensor = (Vsensor / 1024) * 5;
```

```
  I = Vsensor / 100000;
  N4.foto = N4.foto + 0.625 * (10 ^ 6) * I * 1000;


  Vsensor = analogRead(LM35);
  Vsensor = (5.0 * Vsensor * 100.0) / 1024.0;
  N4.temp_nodo = N4.temp_nodo + Vsensor;

}

void escuchar() {

  byte peticion;

  bool done = false;

  //
  // Comenzamos a escuchar
  //

  radio.startListening();

  // Tras la lectura, esperamos que llegue una petición de datos del maestro.
  while (!radio.available()) {
    delay(100);
  }

  while (!done)
  {
    // Se lee hasta que se complete la transferencia.
    done = radio.read( &peticion, sizeof(byte) );
  }
  if (peticion == 1) {

    return;

  }

  else {

    // Si no se trata de la petición esperada, se vuelve a escuchar...
    escuchar();
  }
}

void envio_datos() {

  radio.stopListening();

  Serial.print("Luminancia Total  ");
  Serial.print(N4.total);

  Serial.print(" Luminancia Fotosíntesis:  ");
  Serial.print(N4.foto);

  Serial.print(" Temperatura nodo:  ");
  Serial.println(N4.temp_nodo);

  bool done = false;
  while (!done) {
    done = radio.write( &N4, sizeof(struct N_RADIACION) );
  }
```

```
}
```

Bibliografía

[1] D. Evans, «Internet de las cosas: Cómo la próxima evolución de Internet lo cambia todo,» Cisco, 2011.

[2] Libelium, «Agriculture 2.0 Technical Guide,» 2014.

[3] Cooking-Hacks. [En línea]. Disponible en: http://www.cooking-hacks.com/documentation/tutorials/waspmote#waspmote_vs_arduino.

[4] WIDHOC. [En línea]. Disponible en: http://widhoc.com/.

[5] Arduino, «Arduino Uno,» [En línea]. Disponible en: http://arduino.cc/en/Main/arduinoBoardUno.

[6] Arduino, «Arduino Mega 2560,» [En línea]. Disponible en: http://arduino.cc/en/Main/ArduinoBoardMega2560.

[7] Arduino, «Arduino Nano,» [En línea]. Disponible en: http://arduino.cc/en/Main/ArduinoBoardNano.

[8] Arduino, «Arduino Mini,» [En línea]. Disponible en: http://arduino.cc/en/Main/ArduinoBoardMini.

[9] Arduino, «Arduino Leonardo,» [En línea]. Disponible en: http://arduino.cc/en/Main/ArduinoBoardLeonardo.

[10] Arduino, «Arduino Yún,» [En línea]. Disponible en: http://arduino.cc/en/Main/ArduinoBoardYun.

[11] Didatec Tecnología Educativa, Introducción al Sistema PICAXE, Revolution Education, 2001.

[12] J. J. Monteverde, Introducción al Microcontrolador MSP430 TI, 2013.

[13] Hamamatsu, «advanticsys.com,» [En línea]. Disponible en: http://www.advanticsys.com/wiki/index.php?title=Hamamatsu%C2%AE_S1087_Series.

[14] Hamamatsu, «Si photodiodes. S1087/S1133 series,» 2014.

[15] LogMeIn, «Xively,» [En línea]. Disponible en: www.xively.com.