ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Prototipo de un sistema de monitorización domótico basado en la plataforma Arduino usando protocolos estándar de gestión de red

(A home monitorization system prototype based on the Arduino platform using standard network management protocols)

Para acceder al Título de

Graduado en
Ingeniería de Tecnologías de Telecomunicación

Autor: Sergio Santa Cruz Alario

Julio - 2023

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Serio Santa Cruz Alario
Director del TFG: José Ángel Irastorza Teja

Título: "Prototipo de un sistema de monitorización domótico basado en la plataforma Arduino usando protocolos estándar de gestión de red"
Title: "A home monitorization system prototype based on the Arduino platform using standard network management protocols "

Presentado a examen el día: 18 de julio de 2023

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre):
Secretario (Apellidos, Nombre):
Vocal (Apellidos, Nombre):

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

El Internet de las cosas ha experimentado un crecimiento notorio en los últimos años, impulsado por los avances tecnológicos y la creciente demanda de conectividad instantánea y ubicua. En la actualidad, nos encontramos inmersos en una realidad en la que el número de dispositivos conectados a la red ha experimentado un aumento exponencial. A principios de los años 2000, una vivienda tipo contaba con solo 2 o 3 dispositivos conectados a un punto de acceso. Sin embargo, en la actualidad, gracias a la proliferación de soluciones domóticas "do it yourself" controlables a través de Wi-Fi, nos encontramos con un escenario en el que es posible tener 30, 40 o incluso 50 dispositivos muy diversos conectados al mismo punto de acceso.

A pesar de que la tecnología lo permite y se promocionan estas soluciones como una opción rápida y sencilla para incorporar la domótica en nuestra vida diaria, es importante ser conscientes de que todo tiene sus límites. Si bien en el día a día tener 40 dispositivos conectados al enrutador puede no afectar significativamente el uso habitual de Internet, surgen algunas preguntas relevantes como: ¿qué sucede el día en que se considere necesario cambiar de compañía y se requiera reconfigurar todo el sistema? Frente a escenarios como este, ya hay empresas que ofrecen sistemas domóticos propios que buscan reducir la dependencia de dispositivos con acceso directo a Internet. Estas empresas implementan arquitecturas internas propietarias, con un módulo de conexión al enrutador. Sin embargo, estas soluciones aún son minoritarias, ya que no existe un estándar claro y definido para esta rama de dispositivos IoT.

A lo largo del plan de estudios del grado en Ingeniería de Tecnologías de Telecomunicación, se abordan de manera recurrente dos cuestiones fundamentales: los estándares y los dispositivos en red. Sin embargo, ¿existe algún punto de conexión entre ambos? La respuesta es claramente sí: el Protocolo Simple de Administración de Red (SNMP, por sus siglas en inglés). SNMP actúa como un vínculo vital que permite la gestión y supervisión de los dispositivos de red según los estándares establecidos. Partiendo de la gestión de dispositivos de red, es posible llevar a cabo una abstracción de estos elementos hacia componentes más pequeños: sensores y actuadores simples. Sin embargo, aún es posible ir un paso más allá, gestionando exclusivamente el controlador al que están conectados el resto de los dispositivos. Bajo un dispositivo central, se pueden conectar una multitud de elementos, todos con una dirección única hacia la red. Esto brinda la posibilidad de gestionar de manera individual y centralizada decenas de elementos distintos.

Una vez que se han establecido las principales condiciones de contorno que dan origen a este proyecto, se abordarán cuestiones preliminares relacionadas con la motivación y los objetivos subyacentes. Asimismo, se presentará la estructura que se seguirá durante el desarrollo del documento.

1.1 Objetivos y motivación

El objetivo principal de este proyecto es la aplicación directa de la arquitectura SNMP sobre placas de bajas prestaciones, como Arduino. El propósito es implementar un modelo de gestión que ha tenido una importancia histórica sobre un dispositivo basado en una plataforma muy limitada, pero que ha ganado una gran popularidad en la última década.

Partiendo sobre un hipotético caso de uso bajo el ámbito domótico, se elaborará una maqueta en la que se instalarán y probarán diferentes sensores y actuadores, los cuales serán monitorizables

vía SNMP desde un ordenador. Entrando algo más en detalle, las etapas que se irán abordando son:

- Análisis de los requisitos.
- Elección de los controladores.
- Elección de los módulos de comunicación.
- Elección de los sensores y actuadores.
- Escritura del código para el funcionamiento de los sensores y actuadores.
- Montaje de la arquitectura SNMP sobre uno de los controladores.
- Escritura del código para las comunicaciones.
- Pruebas de funcionamiento.
- Montaje de una maqueta que de visión de conjunto al sistema.

La tarea de ensamblar y hacer funcionar correctamente los diversos códigos involucrados en la implementación de este proyecto será un desafío arduo. Un proceso constante de prueba y error, donde será necesario adaptar planes, abandonar ideas y reinventarse a medida que se avance.

1.2 Estructura del documento

A lo largo de todo el documento, se encontrarán secciones claramente diferenciadas. El primer apartado se centrará en aspectos técnicos, donde se hará hincapié en la diversidad de dispositivos de control, sensores, actuadores y opciones de comunicación disponibles en el mercado. También se introducirán los protocolos que podrían ser soluciones adecuadas para gestionar un sistema en red.

En el segundo apartado, se desarrollan las bases teóricas del proyecto, profundizando en las necesidades que se pretenden cubrir, los dispositivos específicos que se plantean instalar y la forma en que todo el sistema debe funcionar en conjunto.

En el tercer apartado, se llevará a la práctica toda la teoría previamente expuesta, poniendo especial énfasis en los diferentes códigos desarrollados y en cómo estos son utilizados para crear los diversos sistemas dentro del paradigma de la domótica.

En el cuarto apartado, se proporcionará una visión general del conjunto, culminando con la finalización de la maqueta y la integración de todos los programas para lograr un sistema SNMP funcional. Se mostrará cómo todos los componentes individuales se unen para formar un sistema completo y operativo.

En el quinto y último apartado del trabajo, se realizarán las pruebas necesarias para verificar el correcto funcionamiento del sistema, evaluándolo desde diversas perspectivas. Algunas de estas pruebas se enfocarán en la interacción con un usuario promedio, mientras que otras se centrarán en los aspectos técnicos y la ingeniería que respaldan el proyecto.

Como añadido, en las últimas páginas, se mostrarán como anexos informaciones concretas de algunos dispositivos y algunos desarrollos extensos de código, de forma que sea posible analizar en mayor profundidad algunos temas en los que no se hará especial énfasis durante el desarrollo del proyecto.

2. ASPECTOS TÉCNICOS PREVIOS

Comenzando con una visión genérica del panorama tecnológico actual, es importante conocer el sector en que se moverá este proyecto. Existiendo multitud de vías para llegar al mismo destino, barajar las opciones disponibles, en base a los requisitos establecidos, es el primer paso para el éxito. Con un mercado tan extenso como es el de la electrónica en estas fechas, se realizará durante este capítulo un repaso superficial sobre conceptos, dispositivos y protocolos, que más adelante conformarán el resultado final del proyecto.

2.1 IoT / Domótica

El Internet de las cosas es un concepto que hace referencia a la interconexión de dispositivos y objetos a través de Internet. Dotar de inteligencia a estos elementos abre una puerta hacia nuevas rutas de flujo de información, pudiéndose utilizar desde para la recopilación de datos, hasta para la automatización de tareas. Ambos casos guardan estrecha relación con un término históricamente muy anterior al Internet, la domótica [1].

La domótica, o tecnologías del hogar digital/inteligente, es un conjunto de sistemas instalados sobre una vivienda para la automatización y control de diferentes sistemas o tareas mayores. En líneas generales, el objetivo principal de la domótica siempre han sido mejorar la calidad de vida de sus usuarios, desde ofrecer mayores comodidades hasta aspectos de seguridad en el hogar [2].

Con la fusión del Internet de las cosas y la domótica en las últimas décadas, cada vez es mayor la variedad de dispositivos que se pueden adquirir en el mercado bajo la etiqueta de “domótica Wi-Fi”, y es que trabajando en sinergia, son soluciones verdaderamente simples y con precios bastante asequibles.

Dejando un poco de lado toda esa domótica comercial, la base de la misma no son más que controladores y sensores o actuadores, por lo cual cualquiera con un poco de experiencia en el uso de estos podría desarrollar su propio sistema. En los siguientes apartados se desarrollarán algunos controladores y componentes con los que se podría montar un sistema a medida para el usuario [3].

2.2 Plataformas de desarrollo

Las plataformas de desarrollo son sistemas de prototipado que permiten a sus usuarios generar proyectos a través de código. Partiendo como una solución accesible para aquellos más interesados en el mundo de la electrónica, estas plataformas se manifiestan como las denominadas placas de prototipado. Una placa de prototipado, compuesta por un microcontrolador que ejerce como cerebro, una serie de pines de entrada y salida para el control de otros dispositivos, y una memoria para el almacenamiento de programas, se reduce a un pequeño dispositivo con infinidad de posibilidades. Existiendo numerosas marcas que compiten sobre el mismo nicho de mercado, en este apartado se mencionarán tres de las que más repercusión tienen hoy en día gracias al comercio online.

2.2.1 Arduino

Arduino es una plataforma de prototipado electrónico originada en Italia a principio de los años 2000 como medio para acercar la electrónica y la programación tanto a profesionales como a principiantes. Basadas en circuitos impresos, microcontroladores, y pines de entrada y salida, como se puede ver en la Fig. 2.1, las placas Arduino son una vía sencilla para el desarrollo de proyectos electrónicos, bajo un entorno de programación muy simplificado conocido como el IDE de Arduino, utilizando lenguaje C o C++. Durante el desarrollo de un proyecto, la placa ejerce como núcleo lógico, mediante la ejecución lineal del código cargado vía USB, pero el gran rango de aplicaciones con que cuenta esta tecnología se debe a la infinidad de pequeños componentes electrónicos que es posible conectar a las entradas y salidas de la placa. Sistemas domóticos, robóticos, artísticos, o incluso industriales, son abordables con este pequeño dispositivo, y es ese crecimiento de áreas de uso lo que ha llevado a Arduino a ser una empresa referente en su sector, habiéndose partido en 2005 con su placa más distintiva, Arduino One, pero contando hoy en día con multitud de otros modelos con diferentes prestaciones, tamaños y consumos [4].

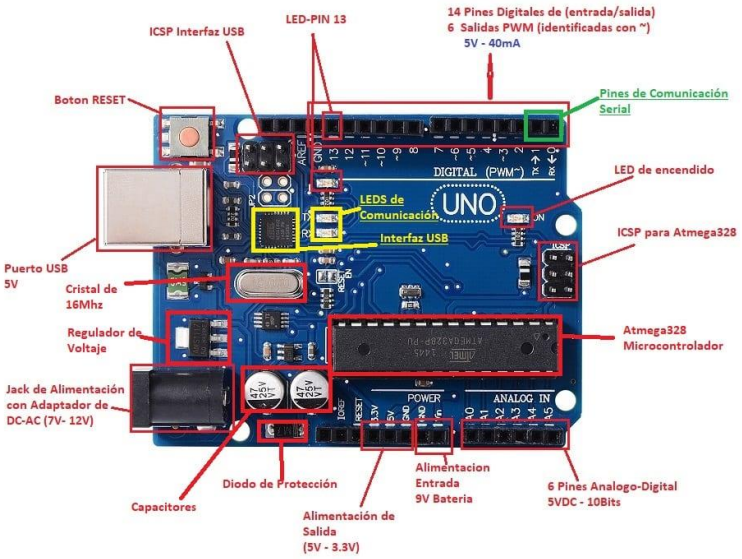


Fig. 2.1 Esquema de una placa Arduino

2.2.2 AZDelivery

AZDelivery es una empresa especializada en la venta de componentes electrónicos para plataformas de desarrollo. Aunque su principal mercado son los sensores y actuadores, AZDelivery cuenta con su propia línea de placas, fuertemente inspiradas en Arduino, con unos precios muy asequibles. Siendo un proveedor popular entre aficionados y estudiantes, AZDelivery ofrece kits de desarrollo con todo lo que pueda hacer falta para introducirse en el mundo de la electrónica y la programación, contando con recursos y soporte en línea para guiar a todos sus usuarios y sacar el máximo provecho de su catálogo de artículos. Tratando sus placas, existe una relación directa entre estas y modelos concretos de Arduino, usándose controladores idénticos que permiten a AZDelivery utilizar el IDE de Arduino como medio para la programación. Aunque menos conocida, o menos mencionada en el ámbito de la electrónica, AZDelivery cuenta con una clientela inmensa, y es su presencia en mercados como Amazon, y la velocidad que estos brindan en el envió de paquetes, lo que ha convertido a esta compañía en un absoluto favorito para la compra de componentes [5].

2.2.3 Elegoo

Elegoo es una empresa dedicada a la fabricación y venta de placas y componentes enfocada en un ámbito educativo de la electrónica. Contando con gran cantidad de kits para diferentes proyectos y diferentes niveles de experiencia, Elegoo se ha creado un nicho de mercado importante entre aquellos que quieren introducirse en este mundillo. Una gran comunidad en línea, además de tutoriales, guías, y proyectos de ejemplo, hacen del uso de Elegoo una actividad didáctica muy guiada, pudiendo aprender a utilizar gran cantidad de componentes, con la tranquilidad de poder hacerlo al ritmo que se desee. Al igual que con AZDelivery, existe un catálogo de placas fuertemente inspiradas en los modelos de Arduino, llamándose la mayoría de la misma forma, y siendo programables desde el IDE de Arduino como si de una placa propia se tratase [6].

En la Tabla 2.1 Comparativa de plataformas de desarrollo

se puede ver una pequeña comparación entre placas ofrecidas por las 3 compañías, viéndose, con tan solo mirar el nombre, la gran influencia que tiene Arduino sobre el resto de los dispositivos semejantes del mercado. Siendo entre ellas dispositivos prácticamente idénticos, las características técnicas de cada modelo son las mismas, independientemente de su productor, pero comparando los precios hay una brecha notable entre la marca con más renombre, Arduino, y la marca que busca expandirse por el mercado, AZDelivery.

Marca	Modelo	Memoria Flash	SRAM	EEPROM	Pines digitales	Pines analógicos	Precio
Arduino	Uno	32 KB	2 KB	1 KB	14	6	30 €
	Mega	256 KB	8 KB	4 KB	54	16	50 €
AZDelivery	Uno	32 KB	2 KB	1 KB	14	6	12 €
	Mega	256 KB	8 KB	4 KB	54	16	20 €
Elegoo	Uno	32 KB	2 KB	1 KB	14	6	20 €
	Mega	256 KB	8 KB	4 KB	54	16	30 €

Tabla 2.1 Comparativa de plataformas de desarrollo

2.3 Microcomputadores

Los microcomputadores son una serie de minúsculas computadoras de placa única diseñadas entorno al bajo consumo de energía y la versatilidad. Estos dispositivos, muchas veces de las dimensiones de un teléfono móvil, comprimen a menor escala todos los componentes que un ordenador convencional puede albergar. Un procesador, memorias RAM, múltiples puertos para conexión de periféricos, conectividad inalámbrica, y pines de entrada y salida son algunos de los elementos más comunes, y ya casi indispensables, en estos pequeños dispositivos. Abriéndose un paradigma completamente diferente frente a las ya mencionadas placas de desarrollo, estos dispositivos cuentan con capacidades inmensamente mayores. Aspectos como montar un sistema operativo, o la posibilidad de tener diferentes líneas de código funcionando de forma simultánea,

son funcionalidades que en base a las necesidades del proyecto pueden resultar de infinita utilidad. Existiendo una marca de referencia, y muchas marcas más recientes en el sector, en este apartado se hablará brevemente sobre aquellas que más repercusión tienen en la actualidad.

2.3.1 RaspberryPi

La Raspberry Pi es una serie de computadoras compactas y de bajo coste desarrolladas por la fundación Raspberry Pi. Bastante diferentes a todas las placas mencionadas hasta ahora, Raspberry Pi, mostrada en la Fig. 2.2, es en su concepto es un ordenador a pequeña escala. Dejando de lado todos los extras con que cuenta, y enfocándose en ello como si de una placa Arduino se tratase, Raspberry Pi es un dispositivo con entradas y salidas capaz de hacer la misma labor que cualquier otra placa, pero con unas capacidades internas inmensamente mayores. Al tratarse de un pequeño ordenador, existe una memoria, un sistema operativo y una conectividad impensables para una placa de bajas prestaciones como las mencionadas previamente, pero en la misma línea, el aumento de estas capacidades repercute en un aumento notable en el precio. Tras la pandemia vivida en el año 2020, Raspberry Pi sufrió grandes problemas con el suministro de microchips, y sus computadoras han sido un bien muy codiciado ante la imposibilidad para adquirir estos artículos durante largos periodos de tiempo en algunos países [7].

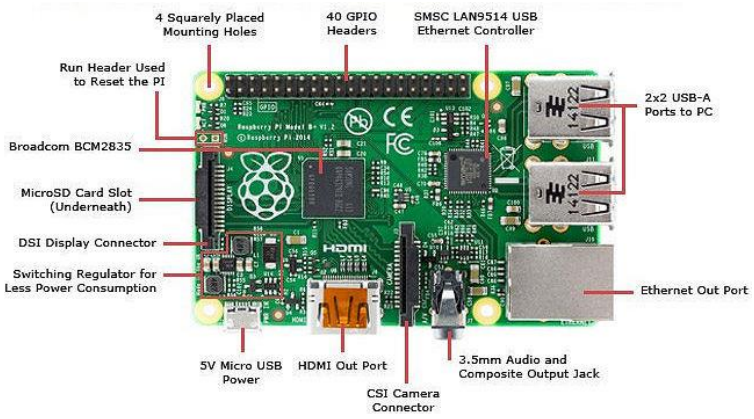


Fig. 2.2 Esquema de una RaspberryPi

2.3.2 OrangePi

La Orange Pi es una serie de computadoras desarrolladas por Shenzhen Xunlong Software. Al igual que Raspberry Pi están pensadas para ser computadoras compactas y de bajo coste, existiendo ligeras diferencias entre ambas. Comparando la Orange Pi con cualquier placa estilo Arduino se encuentra de nuevo un sistema con entradas y salidas, pero con unas capacidades inmensamente superiores. En el ámbito de precios, Orange Pi ha sido capaz de recortar frente a Raspberry Pi, y en comparación a las placas de bajas prestaciones mencionadas la brecha se hace cada vez menor [8].

2.3.3 BananaPi

La Banana Pi es una de las computadoras diseñadas por SinoVoip Corporation. En comparación con sus competidoras, BananaPi se caracteriza por haber apostado por el hardware. Teniendo un tamaño similar a una RaspberryPi, incluso algo más pequeña, la BananaPi cuenta con diversas mejoras, como el aumento en su memoria RAM o un procesador de mayor potencia, pero sus orígenes chinos aún suponen un punto de inflexión para los usuarios más puristas. En el ámbito de precios, Banana Pi gana a sus competidoras, contando con precios muy reducidos en algunos mercados online, reforzándose el pensamiento de que por ser barata es mala, algo que podría cambiar en los próximos años ante su reciente crecimiento en el mercado [9].

En la Tabla 2.2 se puede ver una pequeña comparación entre los modelos más conocidos de las 3 compañías, mostrando todos ellos características idénticas a nivel de capacidades, pero habiendo una diferencia abismal en el ámbito de precios. El dominio en el mercado había sido siempre un punto muy utilizado por la fundación Raspberry Pi para justificar sus precios, pero con todas estas nuevas compañías vendiendo productos muy similares, por una fracción del precio, la tendencia del mercado apunta a caídas considerables de precio incluso en los modelos más codiciados de RaspberryPi.

