UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño de un sistema de control domótico basado en la plataforma Arduino

Proyecto Final de Carrera

[Ingeniería Técnica en Informática de Sistemas]

Autor: Emilio Lledó Sánchez

Directores: Sergio Sáez Barona, Vicente Luis Atienza Vanacloig

Diciembre de 2012

Resumen

En esta memoria se han descrito los conocimientos básicos para entender que es y cómo funciona un sistema domótico y cómo utilizando el hardware libre de Arduino se puede crear un sistema estable con un presupuesto muy inferior al de las viviendas de alta categoría.

La memoria ha sido dividida en cinco fases en las que se van a tratar cada tema de una forma amplia pero simple, es decir, se dará la información necesaria para entender el proceso de creación de un sistema domótico sin dar detalles superfluos.

En la introducción se va a poder dar un primer paso en el mundo de la domótica y conoceremos el porqué de la utilización de Arduino.

Después aprenderemos cómo están construidas las placas Arduino y el entorno de trabajo que disponemos para programarla.

Es importante comentar también los distintos dispositivos que podemos acoplar a una placa para añadir funcionalidad al sistema domótico.

Dado que la parte de gestión del sistema es más compleja se hará hincapié en ella ofreciendo distintas formas de comunicación entre las placas argumentando sus ventajas e inconvenientes.

Acto seguido se expone un sistema domótico simple en el que se aplican los conocimientos adquiridos.

Toda la información de la memoria ha sido reforzada con imágenes para conocer de una forma visual sobre qué se está hablando.

Palabras clave: domótica, Arduino, hardware libre

Tabla de contenidos

1. Introducción y objetivos

Aunque el ser humano todavía no está arraigado a las propiedades que ofrece la domótica es un hecho que en un futuro estará instalada en cualquier vivienda. Pero ¿qué es la domótica? Se podría definir como el conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía además de aportar seguridad, confort, y comunicación entre el usuario y el sistema. Para poder conseguir las propiedades comentadas anteriormente es necesario que los sistemas recojan la información de su entorno con sensores y dispongan de la lógica para actuar en consecuencia utilizando actuadores.

Actualmente los sistemas domóticos tienen un precio muy alto de instalación con lo cual solo es posible verlo en casas de lujo. Estos suelen utilizar buses de transmisión de información que posibilitan una domótica robusta como son el EIB, X10, CEBus, LonWorks/LongTalk y ZigBee. Una alternativa más barata y casera consiste en la utilización de placas Arduino.

En este proyecto utilizaremos la plataforma Arduino en la que nos apoyaremos con otros dispositivos para poder construir un sistema domótico simple. Arduino es una plataforma de hardware libre creada en 2005, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

Para crear el sistema domótico han de tenerse en cuenta varios aspectos. Hay que conocer el capital que disponemos para invertir en el sistema y seleccionar los dispositivos que más se ajusten a nuestras necesidades. De poco sirve comprar un elemento con grandes prestaciones si luego no se va a aprovechar. Por ejemplo, en vez de utilizar como servidor del sistema un PC dedicado y muy simple que costase unos 150€ podríamos usar otras alternativas. Este año han aparecido alternativas muy baratas como la placa computadora Raspberry Pi que se puede obtener por unos 30€.

Luego está el factor estético que normalmente evitaría la instalación de cableado para comunicar las placas Arduino, es decir, aprovecharíamos dispositivos que trabajasen inalámbricamente. Aunque el precio de un dispositivo inalámbrico es ligeramente superior, podremos evitar tener que comprar cables que poco a poco aumentaría el coste total haciéndolo incluso más caro. Pero también debemos saber que los elementos inalámbricos interfieren entre sí y eso por ejemplo en una zona densamente habitada en el que los vecinos también dispongan de este tipo aparatos puede reducir las prestaciones de la comunicación del sistema. Es por ello que cada caso hay que estudiarlo por separado y actuar en consecuencia.