Marca	Modelo	RAM	Conectividad	Pines digitales	Precio
RaspberryPi	4 modelo B	4 GB	Wi-Fi, Bluetooth, Ethernet	40	70 €
OrangePi	4	4 GB	Wi-Fi, Bluetooth, Ethernet	40	30 €
BananaPi	BPI-M2	4 GB	Wi-Fi, Bluetooth, Ethernet	40	10 €

Tabla 2.2 Comparativa de microcomputadores

2.4 Dispositivos sensores y actuadores

Los dispositivos, bajo el contexto de las plataformas de desarrollo, son todos aquellos componentes o módulos conectables a la placa y que aportan una funcionalidad. Algunos pueden servir para ampliar las capacidades de la placa, otros permiten actuar sobre el entorno, y los más comunes son aquellos con que se puede leer el entorno. Divididos en diferentes subcategorías, los dispositivos más representativos son los sensores y los actuadores, habiendo cabida también para relojes, módulos de comunicación, módulos de almacenamiento, pantallas, y multitud de otros elementos tanto de entrada como de salida [10].

2.4.1 Sensores

Enfocándose exclusivamente en los sensores, estos son dispositivos que permiten realizar lecturas sobre el entorno. Magnitudes físicas, o magnitudes relativas, se convierten en señales eléctricas fácilmente analizables desde una placa. Algunos dispositivos pueden trabajar en modo digital, otros en modo analógico, y mientras la salida de un sensor puede ser directamente el valor requerido, otros sensores pueden necesitar el uso de librearías de código específicas para la

recogida de datos. En todo caso, un sensor no es más que un captador de una realidad, y habiendo un mercado muy extenso de ellos, es posible conseguir sensores de temperatura, luz, humedad, movimiento, sonido, o presión, entre otros muchos.

Temperatura

Los sensores de temperatura son dispositivos diseñados para medir y detectar cambios en la temperatura de su entorno. Comúnmente utilizados para sistemas de climatización, existen numerosos tipos de sensores de temperatura, los más comunes se muestran a continuación:

- LM35: Sensor analógico de alta precisión. Genera a su salida un voltaje proporcional a la temperatura en grados Celsius. Es fácilmente conectable a los pines analógicos de la placa Arduino.
- DS18B20: Sensor digital de alta precisión. Proporciona lecturas digitales tanto en grados Celsius como en Fahrenheit. Varios sensores pueden conectarse sobre un mismo pin digital de la placa Arduino.
- DHT11 o DHT22: Sensores digitales para temperatura y humedad relativa del ambiente. Utilizan un solo pin digital para proporcionar ambas medidas.
- MLX90614: Sensor de temperatura infrarrojo. Utiliza un solo pin digital para proporcionar lecturas tanto en grados Celsius como Fahrenheit.
- Termistor: Resistencias que reaccionan al calor. Es posible calcular la temperatura mediante un pin analógico y las operaciones matemáticas pertinentes respecto al valor de la resistencia.

En la Tabla 2.3 se puede ver una pequeña comparación entre algunos de los parámetros más relevantes de estos sensores, viéndose claramente cómo la mejora en rangos de medición y precisión supone un incremento proporcional en el precio.

	Rango	Resolución	Precisión	Precio
LM35	-55 ºC a +150 ºC	0,1°C	± 0,5 ºC en el rango 0 ºC – 100 ºC	0,91 € - 4,57 €
DS18B20	-55 ºC a +125 ºC	0,0625°C	± 0,5 ºC en el rango de -10 ºC a 85 ºC	0,91 € - 4,57 €
DHT11 O DHT22	0 ºC a 50 ºC	1°C	± 2 ºC	0,91 € - 2,74 €
MLX90614	-70 ºC a 380 ºC	0,02°C	± 0,5 ºC en el rango de 0 ºC a 100 ºC	9,14 € - 27,41 €
Termistor	-50 ºC a +150ºC	Depende del termistor	± 0,1 ºC a ± 0,5 ºC	0,65 € - 2,74 €

Tabla 2.3 Comparativa de sensores de temperatura

Humedad

Los sensores de humedad son dispositivos diseñados para medir y detectar la cantidad de humedad o vapor de agua presente en el aire, suelo u otros entornos. Con gran uso en el ámbito agropecuario, existen numerosos tipos de sensores de humedad, los más comunes se muestran a continuación:

- DHT11 o DHT22: Sensores digitales para temperatura y humedad relativa del ambiente. Utilizan un solo pin digital para proporcionar ambas medidas.
- FC-28: Sensor de humedad capacitivo. Proporciona una salida analógica hacia la placa. Es posible encontrar versiones que convierten el resultado a digital de forma interna.
- HR202 o HYT221: Sensor de humedad resistivo. Proporcionan una salida analógica en funcion de la resistencia eléctrica a la humedad.
- YL-69: Sensores de humedad en suelo. Proporcionan una salida analógica parecida a la emitida por los sensores resistivos.

En la Tabla 2.4 se muestra la comparación entre algunos de los parámetros más relevantes de estos sensores, siendo necesario, en la mayoría de ellos, decidir entre rango y precisión, pues para conseguir buenas calidades en ambos los precios se incrementan notablemente.

	Rango	Precisión	Precio
DHT11 O DHT22	20 % - 90%	± 2%	0,91 € - 2,74 €
FC-28	20% - 95%	± 5 % - 10%	0,91 € - 4,57 €
HR202 o HYT221	0% - 100%	± 1,8% - 10%	4,57 € - 36,55 €
YL-69	0% - 100%	± 5% - 10%	9,14 € - 15,55 €

Tabla 2.4 Comparativa de sensores de humedad

Movimiento

Los sensores de movimiento o presencia son dispositivos diseñados para detectar y responder ante cambios en la posición de objetos dentro de su campo de visión. Comúnmente utilizados en sistemas de iluminación inteligente y detección de intrusos, los más comunes se muestran a continuación:

- HC-SR501: Sensores de movimiento por infrarrojos pasivos. Proporcionan cambios en su salida digital ante la percepción de movimiento. Están planteados para la detección de movimiento humano.
- HC-SR04: Sensor ultrasónico pensado para la medida de distancias. La detección de movimiento se implementa como una reacción a variaciones de la distancia contra la que se efectúa una medida.
- RCWL-0516: Sensor de microondas para la detección el movimiento. Utiliza tecnologías de radar y es capaz de detectar movimientos a través de paredes y obstáculos.
- Laser: Sensores que reaccionan a variaciones en la incidencia de un láser. La detección de movimiento se implementa como una reacción a cambios de reflexión o interferencia del haz de luz.
- SW-420: Sensores detectores de vibración. Ampliamente utilizados para detectar tanto movimiento como intentos de manipulación de objetos. Proporcionan una salida digital ante cualquier evento.

En la Tabla 2.5 se puede ver una pequeña comparación entre algunos de los parámetros más relevantes de estos sensores, teniendo algunos de ellos rangos configurables desde el propio hardware del dispositivo, con unos márgenes de precio generalmente cercanos.

	Rango	Ajustable	Precisión	Precio
HC-SR501	Hasta 7 metros	Sí	± 3 m	1,83 € - 4,57 €
HC-SR04	2 cm a 4 metros	No	± 1 cm	1,83 € - 4,57 €
RCWL-0516	5 metros	No	Variable	1,83 € - 4,57 €
Láser	Dependiente del láser	No	Alta precisión	4,57 € - 45,69 €
SW-420	Dependiente de la vibración	Sí	± 0,01 g	0,91 € - 2,74 €

Tabla 2.5 Comparativa de sensores de movimiento

Iluminación

Los sensores de iluminación son dispositivos diseñados para detectar y medir nieves de luz o radiación lumínica en su entorno. Muy presentes en dispositivos cotidianos para la regulación de brillo en pantallas, los más comunes se muestran a continuación:

- LDR: Sensores conocidos como fotorresistores. Pequeñas resistencias con valor variable en funcion de la cantidad de luz incidente sobre ellas. Proporcionan una salida analógica.
- TSL2561: Sensor de luz digital basado en un fotodiodo que mide la intensidad de la luz.
- TSL2591: Sensor de luz analógico de alta precisión.
- BH1750: Sensor de luz ambiental de alta precisión. Proporciona una salida digital de la intensidad de la luz.
- UV: Sensor detector de radiación ultravioleta. La detección de luz se implementa como una reacción a la presencia de determinadas radiaciones UV.

En la Tabla 2.6 se muestra la comparación entre algunos de los parámetros más relevantes de estos sensores, variando entre ellos principalmente en los rangos lumínicos que trabajan, afectando estos directamente al precio del componente.

	Rango	Precio
LDR	Variable según la luz ambiente	0,91 € - 4,57 €
TSL2561	0,1 Lux - 40.000 Lux	4 € - 12 €
TSL2591	0,001 Lux - 88.000 Lux	8 € - 16 €
BH1750	1 Lux - 65.535 Lux	2 € - 4 €
UV	Variable según la intensidad de UV	4 € - 16 €

Tabla 2.6 Comparativa de sensores de iluminación

Gas

Los sensores de gas son dispositivos diseñados para detectar la presencia y concentración de gases en el entorno. Utilizados ampliamente en sistemas de seguridad y protección frente a gases tóxicos, los más comunes se muestran a continuación:

- MQ-2: Sensor para gases inflamables como metano, butano, propano, alcohol, humo y gas de hidrógeno. La concentración del gas en el ambiente es medible mediante un pin analógico de la placa.
- MQ-5: Sensor para gases como gas de propano, gas natural y gas de ciudad. La concentración del gas en el ambiente es medible mediante un pin analógico de la placa.
- MQ-7: Sensor para gases como monóxido de carbono (CO). La concentración del gas en el ambiente es medible mediante un pin analógico de la placa.
- MQ-9: Sensor para gases inflamables como metano, propano y gas natural. También es sensible al monóxido de carbono (CO). Proporciona una salida analógica medible por un pin analógico de la placa.
- MQ-135: Sensor para gases como amoníaco (NH3), óxidos de nitrógeno (NOx), alcohol, benceno y otros compuestos orgánicos volátiles. Proporciona una salida analógica medible por un pin analógico de la placa.

En la Tabla 2.7 se puede ver una pequeña comparación entre algunos de los parámetros más relevantes de estos sensores, siendo la principal diferencia entre ellos la concentración en partes por millón con que trabajan.

	Rango	Ajustable	Precio
MQ-2	300 ppm – 10.000 ppm	Sí	1 € – 4 €
MQ-5	200 ppm – 10.000 ppm	Sí	2 € - 5 €
MQ-7	20 ppm - 2,000 ppm	Sí	2 € - 6 €
MQ-9	10 ppm - 1,000 ppm	Sí	3 € - 8 €
MQ-135	Variable	No	3 € - 8 €

Tabla 2.7 Comparativa de sensores de gas

2.4.2 Actuadores

Cambiando de categoría hacia los actuadores, estos son dispositivos capaces de controlar y modificar el entorno. Utilizados principalmente como punto final de una secuencia lógica, los actuadores ponen de manifiesto las intenciones del código implementado sobre la plataforma de desarrollo, realizando las modificaciones necesarias sobre el ambiente que los rodea, de forma indefinida, o hasta que la lógica determine que es suficiente. Trabajando la mayoría de los actuadores en base a entradas digitales, unos dispositivos serán capaces de operar con tan soplo definir una salida a valor alto o bajo, mientras que otros requerirán de librerías específicas para facilitar su uso. Con un mercado extenso de estos elementos, es posible encontrar relés, motores, o luces, entre otros muchos.

Relé

Los actuadores relé son dispositivos utilizados para controlar la conexión y desconexión de circuitos eléctricos de mayor potencia funcionando como interruptores digitalmente activables. Presentes en la gran mayoría de sistemas automatizados, los más comunes se muestran a continuación:

- 1 canal: Módulo de relé electromecánico con capacidad para una sola carga. Su activación y desactivación es controlable mediante un pin digital de la placa.
- Multicanal: Módulo de relé electromecánico con capacidad para varias cargas, normalmente aumentando en potencias de 2. Las activaciones y desactivaciones se controlan mediante pines digitales de la placa, habiendo un pin por carga.
- SSR: módulo de relé de estado sólido basado en tiristores o transistores. Son más silenciosos que los relés electromecánicos pero de igual forma funcionan con salidas digitales de la placa.

En la Tabla 2.8 se puede ver la comparación de algunos de los parámetros más relevantes de estos actuadores, siendo la velocidad de reacción el principal factor de incremento de los precios.

	Velocidad de reacción	Precio
1 canal	Media	2 € - 10 €
Multicanal	Media a alta	5 € – 20 €
SSR	Alta	10 € - 30 €

Tabla 2.8 Comparativa de actuadores relé

Luces LED

Las luces led son un actuador capaz de emitir luz ante la presencia de una corriente eléctrica circulando por ellas. Muy comunes en la actualidad, y presentes en la mayoría de los dispositivos categorizados como “gaming”, las más comunes se muestran a continuación:

- LED estándar: Formato más básico de luz led. Utilizadas en proyectos sencillos, son luces capaces de emitir un solo color al recibir corriente desde un pin digital.
- LED WS2812B: Luces programables vía software. Siendo las más conocidas y utilizadas de esta lista, las LED RGB son pequeños puntos de luz capaces de emitir diferentes colores en funcion de su programación. Existen numerosas librerías de código pensadas para facilitar su uso, pudiendo ser controladas desde un pin digital de la placa.
- LED infrarrojo: Pequeñas luces led pensadas para sistemas combinados de sensores y actuadores. Emitiendo luz fuera del espectro visible humano, estas LED son comúnmente utilizadas para el control remoto, la detección de movimiento, o incluso la comunicación inalámbrica. Son elementos sencillos de usar a través de puertos digitales de la placa.

En la Tabla 2.9 se puede ver una pequeña comparación entre algunos de los parámetros más relevantes de estos actuadores, sirviendo realmente cada una para un uso especifico.

	Tipo de luz	Velocidad de reacción	Precio
LED estándar	Visible	Rápida	0,5 € - 2 €
LED WS2812B	RGB	Muy rápida	1 € - 5 €
LED infrarrojo	Infrarrojo	Variable	1 € - 4 €

Tabla 2.9 Comparativa de actuadores luz led

2.5 Estándares de comunicación

Las comunicaciones, enfocadas en el aspecto humano, son uno de los componentes básicos de todo intercambio de información. Diferentes idiomas, diferentes representaciones, o diferentes metodológicas, son solo algunos ejemplos de cómo el ser humano ha desarrollado numerosas vías para un fin común, el entendimiento.

Redirigiendo el enfoque hacia un ámbito más tecnológico, las comunicaciones también suponen un componente básico en todo aquello que hoy se conoce. Pudiendo pensar en las comunicaciones como el acto de mandar un mensaje, o realizar una llamada telefónica, la realidad es que hay mucho más por detrás de lo que el usuario común puede identificar. Diferentes tecnologías, diferentes protocolos, o diferentes arquitecturas generan una nebulosa de opciones, habiendo procedimientos muy concretos según qué se quiera hacer, pero habiendo margen de elección para otros proyectos menos rígidos. Recogiendo algunas de las tecnologías más comunes en la actualidad, en este apartado se dará una pequeña introducción a ellas, disponiendo algunas que finalmente serán parte de la maqueta [11].

2.5.1 Ethernet

Ethernet es un estándar de comunicación de redes basado en el modelo OSI. Desarrollado en la década de los 70, y posteriormente enmarcado bajo la especificación IEEE 802.3, es una solución para la transmisión de datos en áreas locales de forma eficiente y confiable. Ofreciendo soluciones cableadas, Ethernet es una de las opciones más utilizadas en entornos empresariales y educativos, existiendo un gran número de componentes, del formato switch o router, que pueden conectarse a lo largo de la red para generar despliegues sumamente complejos. Habiéndose realizado grandes avances en las capacidades de transmisión de esta tecnología, las posibilidades son casi infinitas, y tratándose de un sistema fundamentado en las capas más bajas de la pila de protocolos, el encapsulado de otros protocolos sobre Ethernet resulta una tarea sencilla y eficaz [12]. Dispositivos Ethernet muy variados rondan el mercado, pero centrándose en aquellos fácilmente gestionables desde una placa, la lista se reduce. Algunos de los más comunes y mejor valorados se muestran a continuación:

- W5100: Unos de los Ethernet shield más comunes para Arduino. Conectable directamente sobre la misma placa dota de capacidades para introducirse en una red Ethernet.
- W5500: evolución del W5100, con un chip mejorado, un menor tamaño, y con soporte para mayores velocidades.
- ESP8266 y ESP32: Originalmente dos módulos Wi-Fi, pero con la posibilidad de conectar los mismos a la red vía Ethernet.

En la Tabla 2.10 se comparan algunos de los parámetros más relevantes de estos módulos, siendo el número de conexiones simultáneas el principal factor a tener en cuenta. En el ámbito de los precios todos los dispositivos tienen en general costes bajos.

	Conexiones simultáneas	Protocolos	Velocidad	Precio
W5100	4	TCP/IP, UDP, ICMP, ARP	10/100 Mbps	5 € - 10 €
W5500	8	TCP/IP, UDP, ICMP, ARP	10/100 Mbps	7 € - 15 €
ESP8266 y ESP32	Depende del firmware	TCP/IP, UDP, ICMP, ARP	Hasta 150 Mbps	2 € - 10 €

Tabla 2.10 Comparativa de módulos Ethernet

2.5.2 Zigbee

Zigbee es un estándar de comunicaciones inalámbricas diseñado para redes de sensores, siendo el bajo consumo de energía uno de sus principales valores. Basado en el estándar IEEE 802.15.4, Zigbee abre la posibilidad de grandes despliegues de dispositivos, estructurados bajo una jerarquía de coordinadores, routers y esclavos, con comunicaciones robustas y fiables. Siendo una opción muy popular para despliegues IoT, Zigbee es una de las soluciones con alcance intermedio entre los diferentes competidores [13]. Operando entre los 10 y los 100 metros en entornos interiores, existen diferentes dispositivos capaces de utilizar este estándar para las comunicaciones seriales de las placas Arduino, algunos de ellos comentados a continuación:

- XBee: Pequeños módulos que permiten la comunicación desde el puerto serie a través de Zigbee. Existen numerosas variantes como XBee Series 1, XBee Pro-Series 1, XBee Series 2, o XBee Pro-Series 2, entre otros.
- CC2530 y CC2531: Módulos algo más económicos que XBee con la posibilidad de conectarse directamente al puerto USB de la placa Arduino.
- DigiMesh: Variante de Zigbee desarrollada por Digi que ofrece módulos con características de enrutamiento mejoradas. Pensados para despliegues mallados.

En la Tabla 2.11 se comparan algunos de los parámetros más relevantes de estos módulos, siendo la potencia el principal factor de interés. En el ámbito de precios, hay diferencias notables entre aquellos que aplican soluciones estandarizadas y aquellos que aplican soluciones más propietarias.

	Potencia de transmisión	Protocolos soportados	Precio
Xbee Series	3 dBm	Zigbee	25 € - 40 €
CC2530 y CC2531	1 dBm	Zigbee, IEEE 802.15.4	7 € - 15 €
DigiMesh	10 dBm	DigiMesh	15 € - 25 €

Tabla 2.11 Comparativa de módulos Zigbee

2.5.3 Bluetooth

Bluetooth es una tecnología inalámbrica de corto alcance pensada para la interconexión de dispositivos cercanos sin el uso de soluciones cableadas. Ampliamente utilizada en el control de dispositivos periféricos, Bluetooth es una de las soluciones de corto alcance presentes en el mercado. Existiendo numerosos perfiles, y siendo una tecnología aún en continua mejora, la gran adaptación que tiene en el mercado hace de esta una tecnología casi inevitable [14]. Con un rango de operación entre 1 y 100 metros, en funcion de la clase de dispositivos utilizados, existen multitud de elementos conectables a placas Arduino para el uso de Bluetooth como vía para la comunicación serial, siendo los más comunes comparados a continuación:

- HC-05: Módulo común y económico que permite la comunicación Bluetooth mediante la interfaz serial. Se permite tanto modo maestro como esclavo.
- HC-06: Similar al HC-05 en conexionado y funcionamiento, con la limitación de estar pensado para ejercer exclusivamente como esclavo.

En la Tabla 2.12 se comparan los principales parámetros de estos módulos, siendo su modo de funcionamiento el principal motivo para elegir uno u otro. El aumento de prestaciones se refleja de igual forma en el aumento de precios.

	Modo de funcionamiento	Alcance	Precio
HC-05	Maestro y esclavo	Hasta 100 m	6 € - 20 €
HC-06	Esclavo	10 m	3 € - 9 €

Tabla 2.12 Comparativa de módulos Bluetooth

2.5.4 Wi-Fi

Wi-Fi es un estándar de comunicaciones inalámbricas ampliamente extendido en la actualidad. Planteado para redes de área local, se ha convertido en la principal vía para el acceso a Internet, siendo una tecnología en continuo avance, con mayores velocidades y mejor seguridad. Con un gran peso en las redes IoT y la domótica, Wi-Fi se ha posicionado como una de las tecnologías básicas de la actualidad, y como cabría esperar, existen multitud de componentes pensados para acercar esta tecnología a las placas [15]. Contando las más nuevas con conectividad integrada, aquellas que aún no lo incluyen tienen a disposición dispositivos como los que se comparan a continuación:

- ESP8266: Módulo económico que permite funcionar como punto de acceso o como cliente. Utiliza la interfaz serial para realizar las comunicaciones.
- ESP32: Versión avanzada del ESP8266, con más capacidad de procesador, mayor número de entradas y salidas, y soporte para comunicaciones Bluetooth.
- Wi-Fi Shield: Placas de expansión directamente conectables sobre las placas Arduino. Existen multitud de posibilidades en el mercado, como podían ser el Adafruit CC3000 Wi-Fi Shield o el SparkFun ESP8266 Wi-Fi Shield.

En la Tabla 2.13 se comparan los principales parámetros de estos módulos, diferenciándose únicamente en las versiones de Wi-Fi soportadas, pero habiendo notables diferencias en los precios.

	Conectividad	Protocolos soportados	Precio
ESP8266	Wi-Fi 802.11 b/g/n	TCP/IP, UDP, HTTP, MQTT, etc.	2 € - 10 €
ESP32	Wi-Fi 802.11 b/g/n	TCP/IP, UDP, HTTP, MQTT, etc.	5 € - 20 €
Wi-Fi Shield	Wi-Fi 802.11 b/g	TCP/IP, UDP, HTTP, MQTT, etc.	15 € - 40 €

Tabla 2.13 Comparativa de módulos Wi-Fi

2.6 Protocolos de gestión

Los protocolos de gestión de red son conjuntos de reglas y estándares utilizados para la administración, control y supervisión de elementos en red. Numerosos protocolos y arquitecturas conviven bajo esta descripción, habiendo estado algunos diseñados específicamente para una labor de gestión, mientras que otros han terminado bajo esta etiqueta fruto de las derivas del avance tecnológico. En este apartado se hará una vista general sobre algunos protocolos, que si bien no todos están enfocados en lo mismo, todos tienen alguna aplicación que podría resultar útil ante este proyecto.

2.6.1 SNMP

El protocolo SNMP es un estándar de gestión de red desarrollado en la década de los 80 bajo la premisa de proporcionar un medio unificado para la administración y supervisión de dispositivos de red. Ampliamente utilizado en redes tipo LAN o WAN, SNMP da paso a centralizar la recogida de información de los dispositivos, permitiéndose la configuración remota o la recepción de notificaciones frente a eventos importantes. Trabajando bajo el modelo cliente servidor, como se muestra en la Fig. 2.3, se dividen los dispositivos en agentes y gestores. Los agentes son responsables de la recopilación y almacenamiento de la información respectiva al dispositivo, mientras que los gestores son sistemas encargados de analizar o recopilar los datos presentes en esos agentes [16].

SNMP, siendo una arquitectura en su conjunto, cuenta con diferentes mensajes o comandos para su uso. Transmitidos sobre UDP o TCP, con puertos 161 para los comandos generales, y 162 para las traps, los mensajes más utilizados son:

- GET: Utilizado para la recogida del valor de una variable.
- GETNEXT: Utilizado para recoger el valor de la siguiente variable disponible según el orden jerárquico de estas.
- SET: Utilizado para fijar o modificar el valor de una variable.
- GETBULK: Utilizado para la recogida de gran cantidad de valores con un solo comando.
- TRAP: Utilizado de forma inversa a todos los anteriores para permitir al agente notificar de un evento.

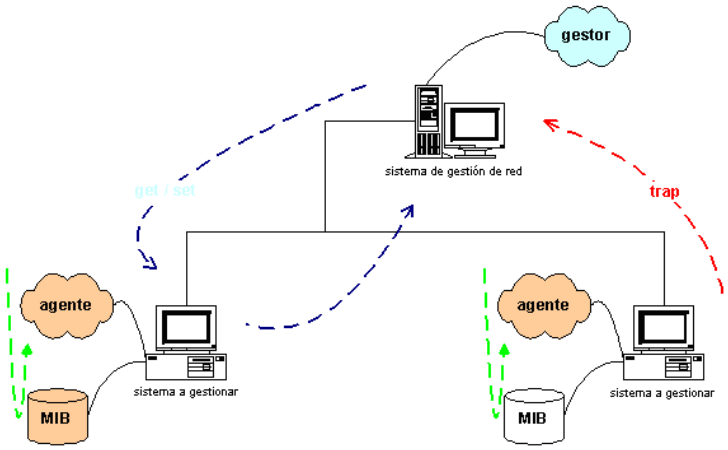


Fig. 2.3 Esquema del protocolo SNMP

Aparte de la capacidad para recoger o enviar la información, en SNMP es sumamente importante también la forma en que se organiza esa información. Siguiendo una estructura jerárquica de datos llamada MIB, todo parámetro a tener en cuenta sobre un dispositivo SNMP viene con definiciones concretas sobre sus atributos, desde el tipo de dato hasta sus posibilidades de lectura y escritura.

Siendo SNMP un mundo muy extenso, mucho más de lo aquí mencionado, cabe destacar la existencia de diferentes versiones:

- SNMPv1: Es la versión original, definiendo los más básico del protocolo, pero con capacidad suficiente para la gestión de dispositivos de red.
- SNMPv2c: Una versión algo mejorada, con nuevas funcionalidades y un gran énfasis en las "comunidades". A pesar de no ser la versión más actualizada del protocolo, SNMPv2c sigue siendo la más utilizada.
- SNMPv3: La versión más reciente y completa. Cuenta con mejoras significativas en el ámbito de la seguridad, con autenticación y cifrados además de un acceso más controlado a los dispositivos de red.

2.6.2 NETCONF

El protocolo NETCONF es un estándar de comunicación utilizado para la administración y configuración de dispositivos de red. Diseñado para facilitar la gestión de estos dispositivos, NETCONF, como se muestra en la Fig. 2.4, estandariza las interacciones entre ambas partes del modelo cliente servidor, donde los dispositivos de red actúan como servidores NETCONF, mientras que los sistemas de gestión se comportan como clientes [17].

Siendo un estándar consolidado, cuenta con numerosas operaciones que permiten la comunicación entre servidores y clientes. Siendo las más comunes solicitar información, o realizar configuraciones, las principales operaciones se muestra a continuación:

- GET: Permite solicitar información específica de estado o de variables de un dispositivo.
- EDIT-CONFIG: Permite enviar comandos de configuración para modificar parámetros del dispositivo.
- RPC: Permite ejecutar comandos específicos de forma remota en los dispositivos de red.
- NOTIFICATION: Permite a los dispositivos enviar información sobre eventos o cambios hacia los sistemas gestores.

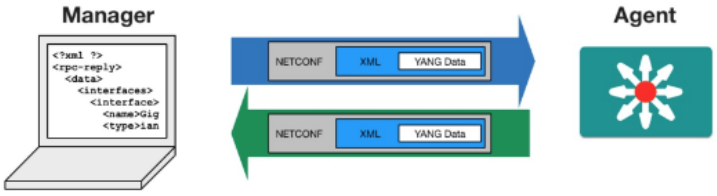


Fig. 2.4 Esquema del protocolo NETCONF

A parte de la capacidad para compartir y modificar la información, es importante una vez más seguir estructuras jerárquicas para el trato de esas informaciones. NETCONF utiliza una estructura de datos basada en XML llamada YANG, que define los objetos gestionables con que cuenta un dispositivo, y sus atributos [18].

Siendo un protocolo relativamente reciente, NETCONF surge en una etapa en que otros protocolos de gestión ya habían ido sufriendo actualizaciones, por lo cual, desde el inicio, es capaz de cubrir los grandes problemas que otros habían tenido en su origen. Aunque que NETCONF siga teniendo actualizaciones próximamente es muy probable, a día de hoy existen dos versiones diferenciadas:

- NETCONF versión 1.0: Primera versión del protocolo. Las funcionalidades básicas ya quedan establecidas, junto con XML como formato para los mensajes, y el uso de SSH para la comunicación entre clientes y servidores.
- NETCONF versión 1.1: Versión mejorada del protocolo. Se introducen las notificaciones por parte de los servidores y las transacciones múltiples, permitiéndose agrupar varias operaciones enviables en conjunto.

2.6.3 MQTT

MQTT es un protocolo de mensajería ligero y de bajo consumo diseñado para la comunicación entre dispositivos conectados bajo una red. Basado en un modelo de publicación y suscripción, MQTT es muy aplicado en el ámbito del IoT. Con dispositivos de dos tipos, publicadores y suscriptores, mostrados en la Fig. 2.5, el sistema funciona gracias a un intermediario, el broker. Publicadores serán los encargados de mandar datos al broker, y los suscriptores, suscritos a ese broker, reciben los mensajes que les correspondan [19].

No siendo como tal un protocolo para la gestión de dispositivos, MQTT se ha abierto un nicho considerable en la gestión de pequeños nodos IoT. El formato simple y ligero que lo compone da

una solución eficiente para todos esos elementos de prestaciones muy limitadas que no son capaces de cargar una arquitectura de gestión completa en su interior.

Contando realmente con pocos elementos, MQTT tiene algunos conceptos principales, explicados a continuación:

- Cliente MQTT: denominación dada tanto a publicadores como a suscriptores. Es cualquier elemento que se conecta al broker para mandar o recibir mensajes.
- Broker MQTT: Es el servidor central que recibe todos los mensajes publicados por los publicadores y que deben ser enviados a los suscriptores interesados.
- Temas: Son las etiquetas o categorías con que son publicados los mensajes. Aquellos suscriptores suscritos a un tema recibirán los mensajes que se categoricen bajo este.

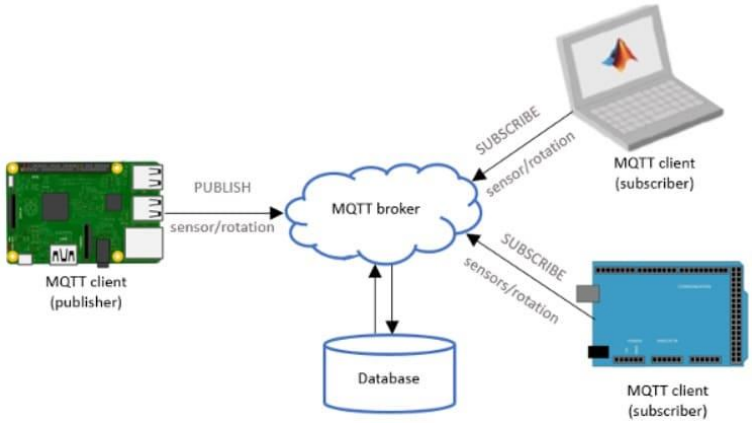


Fig. 2.5 Esquema del protocolo MQTT

Llevando algo más de una década en funcionamiento, MQTT ya cuenta con diferentes versiones del protocolo, explicándose lo esencial de cada una de ellas a continuación:

- MQTT v3.1: La versión original del protocolo. Ofrece los aspectos fundamentales de MQTT, habiendo sido ampliamente utilizada para aplicaciones IoT.
- MQTT v3.1.1: Pequeña actualización de la versión original, habiéndose centrado en la corrección de errores internos y las aclaraciones pertinentes sobre las especificaciones del protocolo.
- MQTT v5.0: La versión más reciente del protocolo, con nuevas características y funcionalidades. Algo menos utilizada hasta la fecha por el incremento de peso sufrido al tener que implementar el protocolos con esas nuevas funcionalidades.

3. PLANIFICACIÓN Y MODELADO DEL PROYECTO

El objetivo principal de este proyecto es el ensamblado de una maqueta a escala de una vivienda, sobre la que probar el funcionamiento de diferentes sensores, y la forma en que se puede responder a la información aportada por estos. Comenzado con la parte más intangible del desarrollo, este primer apartado contemplará todo el proceso previo a la implementación real, con los requisitos, la elección de dispositivos, y el diseño de un despliegue viable.

3.1 Requisitos de la instalación

Para el desarrollo del proyecto se parte de una vivienda con 140m2 interiores, divididos en dos plantas. La primera contiene la principal zona común y un cuarto de baño, y albergará un sistema de climatización inteligente y un detector de presencia para el encendido de la iluminación. La segunda, por otro lado, cuenta con dos dormitorios y otro cuarto de baño, albergando esta un sistema de alarma para la detección de intrusos.

Ambas plantas se implementarán como sistemas aislados, cuyo único punto de conexión será un sistema de comunicación serial para la actualización del estado de variables entre el controlador principal y el controlador secundario. El controlador principal, siendo el dispositivo de mayores prestaciones, contará con salida hacia Internet, albergando en su interior la arquitectura SNMP que permitirá la monitorización en tiempo real de la vivienda.

Partiendo de las necesidades establecidas, resumidas todas bajo la Fig. 3.1, y conociendo el funcionamiento que se busca lograr, es necesario enfocarse en cada uno de los sistemas a implementar, seleccionando lo dispositivos aptos para la labor que desempeñan.

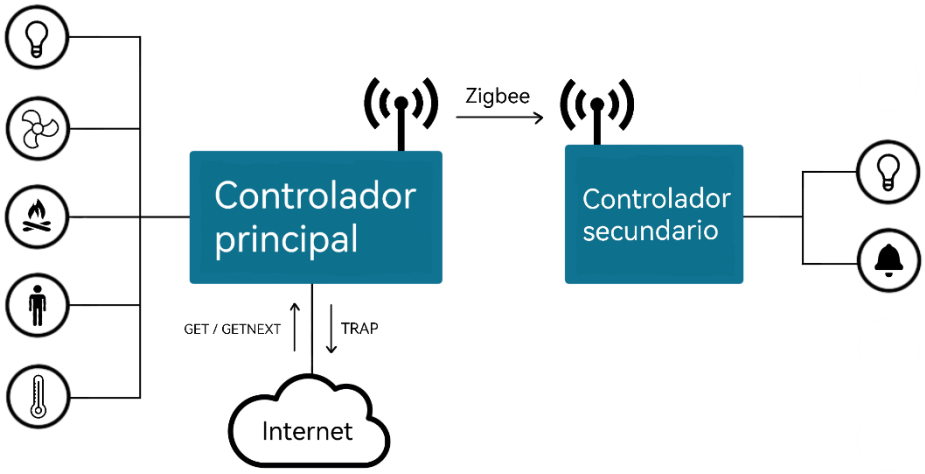


Fig. 3.1 Esquema de los requisitos

Sistema de climatización

Para la puesta en funcionamiento de un sistema de climatización, el elemento más esencial es un sensor de temperatura, en este caso un sensor DHT11. El DHT11 es un dispositivo de bajo coste que permite realizar medidas de temperatura y humedad del aire. Mediante un termistor es capaz de determinar valores de temperatura, mientras que un higrómetro se encarga de los valores de humedad. Para la realización de proyecto solo se tendrán en cuenta los valores de temperatura, los cuales son registrables mediante pines digitales, leyendo una secuencia de pulsos codificados, lo cual es fácilmente realizable gracias a librerías dedicadas para estos sensores. En funcionamiento es un sensor algo lento, pero cuenta con un rango de temperaturas entre los 0 y los 50 grados, un margen perfectamente aplicable en condiciones interiores de una vivienda. Adquirible desde numerosas tiendas, incluidas Aliexpress o Amazon, es un dispositivo popular y de bajo costo, ideal para proyectos de este estilo.

Tratando la detección de un temperatura como una acción, para una reacción adecuada es necesario un actuador. A día de hoy existen numerosas formas tanto para calentar como para enfriar una vivienda, pero todas tienen algo en común: son sistemas que se alimentan de la corriente del tendido interno de la vivienda. Trabajar con el cableado de una vivienda ya se vuelve algo más complejo, y tener conocimientos de electricidad puede resultar básico en este caso. La gestión de estos dispositivos mayores se puede realizar mediante relés. En el mercado existen numerosos relés electromecánicos, y son una opción asequible para la gestión de dispositivos que requieren una tensión de 220V. La forma en que estos relés deben actuar no es más que un pin digital de datos, controlable desde otro dispositivo, y es en funcion del estado que estos reciban que dejarán, o no, pasar esos 220V.

Comprimiendo todo el desarrollo de lo que sería una instalación real sobre una maqueta, está claro que es imposible montar un sistema con maquinarias reales, por lo cual se optará por simular las reacciones del sistema de climatización mediante pequeños dispositivos. En caso de reacción ante el frio, una chimenea montada con pequeños leds se encenderá bajo las órdenes del controlador, y en caso de reacción ante un exceso de calor, un antiguo ventilador de ordenador comenzará a funcionar. Indistintamente de tratarse de una implementación real, o de una maqueta, el funcionamiento será el mismo, en el momento que se activa un pin digital se está dando la orden de ejecutar un dispositivo, sea el más pequeño de los componentes o una máquina de tamaño industrial.

Sistema de presencia

Para el sistema de presencia, que desempeñará dos funciones, se parte del elemento esencial, un sensor PIR HC-SR501. Los sensores HC-SR501 son dispositivos utilizados para detectar la presencia de cuerpos en base a la radiación infrarroja. Movimientos de los cuerpos generan variaciones en los patrones de radiación, y estas variaciones se identifican como la existencia de una presencia. Conectables mediante puertos digitales, los sensores HC-SR501 emitirán una señal activa en caso de percibirse movimiento, y en funcion de la configuración predefinida, esta será activa en momentos puntuales, o fija mientras dure el movimiento. Son sensores baratos e igualmente se pueden adquirir en tiendas como Aliexpress o Amazon. Existen en numerosos tamaños y formas, en base a las necesidades de cada proyecto en que se puedan usar, y su único problema relevante puede ser su sensibilidad ante cambios bruscos de temperatura, que pueden causar que el sensor funcione erráticamente.

De nuevo, tras captar que se ha dado un movimiento, es necesario reaccionar a este, pero esta vez hay dos sistemas de presencia activos simultáneamente. Por un lado un sistema simple para la iluminación inteligente de la primera planta, y por otro un sistema de alarma ante la presencia de intrusos. Para el primer caso se retoma lo tratado durante el apartado de climatización, dispositivos reales. La iluminación de la vivienda de forma tradicional se gestiona mediante interruptores o pulsadores, que realmente no dejan de ser relés manuales, y es el papel de estos el que pasaría a depender de relés electromecánicos. La señal mandada por el controlador hacia el relé daría la orden de activar o desactivar la iluminación, con lo que se podría hacer una analogía entre el acto de pulsar un interruptor, y el hecho de que el controlador mande una señal. Tratando de recrear este sistema de iluminación a menor escala, los puntos de luz se convierten en pequeñas luces led repartidas por el techo de la maqueta, las cuales, como si de luces reales se tratase, reaccionan ante la presencia de movimiento.

Por otro lado, se ha mencionado ya la generación de un sistema de detección de intrusos. Hoy en día los sistemas de alarma en las viviendas son un elemento casi indispensable cuando de chalets se habla, y siendo elementos tan fáciles de implementar resultaría absurdo no aprovechar la existencia de un sensor de movimiento para implementar uno. Sobre un caso real, una alarma no es más que un dispositivo que emite ruido o que notifica de alguna forma que se ha detectado una intrusión. De nuevo, la gestión de esos dispositivos de alarma puede realizarse mediante relés. Para la maqueta, en mucha menor escala, la alarma sonora ha pasado a estar compuesta por un zumbador piezoeléctrico, un pequeño dispositivo que reacciona con sonido ante la subida de su entrada digital. Tratando en conjunto los sistemas de presencia se sigue la misma dinámica que con la climatización, los sensores detectan, el controlador actúa, siendo en esencia indiferente si la actuación es sobre un pequeño elemento, o sobre un sistema más complejo.

Comunicación serial

Para el establecimiento de unas comunicaciones en serie robustas se ha optado por una solución basada en Zigbee. Siendo una tecnología poco común en una vivienda, el uso de Zigbee garantiza la menor interferencia posible con otros dispositivos del hogar, siendo un protocolo completamente diferente a Bluetooth o a Wi-Fi.