Como objetivo de este proyecto nos hemos propuesto crear un sistema domótico simple utilizando las placas de bajo coste Arduino y otros dispositivos, como sensores, actuadores y comunicadores. Habrá que dotar al sistema de la lógica necesaria para que puedan comunicarse las placas que estarán controlando la habitación en la cual hayan sido instaladas.

2. Características básicas de Arduino

En este apartado vamos a describir los principales elementos que componen una placa Arduino y el entorno de desarrollo en el que se programa el código, es decir la parte hardware y software que actúan sobre Arduino.

2.1 Hardware

Al ser Arduino una plataforma de hardware libre tanto su diseño como su distribución puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia. Por eso existen varios tipos de placa oficiales, las creadas por la comunidad Arduino o las no oficiales creadas por terceros pero con características similares. En la placa Arduino es donde conectaremos los sensores, actuadores y otros elementos necesarios para comunicarnos con el sistema.

En el proyecto se han utilizado las placas Arduino Uno y Seeeduino que describiremos a continuación.

2.1.1 Arduino Uno

Ilustración 1: Frontal y reverso de la placa Arduino Uno

Es el último modelo diseñado y distribuido por la comunidad Arduino. La placa tiene un tamaño de 75x53mm. Su unidad de procesamiento consiste en un microcontrolador ATmega328. Puede ser alimentada mediante USB o alimentación externa y contiene pines tanto analógicos como digitales. La tabla siguiente resume sus componentes:

Microcontrolador	ATmega328
Voltaje operativo	5V
Voltaje de entrada(recomendado)	7-12V
Voltaje de entrada (limites)	6-20V
Pines digitales E/S	14 (de los cuales 6 proporcionan salida PWM)
Pines de entrada analógica	6
Corriente continua para pines E/S	40 mA
Corriente continua para pines de 3.3V	50 mA
Memoria Flash	32 KB (ATmega328) de los cuales 0.5 KB son para el bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidad del reloj	16 MHz

A continuación se muestra en la Ilustración 2 donde están ubicados los elementos más importantes que componen la placa Arduino Uno que son descritos de arriba abajo y de izquierda a derecha:

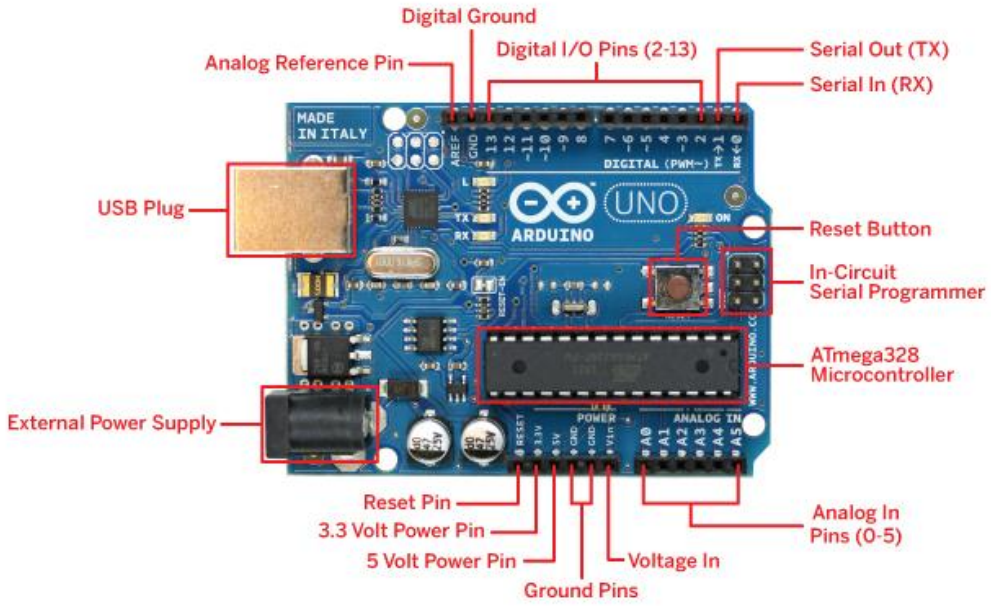

Ilustración 2: Elementos de la placa Arduino Uno

Referencia para pines analógicos (AREF)

Tensión de referencia para entradas analógicas. Se utiliza con la función *analogReference()*.