Para la puesta en funcionamiento de esta línea de comunicación, se emplearán los módulos de comunicaciones Xbee Pro S1, un popular producto de Digi para la creación de redes de sensores inalámbricos. Siendo una de las versiones de alta potencia y alcance, y operando en la banda de 2,4GHz, estos módulos son programables desde el software XCTU, y ofrecen diferentes aspectos de direccionamiento y seguridad que pueden ser útiles en una vivienda.

Vistos desde el controlador, estos dispositivos no son más que una puerta al envío de datos. Poco presentes dentro del código a implementar, estos dispositivos se encargarán de mandar y recibir mensajes en funcion de las órdenes dadas por el controlador. La utilización de un módulo Xbee tanto en el controlador central como en el controlador secundario representa una solución eficaz tanto para la maqueta desarrollada como para un despliegue real

Comunicación Ethernet

Para la utilización de SNMP como protocolo de gestión para los dispositivos, se ha optado por el uso de Ethernet como medio único para el envío de información hacia Internet. Utilizando esta solución cableada, se plantea el posicionamiento del controlador principal como un módulo conectado directamente al router de la vivienda, dependiéndose únicamente de la dirección IP que ese dispositivo reciba.

El módulo de comunicaciones Ethernet utilizado será el W5500, un controlador integrado que permite la conexión a Internet. Siendo un dispositivo de bajo consumo, proporciona una conexión de alta calidad ideal para proyectos IoT. Mediante un puerto RJ45, el W5500 permite comunicaciones hasta los 100Mbps de forma teórica, habiéndose mejorado frente a su hermano menor, el W5100, y contándose además con una pequeña memoria interna de 32KB utilizada como buffer.

Trabajando como punto de salida del hogar hacia Internet, el W5500 establecerá las bases de la pila de protocolos, montando sobre él todas las recepciones y envíos necesarios para el funcionamiento como cliente SNMP, siendo el mismo modo de operación tanto para una maqueta como para un despliegue real.

3.2 Despliegue de dispositivos

Conocidos los sistemas a implementar, es necesario enfocarse en la forma en que estos conviven con el entorno. Como se ha mencionado previamente, se parte de una vivienda de dos plantas, interpretándose cada planta como un sistema independiente, pero existiendo una comunicación entre controladores para dotar a la vivienda de una visión de conjunto. Y habiéndose hablado ya de los diferentes dispositivos que conformarán el sistema, solo queda un elemento no tratado, los controladores.

Ya mencionados a lo largo del proyecto, los controladores son el centro de toda la implementación. Dispositivos capaces de tomar decisiones en base al código previamente implementado, y en continuo contacto con los sensores y actuadores. Dadas las numerosas opciones disponibles en el mercado, se ha elegido utilizar placas de la marca Arduino por su reconocimiento y la sólida comunidad de desarrollo que las respalda

Necesitándose dos controladores de capacidades bastante diferentes, el controlador principal, o cerebro de la vivienda, será encarnado por una placa Arduino Mega, mientras que el secundario será un dispositivo algo más sencillo, una placa Arduino One.

Arduino Mega

Arduino Mega es una placa de desarrollo de código abierto establecida como la versión mejorada de la Arduino One. Esta placa cuenta con un microprocesador ATmega2560, con 256KB de memoria Flash para el almacenamiento de programas. Siendo una placa basada en pines de entrada y salida, se cuenta con 54 pines digitales y 16 analógicos, un número considerablemente mayor frente a otras placas del catálogo, y que permite la realización de proyectos más complejos. El aumento de capacidades repercute de igual forma en un aumento de precio, y siendo una de las placas más caras de la empresa es posible encontrarla en rangos entre los 30€ y los 60€, existiendo un gran mercado de “inspiraciones” más baratas por parte de otras compañías.

El papel de esta placa para el desarrollo del proyecto es fundamental. Arduino Mega será el verdadero cerebro de la casa, y su papel principal recae en dar vida al cliente SNMP. Es conocido que SNMP es una arquitectura robusta, con una gran meticulosidad en el trato de la información, y una gran carga detrás de ello, pero tratando de poner en marcha algo de tal magnitud, sobre un sistema tan minúsculo como termina resultando Arduino, es de esperar que durante el desarrollo de algunas ideas se encuentren contratiempos difíciles de abordar.

Además del papel lógico, Arduino Mega desarrollara también una labor importante en la administración de los diferentes dispositivos repartidos por la vivienda, actuando de forma directa sobre algunos, o teniendo que relegar hacia más placas para otros.

Arduino One

Arduino One es una de las placas de desarrollo de código abierto más conocidas del mercado. Siendo el dispositivo por excelencia de la marca Arduino, esta compacta placa contiene un microprocesador ATmega328P, junto a una memoria flash de 32KB para el almacenamiento de programas. Siendo el inicio de una larga cadena de placas basadas en pines de entrada y salida, esta cuenta con 14 pines digitales y 6 analógicos, ideal para introducirse en el mundo de la electrónica con proyectos sencillos y de bajos requisitos. Tratándose del buque insignia de la marca, esta placa es relativamente barata, encontrándose entre los 20€ y los 30€, existiendo para este caso también un gran mercado de placas fuertemente “inspiradas” en ella.

Para el desarrollo de este proyecto, la presencia de Arduino One supone la entrada de un nuevo frente de cara a la programación, la administración de dos controladores, junto a todos los elementos que de estos cuelgan, desde un único cerebro. Sobre una maqueta una sola placa es más que suficiente para montar un buen sistema domótico, pero en una aplicación real, la cantidad de cable necesario para extender la domótica por toda una vivienda es impensable. Trabajando con tensiones y corrientes tan bajas, las perdidas serian inmensas, por lo cual un despliegue inalámbrico de diferentes controladores podría ser una opción fácil. Arduino One pone de manifiesto esta comunicación inalámbrica entre controladores, ejerciendo como esclavo del cerebro de la casa, al tiempo que es maestro de aquellos dispositivos que controla.

Conociendo los dispositivos que conformaran el sistema, es momento de plantear sobre plano las conexiones entre ellos, definiendo pines y localizaciones determinadas para cada sistema, facilitando en todo momento las posteriores modificaciones que sean necesarias sobre el hardware de la maqueta.

Comenzando con la placa Arduino Mega, se conectan a ella la mayoría de los dispositivos. A nivel comunicaciones, un módulo Xbee y el módulo Ethernet se conectan a sus salidas de comunicación y de puerto serie respectivamente. La sensórica por otro lado cuenta con los dos sensores principales, cada uno con su pin de datos. Y como respuesta para la sensórica, existen 3 sistemas que ejercerán como actuadores para la maqueta, igualmente conectados todos ellos gracias a pines digitales. Entrando en detalle más profundos, y partiendo de que la placa Arduino Mega es el centro, se desarrollaran los conexionados individuales de cada dispositivo:

- Xbee: El módulo Xbee Pro S1, montado sobre su shield de conexión, cuenta con 5 salidas. 4 de ellas serán utilizadas para recepción, transmisión y alimentación, respectivamente, mientras que la quinta queda al aire. Sobre Arduino Mega, los puertos de comunicación utilizados para trabajar sobre Zigbee son aquellos predefinidos para las comunicaciones en serie, los pines 0 y 1. De forma gráfica, como se puede observar en la Fig. 3.2, de derecha a izquierda, los pines del módulo de conexión Xbee quedan como: 5V, GND, Tx y Rx.

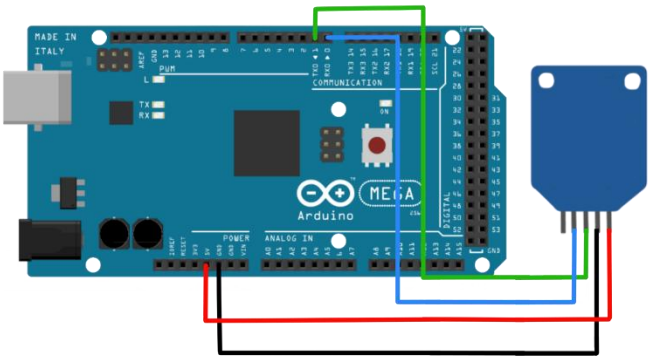


Fig. 3.2 Conexionado módulo Xbee

- Ethernet: El módulo W5500, en su versión de tamaño reducido, cuenta con 10 salidas. 7 de ellas serán utilizadas, mientras que las otras tres quedan al aire. Siendo algo más complejo en su conexionado, el módulo Ethernet conecta 6 de sus puertos directamente con los pines centrales de comunicación de Arduino, mientras que deja uno conectado al pin digital 10. En función de la placa, este último puerto puede variar, y es que las diferentes versiones de microprocesador tienen diferentes despliegues, no coincidiendo el pin 10 de la Arduino Mega 2560 con el de su predecesora, la Mega 1280. En la Fig. 3.3 quedan representadas todas las conexiones.

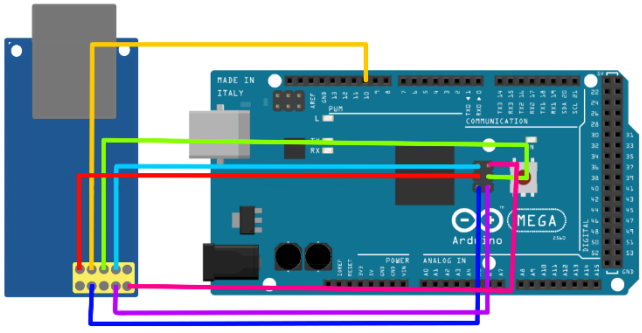


Fig. 3.3 Conexionado módulo Ethernet

- Temperatura: El módulo DHT11, en su versión embebida sobre placa de 3 pines, cuenta con dos salidas para la alimentación y un pin de datos. No necesitándose elementos intermedios, el sensor se puede conectar directamente sobre la placa, como se ve en la Fig. 3.4, colocando la alimentación respectiva sobre 5V y GND, y quedando el pin de datos establecido en el pin digital 24 de la Arduino Mega.

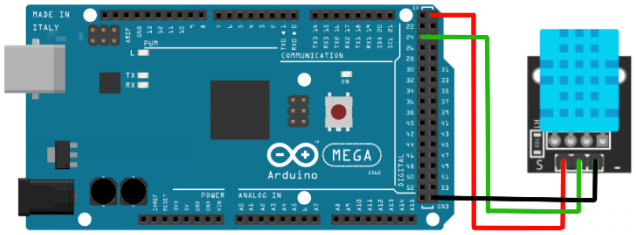


Fig. 3.4 Conexionado sensor de temperatura

- Movimiento: El módulo HC-SR501, modelo con cúpula, cuenta con dos salidas para la alimentación y un pin de datos. Al igual que con el sensor de temperatura, este es completamente independiente, y funciona con tan solo conectarlo de forma correcta. Con los pines de alimentación en su correspondiente posición, el pin de datos quedará en el puerto 25 de la placa Arduino Mega, como se muestra en la Fig. 3.5.

  Algo a tener en cuenta a la hora de conectar este sensor son los diferentes parámetros que son modificables directamente desde el hardware. En la parte frontal se encuentran dos pequeñas ruedas, siendo una el tempo que duran los impulsos mandados ante la detección de movimiento, y siendo la otra una estimación de la distancia a la que se aproxima vaya a estar el movimiento que se busca detectar. En este segundo caso, el sensor está pensado para distancias entre los 3 y los 7 metros, distancias inmensas en relación al tamaño de la maqueta. Por comodidad en su programación se ha mantenido este sensor, pero a la hora de probar su funcionamiento sobre un entorno tan minúsculo es muy probable que se encuentren fallos en la detección.

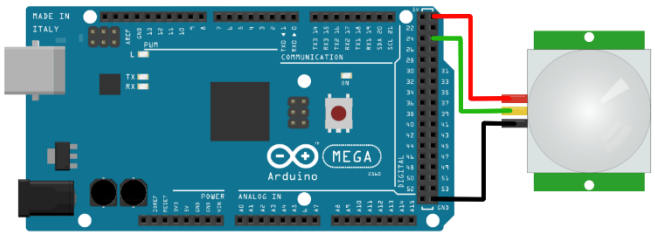


Fig. 3.5 Conexionado sensor de movimiento

- Luces: Para simular la existencia de un tendido lumínico semejante al de una vivienda, se han repartido luces led por el interior de todos los cuartos. Cada planta cuenta con una red propia, y cada red está conectada a la placa de su respectiva planta. En el caso de la primera planta, sobre la Arduino mega, como se muestra en la Fig. 3.6, se han conectado estas luces led al pin 22, existiendo otra toma que se conecta al GND.

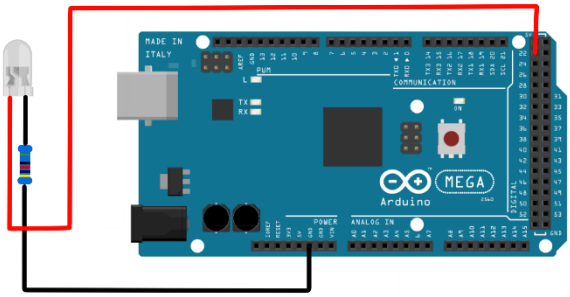


Fig. 3.6 Conexionado iluminación

- Chimenea: Para la simulación de un sistema que reaccione ante las temperaturas bajas captadas por el sensor de temperatura, se ha optado de nuevo por el uso de luces led, esta vez simulando una chimenea que se enciende. Existiendo solo un "punto de calor" en la maqueta, la instalación de cables se asemeja mucho la realizada para la iluminación de la Arduino Mega, quedando en el pin digital 23 como se muestra en la Fig. 3.7.

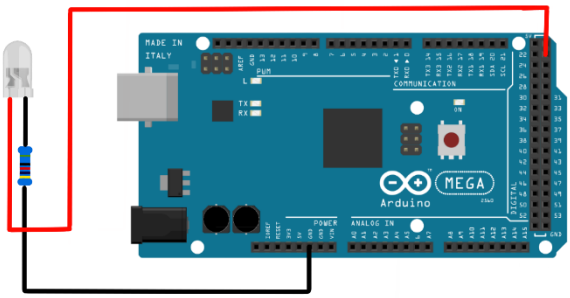


Fig. 3.7 Conexionado chimenea

- Ventilador: En línea con la chimenea, la contrapartida es un sistema que enfríe la casa, habiéndose utilizado en este caso un antiguo ventilador de pc. Escondido en el techo de la primera planta, el ventilador es activado por un pin de datos, quedándose en el pin 29 de la placa Arduino Mega, como se muestra en la Fig. 3.8.

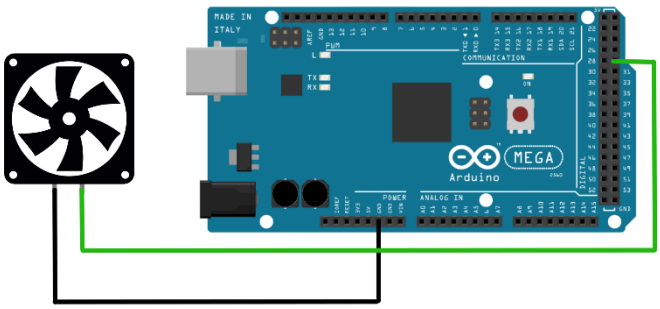


Fig. 3.8 Conexionado ventilador

En el caso de la Arduino One, se reduce notablemente el número de conexiones. A nivel comunicaciones, un módulo Xbee se conecta para la comunicación serial. Sin sensores, esta placa cuenta con un solo sistema que hará funcionar de forma simultánea 2 actuadores con salidas digitales. Entrando en detalle más profundos, y partiendo de que la placa Arduino One es el centro, se desarrollaran los conexionados individuales de cada dispositivo:

- Xbee: Utilizando de nuevo el módulo Xbee Pro S1, se cuenta con 5 salidas, de las cuales solo 4 son utilizadas. Siguiendo la misma estructura seguida para la Arduino Mega, en la Fig. 3.9 se muestran las conexiones de los pines de recepción, transmisión y alimentación, quedando de derecha a izquierda como: 5V, GND, Tx y Rx.

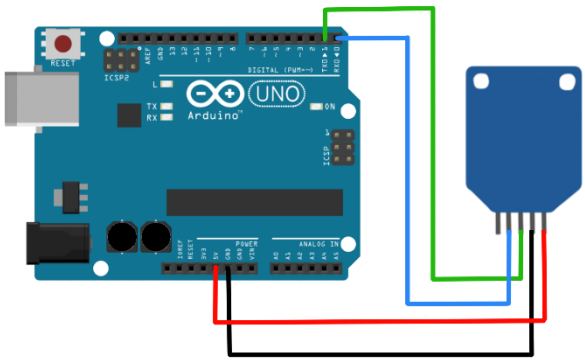


Fig. 3.9 Conexionado módulo Xbee

- Luces: Como se realizó para la planta anterior, la existencia de un tendido luminoso se simula con luces led, quedando para la segunda planta, sobre Arduino One, estas luces conectadas al pin 13, de nuevo con otra toma conectada a GND, como se muestra en la Fig. 3.10.

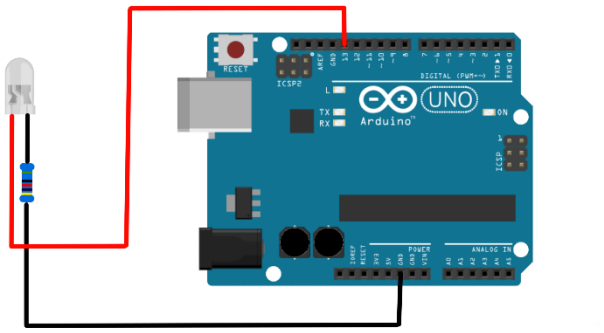


Fig. 3.10 Conexionado iluminación

- Alarma: Por último, para finalizar el único sistema pendiente, se ha instalado sobre la placa Arduino One un pequeño vibrador piezoeléctrico, un dispositivo que alimentado por un pin digital, que como se muestra en la Fig. 3.11 es el pin 9, emite tonos que imitaran una alarma de intrusos.

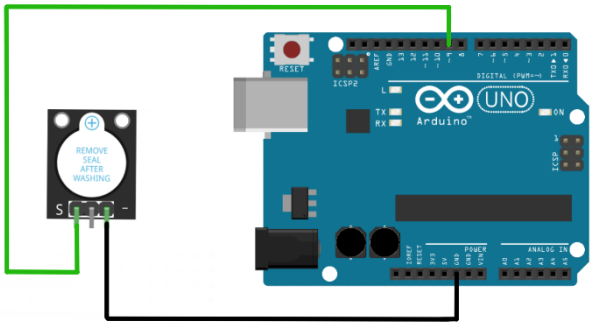


Fig. 3.11 Conexionado alarma

4. IMPLEMENTACIÓN DE LA SOLUCIÓN

Tras el desglose de los requisitos, y el aporte de soluciones que se adapten al proyecto, se puede, finalmente, comenzar con la composición conjunta. Identificando cada sistema como una unidad independiente, los códigos de estos serán desarrollados de forma paralela, como si de proyectos separados se tratase, para finalmente quedar unidos sobre el programa que se cargará sobre las placas Arduino. En este capítulo, se seguirán los desarrollos elaborados durante cada fase del proyecto, finalizándose con el ensamblado de la maqueta, con todos los sistemas listos para su prueba de funcionamiento.

4.1 Desarrollo del código

Arduino, siendo una plataforma de desarrollo, cuenta con una aplicación dedicada para facilitar la programación de sus placas: el IDE de Arduino. Mostrada en la Fig. 4.1, la interfaz de la aplicación es bastante sencilla. Con un gran área destinada para la escritura de código, los programas sobre Arduino se dividen en tres secciones: el inicio, la puesta en marcha y el bucle.

- El inicio, como su propio nombre indica, es la primera parte del código, previa a las funciones de puesta en marcha y bucle. Ejecutándose inmediatamente después de encenderse la placa, es la sección en que se pueden cargar librerías de código, declarar variables, o ejecutar alguna función previa al uso.
- La puesta en marcha, ejecutada justo después, es una parte del código en que se mantiene esa idea de ejecutable una sola vez, cargándose en esta parte parámetros estáticos como la velocidad del puerto serie, o parámetros concretos referentes a librerías definidas en el inicio.
- El bucle, por otro lado, es la parte del código que se repetirá cíclicamente de forma lineal. El funcionamiento estrella de las placas Arduino recae sobre esta función, y el usuario debe ser capaz de administrar sus sistemas siendo consciente de que una vez se entra no se va a parar de repetir. En la funcion bucle se debe recopilar la información de los sensores, trabajar con los actuadores, implementar las comunicaciones externas y gestionar las impresiones por pantalla, todo de forma secuencial y respetando las necesidades de cada bloque de código.

Conocido el medio de trabajo, el siguiente paso es conocer las implementaciones de cada dispositivo, siendo algunos más simples y otros más complejos, pero contando todos con hojas de características y una gran comunidad en línea que aporta soluciones.

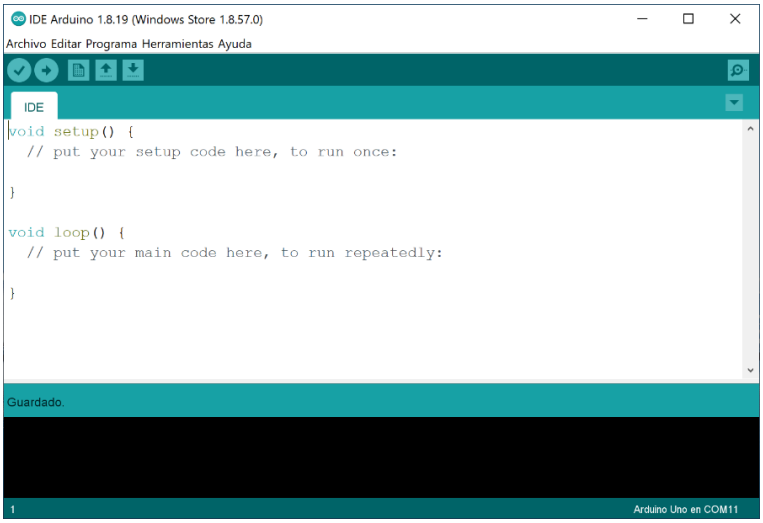


Fig. 4.1 Interfaz IDE de Arduino

4.1.1 Comunicación serial

La comunicación serial en Arduino se define como un método utilizado para el traspaso de información entre diferentes dispositivos o diferentes placas, permitiéndose el uso del protocolo UART bajo las conexiones cableadas de los pines de transmisión y recepción con que cuentan todas las placas. Transmitiendo bytes, y estando disponible el uso de numerosas tecnologías por debajo, la programación de estas conexiones no requiere más que algunos comandos esenciales de Arduino. Como se muestra en la Fig. 4.2, la recepción y transmisión de mensajes es bastante simple, existiendo numerosas ampliaciones que se pueden realizar sobre estos comandos para modificar tipos de datos, formas del envió, o velocidades.

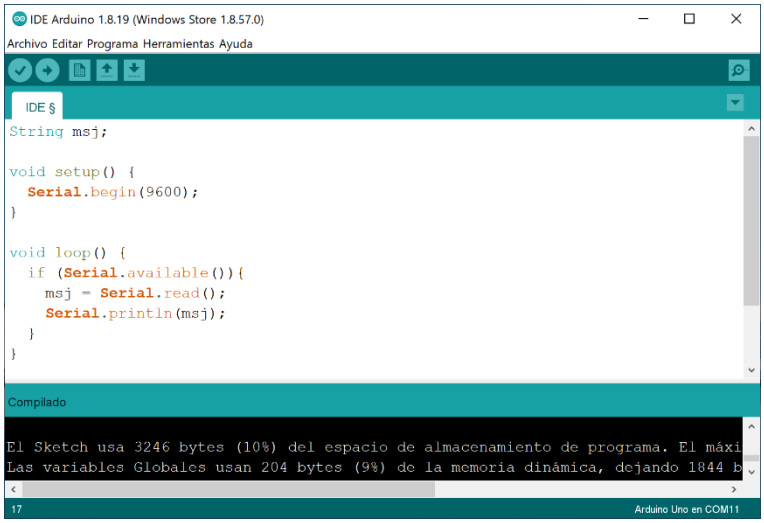


Fig. 4.2 Programación de comunicación serial

Para dar funcionamiento a esta comunicación, en este proyecto se optó por los ya mencionados módulos Xbee, dispositivos conectados a los pines de transmisión y recepción que permiten estructurar una pila completa de protocolos de conexión, siendo Arduino el generador de los datos. Los módulos Xbee cuentan con su propia aplicación, XCTU, mostrada en la Fig. 4.3, desde la cual es posible modificar parámetros del funcionamiento de estos dispositivos. Direcciones origen y destino, modos dispositivo final o coordinador, o contraseñas para cifrado AES, son algunos de los campos modificados durante la elaboración del proyecto, con el fin de realizar un montaje lo más preciso posible [20].

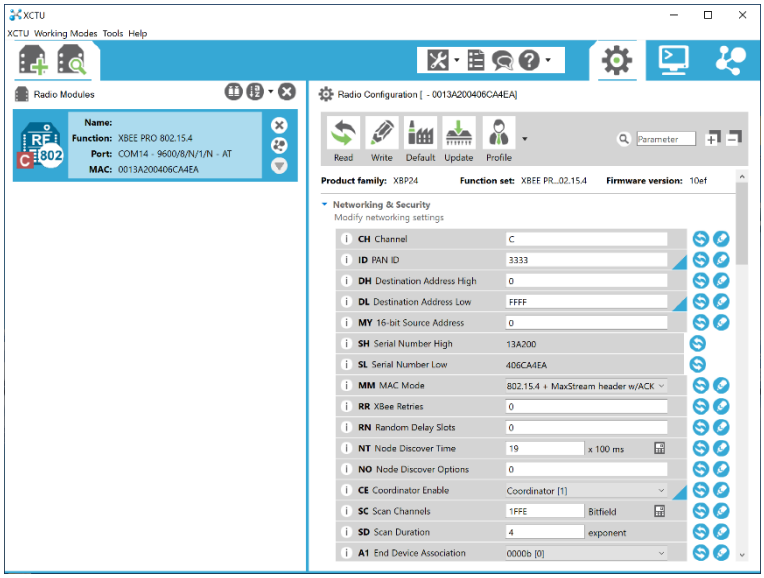


Fig. 4.3 Interfaz de XCTU

Funcionando el módulo conectado a la placa Arduino Mega como coordinador, y el módulo conectado a la Arduino One como dispositivo final, ambos contarán con las direcciones del otro como punto destino de los mensajes, existiendo una comunicación directa en todo momento. Parámetros que pueden configurarse como dinámicos, y que son específicos de Zigbee, como pueden ser la identificación de la PAN o el canal, se dejarán predefinidos y estáticos para facilitar el uso de estos módulos, pues una vez configurados al inicio no se pretende volver a modificar ningún parámetro. Un aspecto clave, tratándose de una comunicación serial inalámbrica, es el uso de seguridad, permitiéndose el establecimiento de contraseñas para la utilización de cifrado AES. Vistos desde la placa Arduino ninguno de estos aspectos afecta al código, pues Arduino se limita a aprovechar la puerta dada, sin entrar en detalle de qué o cómo es esa puerta.

Conociendo todo los aspectos relevantes tanto de la comunicación serial general, como de los módulos específicos, el código empleado para la comunicación entre ambas placas es bastante sencillo. En línea con la Fig. 4.2, previamente comentada, Arduino Mega ejercerá como emisor de mensajes, dependiendo estos de los sensores que cuelgan de esta placa, que serán comentados más adelante, mientras que Arduino One será un receptor, que en base a qué recibe identificará el proceso que seguirá tras ello.

4.1.2 Comunicación Ethernet

La comunicación Ethernet en Arduino se define como la capacidad de la placa para conectarse a esta red y comunicarse con otros dispositivos presentes en ella. Aplicada sobre el proyecto, Ethernet actúa como el nexo común entre el sistema domótico e Internet. Siendo la base para la implementación del protocolo de gestión SNMP, la labor de Ethernet se vuelve bastante más extensa a medida que se desarrollan sus funcionalidades. Como se muestra en la Fig. 4.4, utilizar Ethernet sobre Arduino es relativamente sencillo, pues existen librerías específicas que hacen gran parte de la labor por el usuario. La dirección MAC, la dirección IP, la dirección de salida, o la máscara de la subred son parámetros fácilmente configurables de forma estática, y existen diferentes vías para permitir la asignación también de valores dinámicos. A la hora de realizar una comunicación, al igual que pasaba en la serial, la dinámica es comprobar si se ha recibido algo o si se tiene que mandar algo, contando el módulo w5500 empleado con una pequeña memoria que actúa como almacén temporal para los paquetes recibidos [21]. Siendo hasta aquí todo relativamente sencillo, el principal bloque de carga tras la comunicación Ethernet es el uso de SNMP.

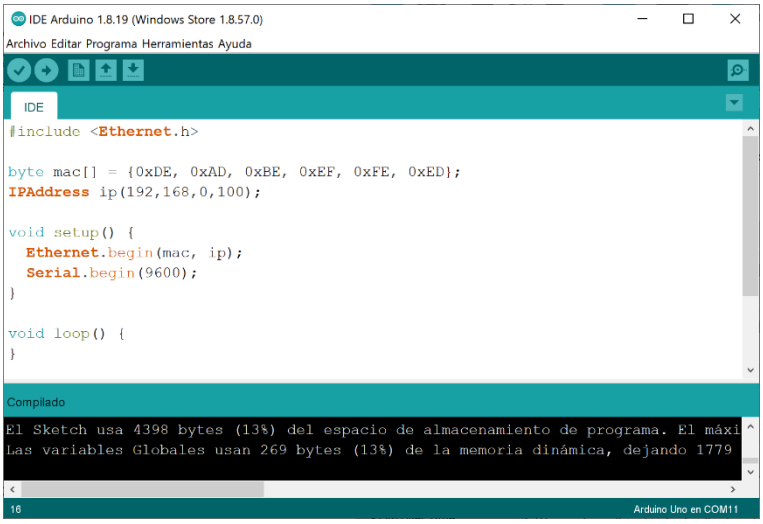


Fig. 4.4 Programación de comunicación Ethernet

SNMP, como protocolo que es, cuenta con estructuras complejas muy marcadas para todos sus tipos de mensajes. Siendo una solución muy específica, se abre un problema para su implementación, buscar librerías que realmente funcionen. Existiendo gran cantidad de "soluciones" presentadas por multitud de usuarios, con más o menos capacidades, la búsqueda de una librería de código que realmente funcionase en alguna de las versiones de SNMP ha sido complicada. SNMP lleva muchos años en el mercado, y pese a ser una solución muy estudiada y eficaz, está lejos de ser optima sobre este tipo de placas de desarrollo. Multitud de proyectos datados de hace una década se ofrecen como soluciones para la implementación de un agente SNMP dentro de una placa Arduino, pero siendo la mayoría proyectos abandonados, las actualizaciones sufridas por el resto de los componentes que conforman el sistema han mellado su funcionamiento. Librerías desactualizadas, compiladores en desuso, o tipos de datos ya no soportados son algunas de las mayores trabas que se han ido encontrando mientras se probaban multitud de librerías. Llegados a un punto en que la solución llegaba a ser peor que el problema se retorna a la idea inicial, una librería antigua, pero simple y fácil de modificar, Agentuino [22].

Agentuino es un proyecto de GitHub publicado en el año 2014 para dar soporte a las funcionalidades básicas de SNMPv1. Contando con cabida para los comandos de GET y GETNEXT, se proporciona una librería muy limitada, pero suficiente para las labores de monitorización buscadas en este proyecto. De forma añadida a las funcionalidades iniciales, se decidió programar el envío de traps como forma de notificar eventos relevantes, empleándose la librería Ethernet de dos formas diferentes. Por un lado, Agentuino es montada directamente sobre Ethernet, aplicando conversiones internas que permiten la generación de tramas sin la necesidad de que el usuario tenga que modificar grandes secciones del código. Por otro lado, las traps se han montado sobre librerías UDP, posteriormente enviadas sobre la librería Ethernet, compartiendo ambos procedimientos esta librería, uno para la inserción directa, y el otro para la inserción combinada.

Continuando con SNMP como parte fundamental del código, uno de los aspectos básicos para su implementación es el montaje de una MIB. Única para la vivienda completa, y con parámetros tanto informativos como de elementos presentes en ambos controladores, la MIB queda definida sobre la placa Arduino Mega como el almacén principal de variables. Mostrada en la Fig. 4.5, la MIB del proyecto cuenta con 3 partes diferenciadas. La parte inicial cuenta con la información de la vivienda, datos estáticos que permiten conocer el sistema que se está monitorizando. La segunda parte, compuesta realmente por la segunda y la tercera, es la zona de los controladores. Existiendo para este proyecto dos, se realiza un desglose de toda la información de cada una de las placas presentes en la vivienda, con la información estática relativa a ellas, y con apartados específicos para sensores y actuadores, donde se almacenan los valores puntuales de cada uno de los sistemas. Algunos, como el sensor de temperatura, tendrán valores numéricos en un rango extenso, mientras que otros, como el estado de las luces, se limitarán a valores 1 o 0, on u off.

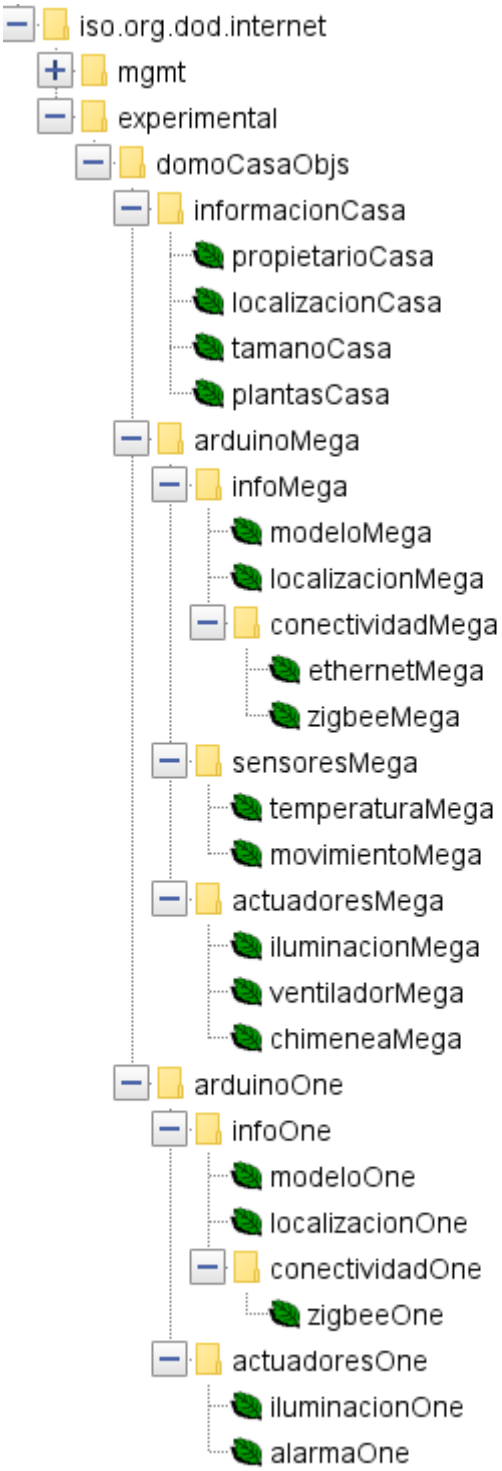


Fig. 4.5 Estructura de la MIB

4.1.3 Control de los dispositivos

Habiéndose desarrollado ya tanto la estructura de SNMP como los procedimientos para las diferentes comunicaciones, el único factor restante es la interacción con el medio que de vida al sistema. Contando con sensores y actuadores, y centrados en la maqueta, cada elemento cuenta con un código especifico, algunos similares, pero otros radicalmente diferentes. Por motivos de claridad cada elemento será explicado por separado, de forma que el fundamento tras cada código sea fácilmente comprensible.

Temperatura

El módulo DHT11 es un pequeño pero potente elemento con una programación particular. Como se muestra en la Fig. 4.6, este dispositivo aprovecha la librería DHT para realizar las lecturas de diferentes datos sobre el mismo pin digital de la placa. Capaz de leer temperatura y humedad, todos los datos serán introducidos en el código final, pero será solo el dato de temperatura el que pase a rellenar su campo respectivo en la MIB. Para facilitar la depuración de código en las etapas finales del proyecto, todo lo necesario para las lecturas del sensor conformará una funcion independiente, que será llamada en cada iteración de la funcion bucle, realizándose acto seguido las activaciones o desactivaciones pertinentes de actuadores en base a los datos recogidos. En caso de subir la temperatura de los 25 ºC, el sistema está configurado para dar la orden de activar el ventilador, mientras que por debajo de los 15 ºC se activa la chimenea. En caso de volverse a valores entre ambos umbrales los sistemas correspondientes se apagarían, generando ante todas las modificaciones una trap que notifica al usuario estos eventos [23].

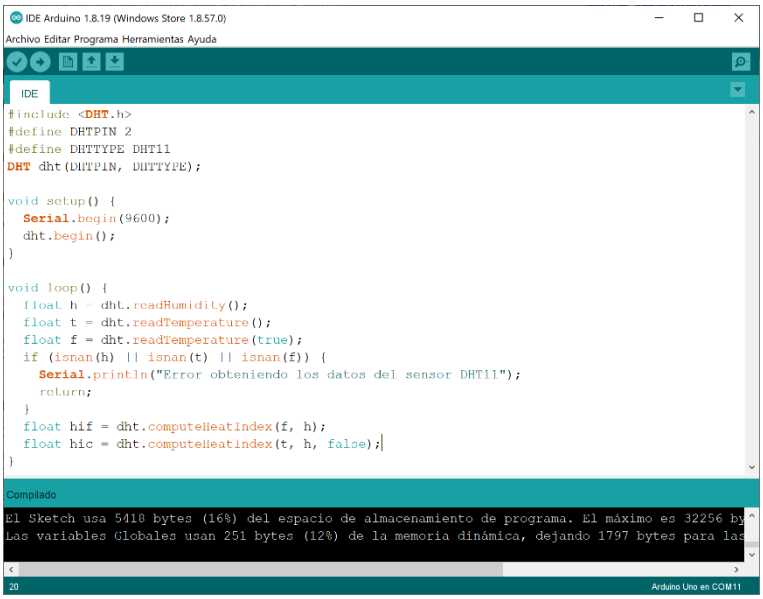


Fig. 4.6 Programación del sensor de temperatura

Movimiento

El módulo HC-SR501 es un dispositivo bastante fácil de utilizar. Como se puede ver en el código de la Fig. 4.7, su funcionamiento se basa en la detección de un valor alto o bajo. Sin necesidad de librerías ni grandes desarrollos de código, la detección de movimiento acarreara una secuencia de procesos dependientes de más factores. En caso de que la alarma este manualmente desconectada, una detección de presencia afectará exclusivamente al sistema de iluminación de la planta inferior, encendiéndose las luces mientras dure el movimiento, y quedando todos estos cambios presentes en valores de objetos de la MIB. Por otro lado, en caso de que la alarma este activa, la detección de movimiento realizará su habitual encendido de luces de la planta baja, pero además mandará la información pertinente al Arduino One mediante comunicación serial, para hacer sonar la alarma y encender las luces también de la segunda planta [24].

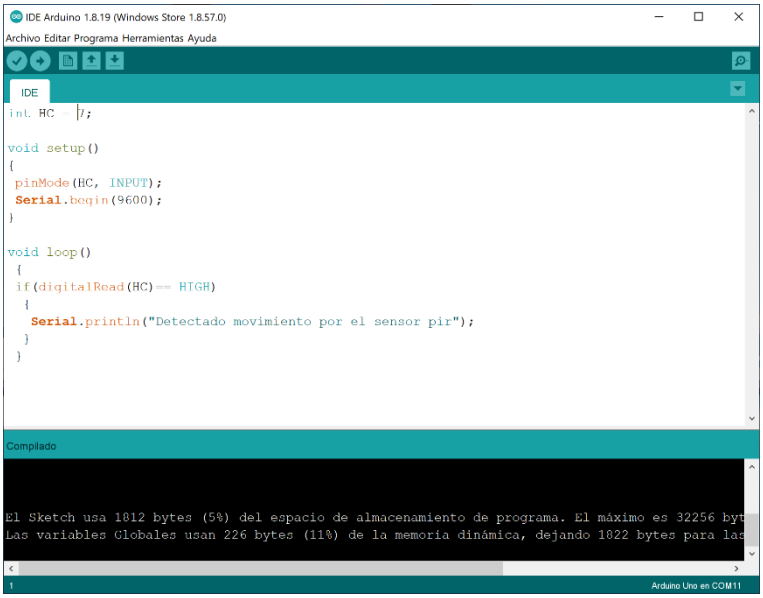


Fig. 4.7 Programación del sensor de movimiento

Chimenea y ventilador

Dos actuadores dependientes del sensor de temperatura, con una programación idéntica. Tratándose de actuadores, estos dispositivos se basan en la existencia de un pin digital que active o desactive el sistema. Quedando ambos controlados por el código que conforma el sistema de climatización, su programación se asemeja a la mostrada en la Fig. 4.8.

Iluminación

Independientemente de la planta a la que se refiera, la iluminación no es más que un actuador dependiente del sensor de movimiento. Con una programación sencilla, de nuevo semejante a la mostrada en la Fig. 4.8, valores altos o bajos determinarán el encendido o apagado de luces a petición de la configuración previamente desarrollada en el programa.

Fig. 4.8 Programación de actuadores simples

Alarma

Por último, y siendo el actuador más diferente, el vibrador piezoeléctrico requiere el uso de funciones internas de Arduino para la generación de un tono. Como se muestra en la Fig. 4.9, y sin ser tampoco un código muy complejo, este dispositivo recibe varios parámetros, y en funcion de ellos emite unos sonidos u otros. Como parte del sistema de alarma, este actuador se ejecuta desde la placa Arduino One tras la recepción por comunicación serial de un aviso que lo indique [25].

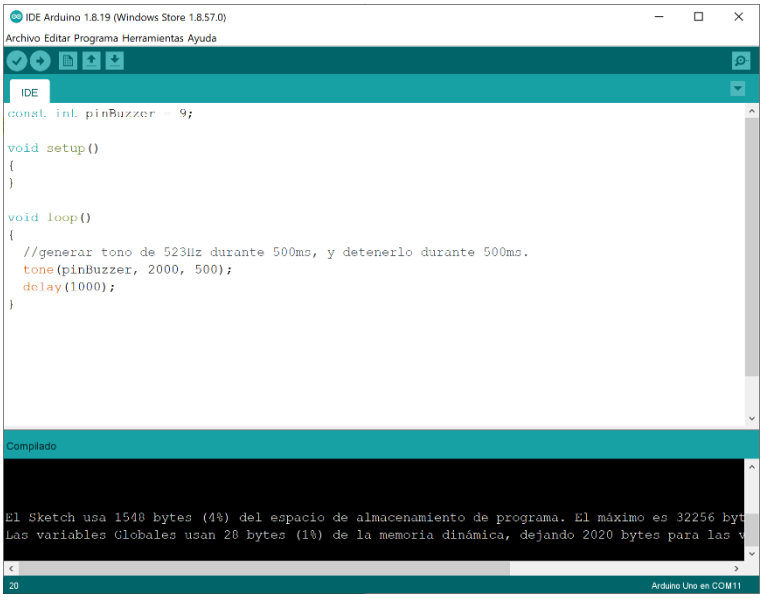


Fig. 4.9 Programación de alarma

4.2 Montaje físico y ensamblado del código

Contando con todos los códigos, y sabiendo la estructura física y digital en que se engloba el proyecto, se puede iniciar con el montaje. La maqueta, una estructura a escala de una vivienda real, construida con tablero de contrachapado, será la pieza fundamental, requiriéndose el despliegue de los dispositivos por su interior de forma estética pero fácilmente accesible. Habiendo un cajón construido bajo la base de la maqueta, todos los elementos de tamaño mayor serán estructurados sobre una lámina de MDF, Fig. 4.10, facilitando la disposición de cables y el conexionado. En línea con la disposición de cables, un panel con interruptores se monta sobre una de las tapas traseras de la maqueta, Fig. 4.11, sirviendo para la habilitación y deshabilitación de todos los dispositivos, al tiempo que se usa como pasacables hacia la pequeña electrónica. Esta pequeña electrónica está repartida por toda la construcción, habiéndose tratado de hacer la labor más estética y limpia posible, quedando la mayoría de los elementos empotrados sobre falsos techos en todas las estancias. Dando el apartado físico por terminado, la única labor pendiente es el ensamblado de todos los códigos bajo dos programas que sean capaces de trabajar a conjunto [26].

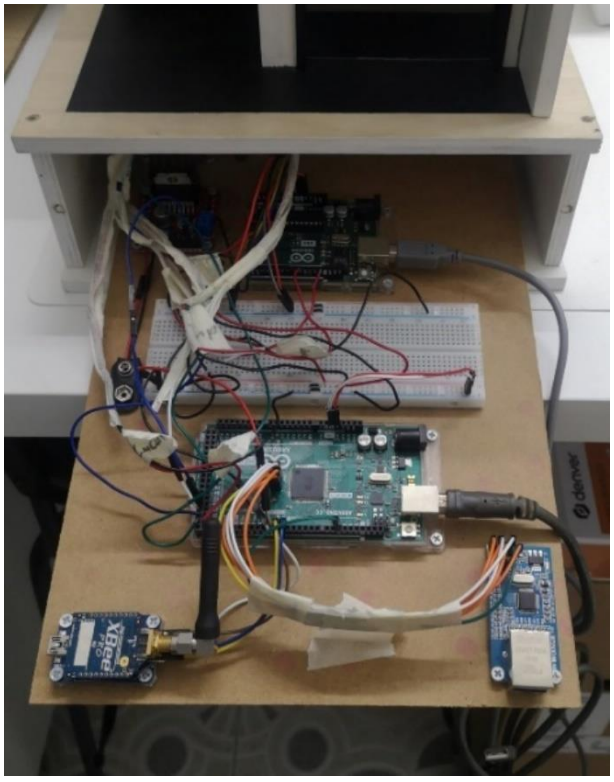


Fig. 4.10 Elementos mayores del sistema

Fig. 4.11 Panel trasero de la maqueta

Arduino Mega, siendo el controlador central, cuenta con la mayoría del código importante. La declaración de la MIB, mostrada en la Fig. 4.12, con variables definidas para la asignación de los OID y variables definidas para los valores de esos elementos, abre el código, dando pie a numerosas líneas para la propia implementación de las funcionalidades GETNEXT y GET de SNMP, en ese orden. Posteriormente, comienzan las funciones especificas de cada sistema del hogar. El sistema de temperatura es el primero, realizándose la lectura, comprobación, comunicación con actuadores, y actualización de valor de la MIB, todo seguido. El sistema de movimiento, algo más sencillo, realiza la lectura, comprueba el estado del interruptor manual de la alarma, y en funcion de su valor enciende un sistema luminoso o el sistema completo de alarma, realizándose también la actualización de valores en la MIB de todos los dispositivos afectados. Ambas funciones son llamadas desde el bucle central del programa, habiendo poco tiempo entre lectura y lectura para aportar respuestas de la forma más inmediata posible.

```
// MIB OIDs
const char propietarioCasa[] PROGMEM = "1.3.6.1.3.1.1.1.0";
const char localizacionCasa[] PROGMEM = "1.3.6.1.3.1.1.2.0";
const char tamanoCasa[] PROGMEM = "1.3.6.1.3.1.1.3.0";
const char plantasCasa[] PROGMEM = "1.3.6.1.3.1.1.4.0";
const char modeloMega[] PROGMEM = "1.3.6.1.3.1.2.1.1.0";
const char localizacionMega[] PROGMEM = "1.3.6.1.3.1.2.1.2.0";
const char ethernetMega[] PROGMEM = "1.3.6.1.3.1.2.1.3.1.0";
const char zigbeeMega[] PROGMEM = "1.3.6.1.3.1.2.1.3.2.0";
const char temperaturaMega[] PROGMEM = "1.3.6.1.3.1.2.2.1.0";
const char movimientoMega[] PROGMEM = "1.3.6.1.3.1.2.2.2.0";
const char iluminacionMega[] PROGMEM = "1.3.6.1.3.1.2.3.1.0";
const char ventiladorMega[] PROGMEM = "1.3.6.1.3.1.2.3.2.0";
const char chimeneaMega[] PROGMEM = "1.3.6.1.3.1.2.3.3.0";
const char modeloOne[] PROGMEM = "1.3.6.1.3.1.3.1.1.0";
const char localizacionOne[] PROGMEM = "1.3.6.1.3.1.3.1.2.0";
const char zigbeeOne[] PROGMEM = "1.3.6.1.3.1.3.1.3.1.0";
const char iluminacionOne[] PROGMEM = "1.3.6.1.3.1.3.2.1.0";
const char alarmaOne[] PROGMEM = "1.3.6.1.3.1.3.2.2.0";

// Local values
static char propietarioCasaV[] = "Sergio";
static char localizacionCasaV[] = "Pielagos";
static int32_t tamanoCasaV = 140;
static int32_t plantasCasaV = 2;
static char modeloMegaV[] = "Arduino Mega 2560 Rev3";
static char localizacionMegaV[] = "Primera planta";
static char ethernetMegaV[] = "Modulo W5500";
static char zigbeeMegaV[] = "Modulo Xbee Pro S1";
static int32_t temperaturaMegaV = 0;
static int32_t movimientoMegaV = 0;
static int32_t iluminacionMegaV = 0;
static int32_t ventiladorMegaV = 0;
static int32_t chimeneaMegaV = 0;
static char modeloOneV[] = "Arduino ONE Rev3";
static char localizacionOneV[] = "Segunda planta";
static char zigbeeOneV[] = "Modulo Xbee Pro S1";
static int32_t iluminacionOneV = 0;
static int32_t alarmaOneV = 0;
```

Fig. 4.12 Código para la creación de una MIB

Arduino One, con mucha menos carga, no cuenta con ninguna implementación SNMP, pues el controlador central se encarga por él, y se limita a controlar actuadores. Configurado para mantenerse escuchando el puerto serie de manera continua, la recepción de mensajes se utiliza para la activación de sus actuadores, estableciendo como mensaje de referencia aquel que Arduino Mega manda ante los valores de los sensores. Simple ante la dimensión del proyecto, las funciones de esta placa Arduino pueden escalarse notablemente, tanto en carga del dispositivo, como en número de ellos que se podían desplegar, pues el programa utilizado en este caso ocupa una minúscula parte de la memoria de esta Arduino One.

5. PRUEBAS DE FUNCIONAMIENTO

Montado el sistema, la única incógnita restante es su funcionamiento. Con las dos placas Arduino conectadas directamente a corriente, y el módulo Ethernet conectado al router, el sistema se inicia sin problema. Tomando el sistema la dirección ip 192.168.0.100, previamente asignada en función de las direcciones libres con que cuenta el router, como se muestra en la Fig. 5.1, es posible hacer ping sobre la maqueta, recibiendo correctamente respuesta.

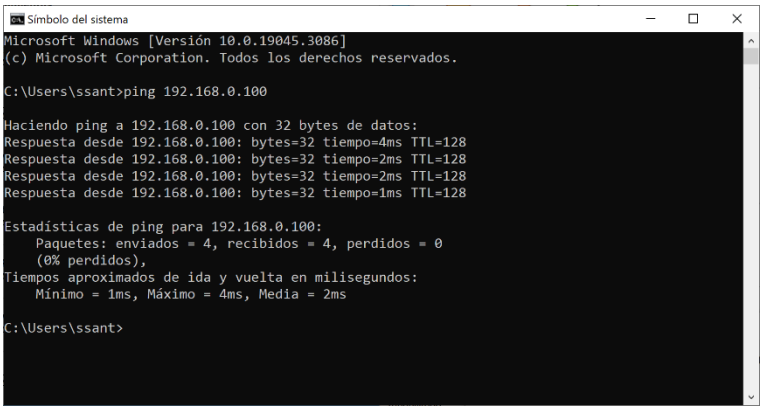


Fig. 5.1 Resultados del comando ping

Sabiendo que hay conexión, un MIB Browser será la herramienta empleada para contar con una representación comprensible de las tramas SNMP generadas por la maqueta. Iniciando con lecturas de datos, el usuario desde su ordenador realizará comandos GET y GETNEXT, pudiendo estos ser mandados desde su MIB Browser o desde una ventana de terminal.

En el caso de utilizar un MIB Browser, la aplicación empleada para comprobar el funcionamiento del sistema es iREASONING MIB Browser, una herramienta gratuita que ofrece una interfaz sobre la que administrar diferentes MIBs y monitorizar comandos SNMP. Rellenando los campos de dirección asignada para el sistema y el OID del elemento del que se desea conocer valor, las respuestas son prácticamente inmediatas, la aplicación retorna la información solicitada sin problema. Realizando una primera vuelta a la MIB completa, empleando comandos GETNEXT, como se puede ver en la Fig. 5.2, se reciben valores para los objetos definidos. Algunos, con carácter descriptivo, cuentan ya con los textos o números estáticos introducidos en el sistema, mientras que otros, más dinámicos, dan valores de estado, estando a 0 con todos los sistemas apagados, pero viéndose que hay parámetros como la temperatura que demuestran que el sistema funciona [27].

Name/OID	Value	Type	IP:Port
tamanoCasa.0	140	Integer	192.168....
plantasCasa.0	2	Integer	192.168....
temperaturaMega.0	20	Integer	192.168....
modeloMega.0	Arduino Mega 2560 Rev3	OctetStri...	192.168....
modeloOne.0	Arduino ONE Rev3	OctetStri...	192.168....
ethernetMega.0	Modulo W5500	OctetStri...	192.168....
zigbeeMega.0	Modulo Xbee Pro S1	OctetStri...	192.168....
zigbeeOne.0	Modulo Xbee Pro S1	OctetStri...	192.168....
localizacionCasa.0	Pielagos	OctetStri...	192.168....
localizacionMega.0	Primera planta	OctetStri...	192.168....
localizacionOne.0	Segunda planta	OctetStri...	192.168....
propietarioCasa.0	Sergio	OctetStri...	192.168....
movimientoMega.0	off (0)	Integer	192.168....
iluminacionMega.0	off (0)	Integer	192.168....
ventiladorMega.0	off (0)	Integer	192.168....
chimeneaMega.0	off (0)	Integer	192.168....
iluminacionOne.0	off (0)	Integer	192.168....
alarmaOne.0	off (0)	Integer	192.168....

Fig. 5.2 Resultados MIB Browser

Cambiando hacia una visión algo menos comercial, la opción más plana para monitorizar el sistema seria la utilización de comandos SNMP sobre una consola. Realizados sobre una terminal de Linux, en la Fig. 5.3 se muestran algunos comandos realizados sobre la maqueta. Siendo comandos, todos los parámetros que una aplicación asume es necesario incluirlos. La versión del protocolo, o la comunidad trabajada son elementos de los que no se ha hablado, pues quedaban definidos con la propia implementación de la librería Agentuino, pero haciendo las llamadas desde una terminal son parámetros clave a tener en cuenta. De nuevo, resultados numéricos o descriptivos se muestran correctamente, sin contarse esta vez con las informaciones extras aportadas por el MIB Browser.

```
student@student-sir:~$ snmpget -v 1 -c public -r 1 192.168.0.100 1.3.6.1.3.1.2.2.1.0
iso.3.6.1.3.1.2.2.1.0 = INTEGER: 21
student@student-sir:~$ snmpget -v 1 -c public -r 1 192.168.0.100 1.3.6.1.3.1.3.2.1.0
iso.3.6.1.3.1.3.2.1.0 = INTEGER: 0
student@student-sir:~$ snmpget -v 1 -c public -r 1 192.168.0.100 1.3.6.1.3.1.2.1.1.0
iso.3.6.1.3.1.2.1.1.0 = STRING: "Arduino Mega 2560 Rev3"
```

Fig. 5.3 Resultados por consola

Conociendo que el sistema está correctamente conectado a la red doméstica, y que SNMP está operativo, lo idóneo es analizar en más profundidad qué sucede a través de la red. Utilizando el analizador de protocolos Wireshark como principal herramienta, se van a escuchar desde el ordenador encargado todos los intercambios existentes ante la petición de información. Mostrando el intercambio en la Fig. 5.4, comandos GET y GETNEXT, junto a sus respectivas respuestas son captados por el analizador. Siendo el ordenador el dispositivo con dirección terminada en 41, y la maqueta la terminada en 100, mensajes de ida y vuelta se intercalan con las preguntas y respuestas [28].

No.	Time	Source	Destination	Protocol	Length	Info
467	40.325977	192.168.0.41	192.168.0.100	SNMP	85	get-request 1.3.6.1.3.1.1.1.0
468	40.329483	192.168.0.100	192.168.0.41	SNMP	97	get-response 1.3.6.1.3.1.1.1.0
580	49.643286	192.168.0.41	192.168.0.100	SNMP	87	get-request 1.3.6.1.3.1.2.1.3.2.0
581	49.645561	192.168.0.100	192.168.0.41	SNMP	111	get-response 1.3.6.1.3.1.2.1.3.2.0
930	62.083713	192.168.0.41	192.168.0.100	SNMP	86	get-next-request 1.3.6.1.3.1.3.1.2.0
931	62.087697	192.168.0.100	192.168.0.41	SNMP	111	get-response 1.3.6.1.3.1.3.1.3.1.0
1078	71.316025	192.168.0.41	192.168.0.100	SNMP	87	get-next-request 1.3.6.1.3.1.2.1.3.2.0
1079	71.319499	192.168.0.100	192.168.0.41	SNMP	96	get-response 1.3.6.1.3.1.2.2.1.0

Fig. 5.4 Captura de Wireshark

Sobre un GET concreto, mostrado en la Fig. 5.5, es posible realizar una lectura más exhaustiva del contenido. Viendo la pila de protocolos, Ethernet y UDP se presentan como la base de todos los mensajes, quedando SNMP por encima de ellos en referencia a la pila OSI. Información respecto al puerto 161, encargado de SNMP, o datos del protocolo como la versión y la comunidad se reflejan también ante esta visión, infinitamente más completa que todo lo visto hasta ahora.

```
> Frame 580: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface \Device\NPF_{B1281357-433B-444A-B452-F7EC5BAFF419}, id 0
v Ethernet II, Src: HuaweiTe_b6:95:c6 (a4:9b:4f:b6:95:c6), Dst: de:ad:be:ef:fe:ed (de:ad:be:ef:fe:ed)
  > Destination: de:ad:be:ef:fe:ed (de:ad:be:ef:fe:ed)
  > Source: HuaweiTe_b6:95:c6 (a4:9b:4f:b6:95:c6)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 192.168.0.41, Dst: 192.168.0.100
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 73
    Identification: 0x3376 (13174)
  > Flags: 0x00
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: UDP (17)
    Header Checksum: 0x8550 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.0.41
    Destination Address: 192.168.0.100
v User Datagram Protocol, Src Port: 60481, Dst Port: 161
    Source Port: 60481
    Destination Port: 161
    Length: 53
    Checksum: 0x8104 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 35]
  > [Timestamps]
    UDP payload (45 bytes)
v Simple Network Management Protocol
    version: version-1 (0)
    community: public
  > data: get-request (0)
    [Response In: 581]
```

Fig. 5.5 Protocolos en Wireshark

Siguiendo con Wireshark, pero con las respuestas, la estructura se mantiene, verificándose el correcto funcionamiento del código implementado, siendo el apartado más importante el de datos. Recogiendo una de las respuestas a un GET, mostrada en la Fig. 5.6, el campo variable-bindings muestra la información solicitada junto a su valor, demostrando que en profundidad el sistema realiza su labor.

```
data: get-response (2)
  get-response
      request-id: 1669836462
      error-status: noError (0)
      error-index: 0
    variable-bindings: 1 item
      1.3.6.1.3.1.2.1.3.2.0: "Modulo Xbee Pro S1"
```

Fig. 5.6 Campo datos de Wireshark

Con todo el desglose más técnico operando, es necesario poner a prueba los elementos finales, y la forma en que estos se relacionan con el entorno. Más visuales, pero más difíciles de comentar, los sensores y actuadores operan en base a las exigencias programadas en el código. Luces se encienden, alarmas suenan, y modificaciones en la temperatura desencadenan procesos informativos. Esto último, con especial interés, es uno de los aspectos ante los que se han programado traps, notificaciones que el agente en la maqueta manda a su gestor, y que sí son fácilmente revisables. Partiendo de la temperatura ambiente, no muy alta, se lleva a la maqueta hasta los 25 ºC, punto en que el sistema de ventilación tiene ordenes de ejecutarse, y como se ve en la Fig. 5.8, punto en que el sistema avisa de esta ejecución.

Source:	192.168.0.100	**Timestamp:**	0 millisecond	**SNMP Version:**	1
Enterprise:	domoCasaObjs			**Community:**	public
Specific:	3	**Generic:**	enterpriseSpecific		
Variable Bindings:					
Name:	ventiladorMega				
Value:	[Integer] on (1)				
Description:					

Fig. 5.8 Recepción de una trap

Visible igualmente a nivel protocolo, en la Fig. 5.7 Wireshark muestra los campos específicos de la trap. Campos genéricos como versión o comunidad aparecen de nuevo, pero junto a otros específicos de las traps. Dirección, tipo de trap, u el objeto al que hace referencia y su valor, son algunos de los campos relevantes para este tipo de mensajes, existiendo 4 tipos diferentes de traps que se pueden recibir de esta maqueta, con las activaciones y desactivaciones de los sistemas de climatización.

```
Simple Network Management Protocol
    version: version-1 (0)
    community: public
  data: trap (4)
    trap
        enterprise: 1.3.6.1.3.1 (iso.3.6.1.3.1)
        agent-addr: 192.168.0.100
        generic-trap: enterpriseSpecific (6)
        specific-trap: 3
        time-stamp: 0
      variable-bindings: 1 item
        1.3.6.1.3.1.2.3.2: 0
            Object Name: 1.3.6.1.3.1.2.3.2 (iso.3.6.1.3.1.2.3.2)
            Value (Integer32): 0
```

Fig. 5.7 SNMP sobre una trap

6. CONCLUSIONES Y LÍNEAS FUTURAS

Una vez finalizado el proyecto, y tras haber experimentado de primera mano los desafíos y soluciones encontrados al implementar SNMP en plataformas de bajas prestaciones, se podría resumir de manera coloquial diciendo que ha sido como matar moscas a cañonazos. SNMP es una arquitectura impecable, y la forma en que se trabaja con el protocolo y sus tramas es muy intuitiva. Siendo uno de los principales focos de estudio de la asignatura "Gestión y operación de redes", del plan de estudios del tercer año de la mención de telemática, SNMP aporta una visión muy didáctica de lo que es realmente la gestión de red, y de la necesidad de poder controlar y monitorizar multitud de dispositivos sin tener que amoldarse específicamente a cada uno de ellos. Aunque existen otros protocolos más modernos y livianos diseñados para dispositivos de capacidad limitada, SNMP sigue siendo un punto de partida válido. No obstante, la comunidad de Arduino carece de soluciones reales para una implementación sencilla de SNMP. Por tanto, esto representa una oportunidad excepcional para futuros proyectos que busquen desarrollar soluciones SNMP que aborden las deficiencias de las librerías disponibles en la actualidad. Como usuario de Arduino que soy desde hace tiempo para la implementación de pequeños sistemas en casa, existen múltiples soluciones independientes para la gestión remota de dispositivos, desde aplicaciones móviles que dan acceso a las placas de desarrollo, hasta servidores en web que permiten alojar información de sensores y actuadores. Pero se retorna al paradigma en que se han encontrado las redes desde su origen, soluciones heterogéneas e imposibles de interconectar.

La arquitectura SNMP fue diseñada específicamente para abordar la falta de interoperabilidad, asumiendo el papel de una solución estándar para la gestión de dispositivos, proporcionando un enfoque unificado y consistente. Sin embargo, llevar a cabo su implementación en la plataforma de Arduino, en lugar de recurrir a otras soluciones independientes, ha resultado en un proceso con desafíos frustrantes en varias ocasiones. No obstante, es importante destacar que el resultado final ha sido sumamente gratificante. Si bien las ideas iniciales estuvieron presentes durante el desarrollo de todo el proyecto, las soluciones iban siendo más puntuales. Numerosas librerías que prometían grandes funcionalidades de SNMP se fueron descartando a medida que se verificaba que no eran más que intentos de solución, y la forma en que sensores y actuadores debían introducirse en el código iba variando en base a las prestaciones con que se contaba. Con la estructura de SNMP terminada, es desarrollo del resto del proyecto fue en gran medida una labor de prueba y error. Sin haberse llegado a realizar el montaje sobre la maqueta, la primera aproximación al sistema fue sobre una placa de prototipos repleta de luces led, donde cada evento se reflejaba con encendidos y apagados, habiéndose hecho en ese punto un análisis intensivo de las capacidades de los sensores para poder afinar lo máximo posible el sistema. Con todo probado y funcionando el último paso fue la instalación sobre la maqueta, que podía haber sido infinitamente más simple, pero que con los medios adecuados realmente ha sido una labor bastante liviana.

Concluyendo de manera breve, me atrevería a decir que para mí este es un proyecto que no ha terminado, pues una maqueta es realmente una concepción muy pequeña de lo que un sistema puede llegar a ser, y habiéndose hablado de muchas otras tecnologías a lo largo de este documento, me queda la espinita de poner a pruebas todas ellas sobre implementaciones reales. La electrónica que hay tras estos sistemas es relativamente barata, y su reducido tamaño permite esconderla fácilmente en las cajas de registro con que cuenta una vivienda habitual, por lo que algún sistema pequeño puede ser abordable hasta para el mayor desconocedor de Arduino. En mi caso, y ante la gran cantidad de modificaciones que se puede realizar tanto sobre la parte de

protocolos como sobre el hardware, este proyecto se ha convertido en un pasatiempo que planeo mantener, y de aquí a unos años, si mis estudios continúan como hasta ahora, igual se dé una segunda versión de este proyecto desde otro punto de vista o bajo una nueva aplicación.

BIBLIOGRAFÍA

[1] "¿Qué es IoT? - Explicación del Internet de las cosas - AWS". Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/iot/#:~:text=The%20term%20IoT,%20or%20Internet,as%20between%20the%20devices%20themselves. (accedido el 17 de julio de 2023).

[2] "🏠 ¿Qué es la domótica? ¿Para qué sirve? ¿Qué opciones tengo?" DOMOTICA. https://domotica-casa.com/que-es-la-domotica-o-hogar-inteligente/ (accedido el 17 de julio de 2023).

[3] "Internet of Things (IoT): A Literature Review". SCIRP Open Access. https://www.scirp.org/html/56616_56616.htm (accedido el 17 de julio de 2023).

[4] "Arduino - Home". Arduino - Home. https://www.arduino.cc/ (accedido el 17 de julio de 2023).

[5] "AZ-Delivery: su experto en microelectrónica". AZ-Delivery. https://www.az-delivery.de/es (accedido el 17 de julio de 2023).

[6] "ELEGOO Deals". ELEGOO Official. https://www.elegoo.com/en-es/pages/deals?gclid=Cj0KCQjwzdOlBhCNARIsAPMwjbz89eiU1W6_3_QfhWO4Yj7LQTK3LEsVYQJStnkbyIf4Vrn36dcb6TsaAkwuEALw_wcB (accedido el 17 de julio de 2023).

[7] "Teach, learn, and make with the Raspberry Pi Foundation". Raspberry Pi Foundation. https://www.raspberrypi.org/ (accedido el 17 de julio de 2023).

[8] "Orange Pi - Orange Pi official website - Orange Pi development board, open source hardware, open source software, open source chip, computer keyboard". Orange Pi - Orange Pi official website - Orange Pi development board, open source hardware, open source software, open source chip, computer keyboard. http://www.orangepi.org/ (accedido el 17 de julio de 2023).

[9] "Banana Pi open source hardware community, Single board computer, Router,IoT,STEM education". Banana Pi open source hardware community,Single board computer, Router,IoT,STEM education. https://www.banana-pi.org/ (accedido el 17 de julio de 2023).

[10] "Sensores y Actuadores". Aprendiendo Arduino. https://aprendiendoarduino.wordpress.com/2016/12/18/sensores-y-actuadores/ (accedido el 17 de julio de 2023).

[11] "Domótica: Protocolos". enerxia.net. https://www.enerxia.net/portal/index.php/i-domo/894-domotica-protocolos (accedido el 17 de julio de 2023).

[12] Equipo editorial de IONOS. "¿Qué es Ethernet (IEEE 802.3)?" IONOS Digital Guide. https://www.ionos.es/digitalguide/servidores/know-how/ethernet-ieee-8023/ (accedido el 17 de julio de 2023).

[13] "A FONDO: ZIGBEE". Todo en domótica de fácil instalación. https://www.domodesk.com/216-a-fondo-zigbee.html (accedido el 17 de julio de 2023).

[14] "Qué es el Bluetooth y todo lo que permite esta tecnología". ADSLZone. https://www.adslzone.net/reportajes/tecnologia/bluetooth/ (accedido el 17 de julio de 2023).

[15] "Descubra los estándares WiFi, incluido el último WiFi 6 (802.11ax)". NetSpot. https://www.netspotapp.com/es/blog/wifi-standards/ (accedido el 17 de julio de 2023).

[16] "IBM Documentation". IBM - Deutschland | IBM. https://www.ibm.com/docs/es/netcoolomnibus/8.1?topic=manager-about-snmp (accedido el 17 de julio de 2023).

[17] https://forum.huawei.com/enterprise/es/introducción-al-protocolo-de-comunicación-netconf-utilizado-en-switches-cloudengine/thread/667222530408202240-667212883219591168 (accedido el 17 de julio de 2023).

[18] https://es.linkedin.com/pulse/yang-y-netconf-la-pareja-perfecta-para-gestión (accedido el 17 de julio de 2023).

[19] "¿Qué es el MQTT? - Explicación del protocolo MQTT - AWS". Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/mqtt/ (accedido el 17 de julio de 2023).

[20] https://xbee.cl/que-es-xbee/ (accedido el 17 de julio de 2023).

[21] "Módulo Ethernet W5500". ArduinoVe. https://www.arduinove.com/index.php?route=product/product&product_id=87 (accedido el 17 de julio de 2023).

[22] "GitHub - 1sw/Agentuino: Arduino Agentuino library expanded of SNMP GET-NEXT-Request." GitHub. https://github.com/1sw/Agentuino (accedido el 17 de julio de 2023).

[23] "Cómo utilizar el DHT11 para medir la temperatura y humedad con Arduino". Programar fácil con Arduino. https://programarfacil.com/blog/arduino-blog/sensor-dht11-temperatura-humedad-arduino/ (accedido el 17 de julio de 2023).

[24] "Sensor HC-SR501 con Arduino. – Electrónica Práctica Aplicada". diarioelectronicohoy desde 2002 periódico técnico electrónica profesional. https://www.diarioelectronicohoy.com/blog/sensor-hc-sr501-con-arduino (accedido el 17 de julio de 2023).

[25] "Práctica 21. Generar sonidos con un piezoeléctrico". MECABOT. http://mecabot-ula.org/tutoriales/arduino/practica-21-generar-sonidos-con-un-piezoelectrico/ (accedido el 17 de julio de 2023).

[26] "GitHub - ssa898/DOMO-CASA". GitHub. https://github.com/ssa898/DOMO-CASA (accedido el 17 de julio de 2023).

[27] "MIB browser". MIB Browser. https://ireasoning.com/mibbrowser.shtml (accedido el 17 de julio de 2023).

[28] "Wireshark · Download". Wireshark. https://www.wireshark.org/download.html (accedido el 17 de julio de 2023).

ANEXOS

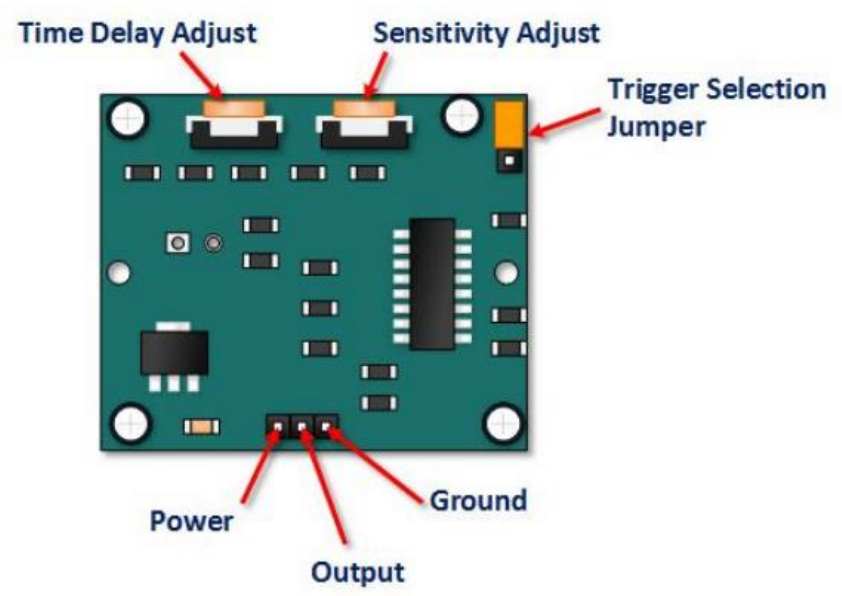
Módulo sensor HC-SR501:

HC-SR501 Passive Infrared (PIR) Motion Sensor

This motion sensor module uses the LHI778 Passive Infrared Sensor and the BISS0001 IC to control how motion is detected.

The module features adjustable sensitivity that allows for a motion detection range from 3 meters to 7 meters, also includes time delay adjustments and trigger selection that allow for fine tuning within your application.

HC-SR501 Pin Outs and Controls



Pin or Control	Function
Time Delay Adjust	Sets how long the output remains high after detecting motion.... Anywhere from 5 seconds to 5 minutes.
Sensitivity Adjust	Sets the detection range.... from 3 meters to 7 meters
Trigger Selection Jumper	Set for single or repeatable triggers.
Ground pin	Ground input
Output Pin	Low when no motion is detected.. High when motion is detected. High is 3.3V
Power Pin	5 to 20 VDC Supply input

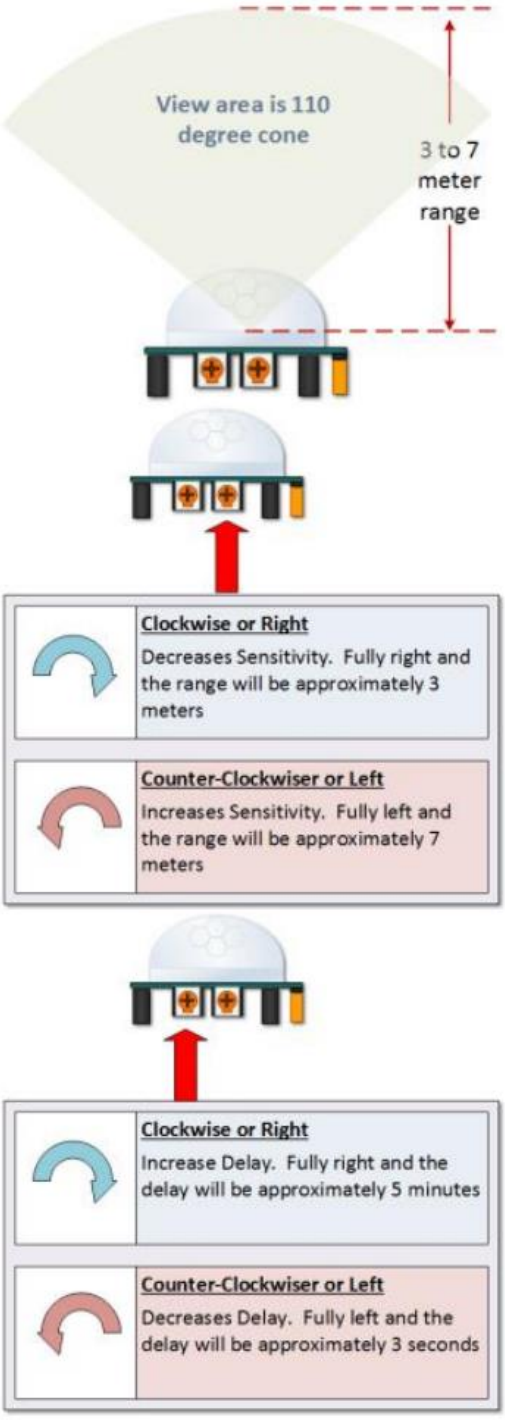
HC SR501 PIR Functional Description

The SR501 will detect infrared changes and if interpreted as motion, will set its output low.
What is or is not interpreted as motion is largely dependent on user settings and adjustments.

Device Initialization

The device requires nearly a minute to initialize.
During this period, it can and often will output false detection signals.
Circuit or controller logic needs to take this initialization period into consideration.

Device Area of Detection



The device will detect motion inside a 110 degree cone with a range of 3 to 7 meters.

PIR Range (Sensitivity) Adjustment

As mentioned, the adjustable range is from approximately 3 to 7 meters.

Time Delay Adjustment

The time delay adjustment determines how long the output of the PIR sensor module will remain high after detection motion. The range is from about 3 seconds to five minutes.

3 Seconds Off After Time Delay Completes – IMPORTANT

The output of this device will go LOW (or Off) for approximately 3 seconds AFTER the time delay completes. In other words, ALL motion detection is blocked during this three second period.

For Example:

- Imagine you're in the single trigger mode (see below) and your time delay is set 5 seconds.
  - The PIR will detect motion and set it high for 5 seconds.
  - After five seconds, the PIR will sets its output low for about 3 seconds.

- During the three seconds, the PIR will not detect motion.
- After three seconds, the PIR will detect motion again and detected motion will once again set the output high and the output will remain on as dictated by the Time Delay adjustment and trigger mode selection.

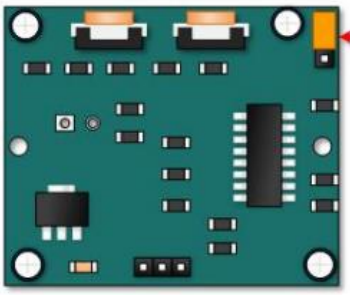
OVERRIDING THE TIME DELAY – If you're connecting your HC-SR501 to an Arduino, it is likely that you are going to take some sort of action when motion is detected. For example, you may wish to brighten lights when motion is detected and dim the lights when motion is no longer connected

Simply delay dimming within your sketch.

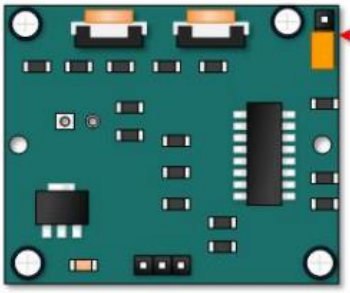
Trigger Mode Selection Jumper

The trigger mode selection jumper allows you to select between single and repeatable triggers. The affect of this jumper setting is to determine when the time delay begins.

- **SINGLE TRIGGER** – The time delay begins immediately when motion is first detected.
- **REPEATABLE TRIGGER** – Each detected motion resets the time delay. Thus the time delay begins with the last motion detected.



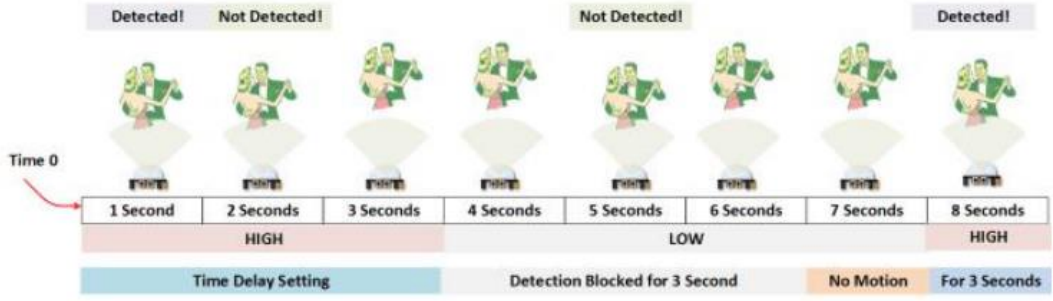
Single Trigger Mode – Time Delay is started immediately upon detecting motion. Continued detection is blocked



Repeatable Trigger Mode – Time Delay is re-started every time motion is detected.

HC-SR501 Dance Floor Application Examples
Imagine that you want to control lighting on a dance floor based upon where the dancers are dancing.
Understanding how the time delay and trigger mode interact will be necessary to controlling that lighting in the manner that you want.

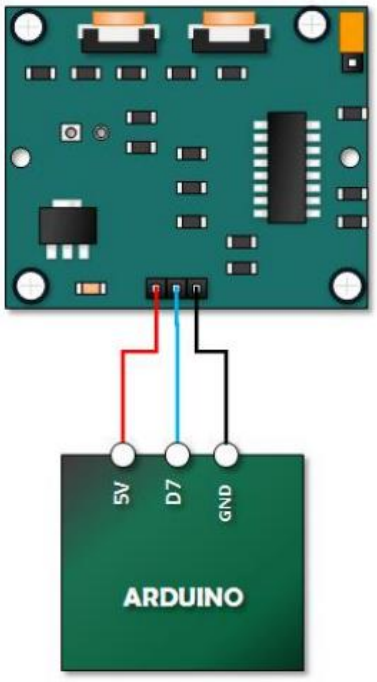
Example One
In this first example, the time delay is set to three seconds and the trigger mode is set to single.
As you can see in the illustration below, the motion is not always detected.
In fact, there is a period of about six seconds where motion can not be detected.



Example Two

In the next example, the time delay is still at three seconds and the trigger is set to repeatable. In the illustration below, you can see that the time delay period is restarted.
However, after that three seconds, detection will still be blocked for three seconds.As I mentioned previously, you could override the 3 second blocking period with some creative code, but do give that consideration.
Some of the electronics you use may not like an on and then off jolt. The three seconds allows for a little rest before starting back up.

Arduino HC-SR501 Motion Sensor Tutorial, connect Your Arduino to the HC-SR501



The sketch simply turns on Your Arduino LED connected to Pin 13 whenever motion is detected.
Be sure to beware of and somehow handle the 1 minute initialization in whatever application you develop.

```
// HC-SR501 Motion Detector Sample Sketch
int ledPin = 13;   // LED on Pin 13 of Arduino
int pirPin = 7; // Input for HC-S501
int pirValue; // Place to store read PIR Value
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(pirPin, INPUT);
   digitalWrite(ledPin, LOW);
 }
void loop() {
  pirValue = digitalRead(pirPin);
  digitalWrite(ledPin, pirValue);
}
```

Módulo sensor DHT11:

3. Typical Application (Figure 1)

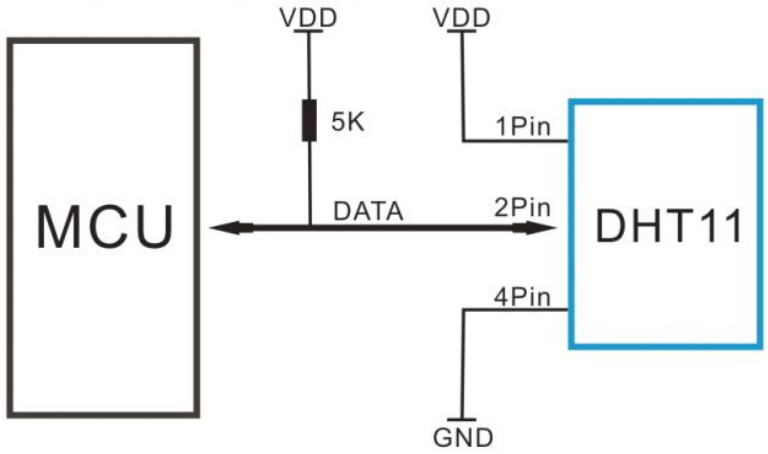


Figure 1 Typical Application

Note: 3Pin – Null; MCU = Micro-computer Unite or single chip Computer

When the connecting cable is shorter than 20 metres, a 5K pull-up resistor is recommended; when the connecting cable is longer than 20 metres, choose a appropriate pull-up resistor as needed.

4. Power and Pin

DHT11's power supply is 3-5.5V DC. When power is supplied to the sensor, do not send any instruction to the sensor in within one second in order to pass the unstable status. One capacitor valued 100nF can be added between VDD and GND for power filtering.

5. Communication Process: Serial Interface (Single-Wire Two-Way)

Single-bus data format is used for communication and synchronization between MCU and DHT11 sensor. One communication process is about 4ms.

Data consists of decimal and integral parts. A complete data transmission is **40bit**, and the sensor sends **higher data bit** first.
Data format: 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum. If the data transmission is right, the check-sum should be the last 8bit of "8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data".

5.1 Overall Communication Process (Figure 2, below)

When MCU sends a start signal, DHT11 changes from the low-power-consumption mode to the running-mode, waiting for MCU completing the start signal. Once it is completed, DHT11 sends a response signal of 40-bit data that include the relative humidity and temperature information to MCU. Users can choose to collect (read) some data. Without the start signal from MCU, DHT11 will not give the response signal to MCU. Once data is collected, DHT11 will change to the low-power-consumption mode until it receives a start signal from MCU again.

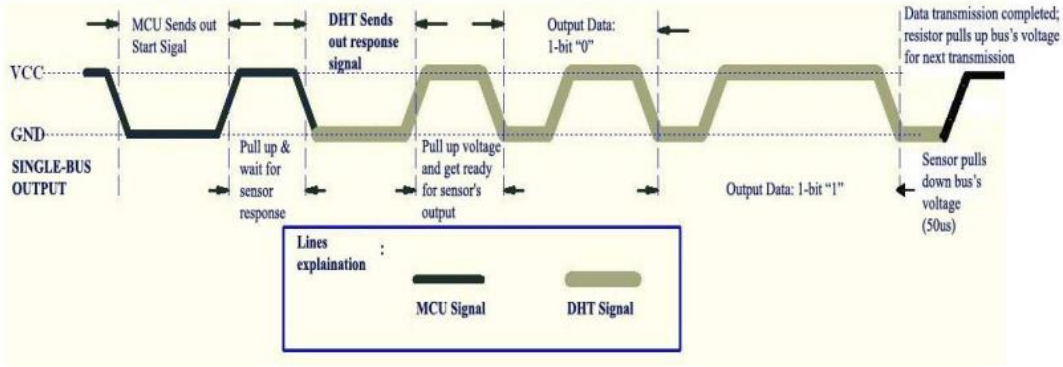


Figure 2 Overall Communication Process

5.2 MCU Sends out Start Signal to DHT (Figure 3, below)

Data Single-bus free status is at high voltage level. When the communication between MCU and DHT11 begins, the programme of MCU will set Data Single-bus voltage level from high to low and this process must take at least 18ms to ensure DHT's detection of MCU's signal, then MCU will pull up voltage and wait 20-40us for DHT's response.

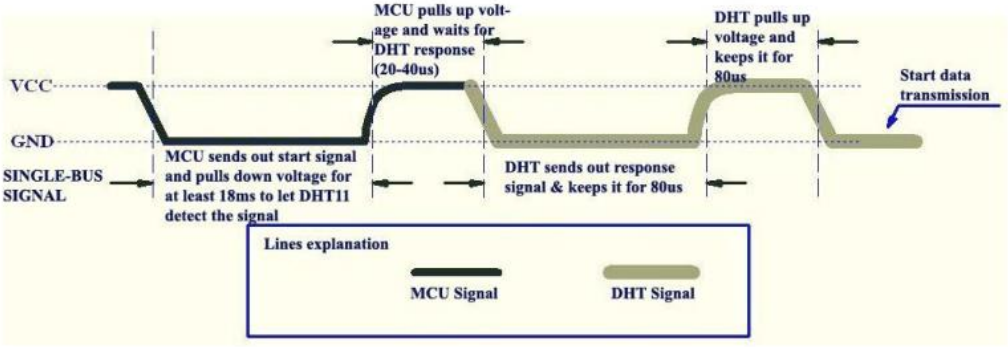


Figure 3 MCU Sends out Start Signal & DHT Responses

5.3 DHT Responses to MCU (Figure 3, above)

Once DHT detects the start signal, it will send out a low-voltage-level response signal, which lasts 80us. Then the programme of DHT sets Data Single-bus voltage level from low to high and keeps it for 80us for DHT's preparation for sending data.

When DATA Single-Bus is at the low voltage level, this means that DHT is sending the response signal. Once DHT sent out the response signal, it pulls up voltage and keeps it for 80us and prepares for data transmission.

When DHT is sending data to MCU, every bit of data begins with the 50us low-voltage-level and the length of the following high-voltage-level signal determines whether data bit is "0" or "1" (see Figures 4 and 5 below).

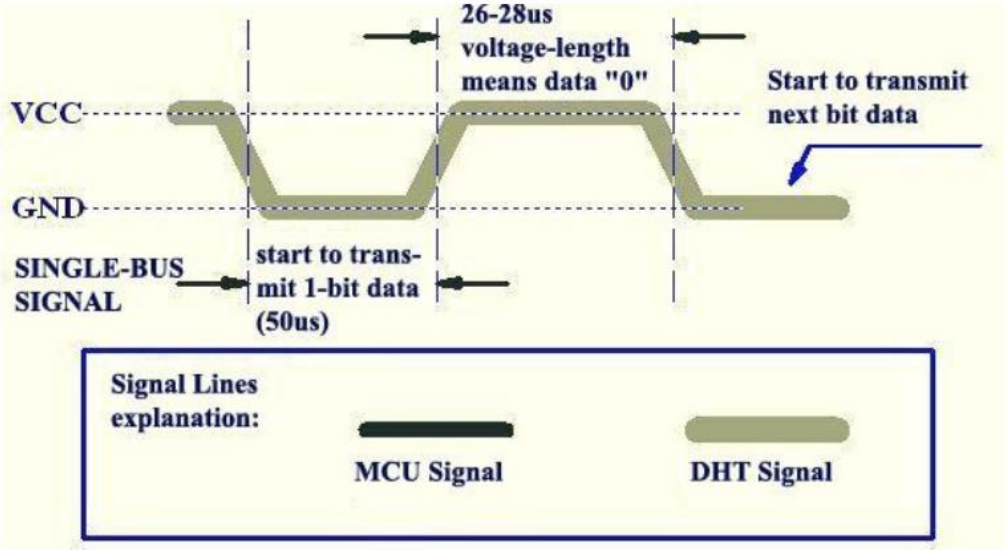


Figure 4 Data "0" Indication

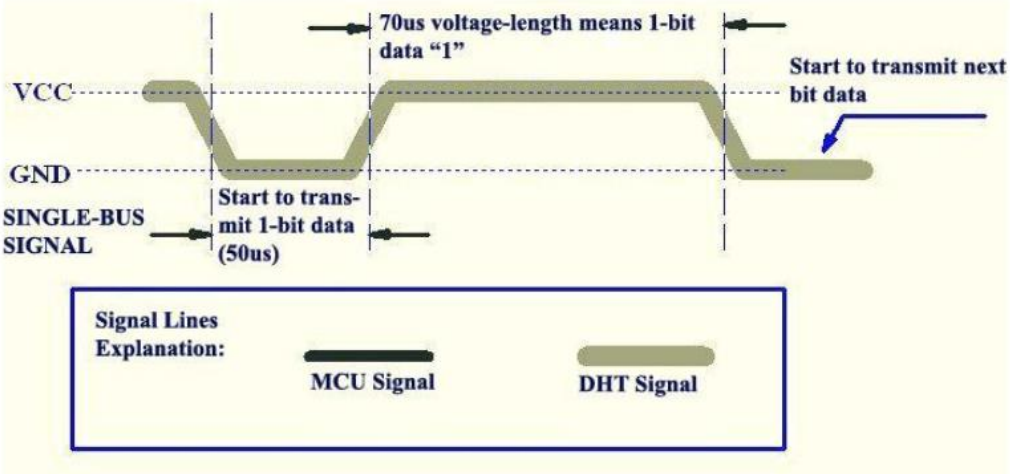


Figure 5 Data "1" Indication

If the response signal from DHT is always at high-voltage-level, it suggests that DHT is not responding properly and please check the connection. When the last bit data is transmitted, DHT11 pulls down the voltage level and keeps it for 50us. Then the Single-Bus voltage will be pulled up by the resistor to set it back to the free status.

6. Electrical Characteristics

VDD=5V, T = 25℃ (unless otherwise stated)

	Conditions	Minimum	Typical	Maximum
Power Supply	DC	3V	5V	5.5V
Current Supply	Measuring	0.5mA		2.5mA
	Average	0.2mA		1mA
	Standby	100uA		150uA
Sampling period	Second	1		

Note: Sampling period at intervals should be no less than 1 second.

7. Attentions of application

(1) Operating conditions

Applying the DHT11 sensor beyond its working range stated in this datasheet can result in 3%RH signal shift/discrepancy. The DHT11 sensor can recover to the calibrated status gradually when it gets back to the normal operating condition and works within its range. Please refer to (3) of

Codificación de la MIB:

```
--
-- DOMO-CASA-MIB-V1.my
-- MIB generated by MG-SOFT Visual MIB Builder 2018 (64-bit) Version 12.0  Build 1200
-- Wednesday, June 14, 2023 at 10:36:21
--
        DOMO-CASA-MIB-V1 DEFINITIONS ::= BEGIN
                IMPORTS
                        OBJECT-TYPE
                                FROM RFC-1212
                        experimental
                                FROM RFC1155-SMI;
--
-- Node definitions
--
                        -- 1.3.6.1.3.1
                        domoCasaObjs OBJECT IDENTIFIER ::= { experimental 1 }
                        -- 1.3.6.1.3.1.1
                        informacionCasa OBJECT IDENTIFIER ::= { domoCasaObjs 1 }
                        -- 1.3.6.1.3.1.1.1
                        propietarioCasa OBJECT-TYPE
                                SYNTAX OCTET STRING
                                ACCESS read-only
                                STATUS mandatory
                                DESCRIPTION
                                        "Informacion sobre el propietario de la vivienda."
                                ::= { informacionCasa 1 }
                        -- 1.3.6.1.3.1.1.2
                        localizacionCasa OBJECT-TYPE
                                SYNTAX OCTET STRING
                                ACCESS read-only
                                STATUS mandatory
                                DESCRIPTION
                                        "Informacion sobre la localizacion de la vivienda."
                                ::= { informacionCasa 2 }
                        -- 1.3.6.1.3.1.1.3
                        tamanoCasa OBJECT-TYPE
                                SYNTAX OCTET STRING
                                ACCESS read-only
                                STATUS mandatory
                                DESCRIPTION
                                        "Informacion sobre el tamano de la vivienda."
                                ::= { informacionCasa 3 }
                        -- 1.3.6.1.3.1.1.4
                        plantasCasa OBJECT-TYPE
                                SYNTAX OCTET STRING
                                ACCESS read-only
                                STATUS mandatory
                                DESCRIPTION
                                        "Informacion sobre el numero de plantas de la vivienda."
                                ::= { informacionCasa 4 }
                        -- 1.3.6.1.3.1.2
                        arduinoMega OBJECT IDENTIFIER ::= { domoCasaObjs 2 }
                        -- 1.3.6.1.3.1.2.1
                        infoMega OBJECT IDENTIFIER ::= { arduinoMega 1 }
                        -- 1.3.6.1.3.1.2.1.1
                        modeloMega OBJECT-TYPE
                                SYNTAX OCTET STRING
                                ACCESS read-only
                                STATUS mandatory
                                DESCRIPTION
                                        "Modelo de la placa."
                                ::= { infoMega 1 }
                        -- 1.3.6.1.3.1.2.1.2
                        localizacionMega OBJECT-TYPE
```

```
                SYNTAX OCTET STRING
                ACCESS read-only
                STATUS mandatory
                DESCRIPTION
                        "Localizacion de la placa."
                ::= { infoMega 2 }
-- 1.3.6.1.3.1.2.1.3
conectividadMega OBJECT IDENTIFIER ::= { infoMega 3 }
-- 1.3.6.1.3.1.2.1.3.1
ethernetMega OBJECT-TYPE
                SYNTAX OCTET STRING
                ACCESS read-only
                STATUS mandatory
                DESCRIPTION
                        "Modelo del modulo ethernet."
                ::= { conectividadMega 1 }
-- 1.3.6.1.3.1.2.1.3.2
zigbeeMega OBJECT-TYPE
                SYNTAX OCTET STRING
                ACCESS read-only
                STATUS mandatory
                DESCRIPTION
                        "Modelo del modulo zigbee."
                ::= { conectividadMega 2 }
-- 1.3.6.1.3.1.2.2
sensoresMega OBJECT IDENTIFIER ::= { arduinoMega 2 }
-- 1.3.6.1.3.1.2.2.1
temperaturaMega OBJECT-TYPE
                SYNTAX INTEGER
                ACCESS read-only
                STATUS mandatory
                DESCRIPTION
                        "Valor devuelto po rle sensor de temperatura."
                ::= { sensoresMega 1 }
-- 1.3.6.1.3.1.2.2.2
movimientoMega OBJECT-TYPE
                SYNTAX INTEGER
                        {
                        on(1),
                        off(0)
                        }
                ACCESS read-only
                STATUS mandatory
                DESCRIPTION
                        "Estado del sensor de movimiento."
                ::= { sensoresMega 2 }
-- 1.3.6.1.3.1.2.3
actuadoresMega OBJECT IDENTIFIER ::= { arduinoMega 3 }
-- 1.3.6.1.3.1.2.3.1
iluminacionMega OBJECT-TYPE
                SYNTAX INTEGER
                        {
                        on(1),
                        off(0)
                        }
                ACCESS read-only
                STATUS mandatory
                DESCRIPTION
                        "Estado de la iluminación."
                ::= { actuadoresMega 1 }
-- 1.3.6.1.3.1.2.3.2
ventiladorMega OBJECT-TYPE
                SYNTAX INTEGER
                        {
                        on(1),
                        off(0)
```

```
                    }
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                    "Estado del ventilador."
            ::= { actuadoresMega 2 }
-- 1.3.6.1.3.1.2.3.3
chimeneaMega OBJECT-TYPE
            SYNTAX INTEGER
                    {
                    on(1),
                    off(0)
                    }
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                    "Estado de la chimenea."
            ::= { actuadoresMega 3 }
-- 1.3.6.1.3.1.3
arduinoOne OBJECT IDENTIFIER ::= { domoCasaObjs 3 }
-- 1.3.6.1.3.1.3.1
infoOne OBJECT IDENTIFIER ::= { arduinoOne 1 }
-- 1.3.6.1.3.1.3.1.1
modeloOne OBJECT-TYPE
            SYNTAX OCTET STRING
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                    "Modelo de la placa."
            ::= { infoOne 1 }
-- 1.3.6.1.3.1.3.1.2
localizacionOne OBJECT-TYPE
            SYNTAX OCTET STRING
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                    "Localizacion de la placa."
            ::= { infoOne 2 }
-- 1.3.6.1.3.1.3.1.3
conectividadOne OBJECT IDENTIFIER ::= { infoOne 3 }
-- 1.3.6.1.3.1.3.1.3.1
zigbeeOne OBJECT-TYPE
            SYNTAX OCTET STRING
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                    "Modelo del modulo zigbee."
            ::= { conectividadOne 1 }
-- 1.3.6.1.3.1.3.2
actuadoresOne OBJECT IDENTIFIER ::= { arduinoOne 2 }
-- 1.3.6.1.3.1.3.2.1
iluminacionOne OBJECT-TYPE
            SYNTAX INTEGER
                    {
                    on(1),
                    off(0)
                    }
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                    "Estado de la iluminacion."
            ::= { actuadoresOne 1 }
-- 1.3.6.1.3.1.3.2.2
alarmaOne OBJECT-TYPE
            SYNTAX INTEGER
                    {
```

```
                                        on(1),
                                        off(0)
                                        }
                                ACCESS read-only
                                STATUS mandatory
                                DESCRIPTION
                                        "Estado de la alarma."
                                ::= { actuadoresOne 2 }
        END
--
-- DOMO-CASA-MIB-V1.my
--
```