Pines de tierra (GND)

Masa del circuito para pines, es decir es la tensión de referencia de 0V.

Pines digitales de entrada y salida

En estos pines conectaremos la patilla de dato del sensor/actuador. Desde ellos podremos leer la información del sensor o activar el actuador.
Hay 14 pines digitales que pueden utilizarse como entrada o salida con las funciones *pinMode(), digitalWrite(), y digitalRead()*. Operan a 5 voltios. Cada pin proporciona o recibe como máximo 40mA y disponen de una resistencia pull-up (desconectada por defecto) de 20-50 kOhmios. Ciertos pines son reservados para determinados usos:

- Serie: 0(RX) y 1(TX). Utilizados para recibir (RX) y trasmitir (TX) datos serie. Están directamente conectados a los pines serie del microcontrolador. Utilizando estos pines podremos conectarnos con otras placas.
- Interrupciones externas: 2 y 3. Estos pines pueden ser configurados para activar interrupciones.
- PWM: 3, 5, 6, 9, 10 y 11. Proporcionan una salida de 8 bits en modo PWM.
- SPI: 10-13. Estos pines soportan la librería de comunicación de dispositivos SPI.
- LED: 13. Este pin está conectado con un led de la placa. Cuando se le asigne un valor HIGH se encenderá, en cambio si lo dejamos en LOW estará apagado.

Conector USB

Existen varios tipos de conectores USB, en concreto esta placa utiliza el tipo B hembra. Con lo cual se necesitará un cable tipo B macho – tipo A macho (aunque se pueden utilizar otros este es el más extendido) que deberá conectarse a un conector tipo A hembra (por ejemplo a un ordenador o al cargador de un móvil). La placa se puede alimentar con la tensión de 5V que le proporciona el bus serie USB.
Cuando carguemos un programa a la placa desde el software de Arduino se inyectará el código del ordenador por este bus.

Botón Reset

Utilizando este botón podremos reiniciar la ejecución del código del microcontrolador.

ICSP (In Circuit Serial Programming)

Es un conector utilizado en los dispositivos PIC para programarlos sin necesidad de tener que retirar el chip del circuito del que forma parte.

Microcontrolador ATmega328

El microcontrolador es el elemento más importante de la placa. Es donde se instalará y ejecutará el código que se haya diseñado. Ha sido creado por la compañía Atmel, tiene un voltaje operativo de 5V, aunque se recomienda como entrada de 7-12V con un límite de 20V. Contiene 14 pines digitales de entrada y salida, 6 pines analógicos que están conectados directamente a los pines de la placa Arduino comentados anteriormente. Dispone de 32KB de memoria flash (de los cuales 512 bytes son utilizados por el bootloader). En la memoria flash se instalará el programa a ejecutar. El bootloader será el encargado de preparar el microcontrolador para que pueda ejecutar nuestro programa. También tiene una memoria EEPROM de 1KB que puede ser leída o escrita con la librería EEPROM.
En la parte de procesamiento dispone de un reloj de 16Mhz y 2KB de memoria RAM.

Fuente de alimentación externa

La placa puede ser alimentada también mediante corriente continua suministrada por el conector jack de 3.5mm que podrá recibir entre 7 y 12V.

Pin de Reset

Podemos imitar el funcionamiento del botón reset suministrando un valor LOW(0V) para reiniciar el microcontrolador.

Pin de 3.3V

Desde aquí podremos suministrar 3.3V a los dispositivos que lo necesiten con una corriente máxima de 50mA. Es generada gracias al chip FTDI integrado en la placa.

Pin de 5V

Este pin saca una tensión de 5v del regulador de la placa. El regulador es necesario puesto que puede ser alimentada con distintos voltajes.

Pin de Vin

Es el voltaje de entrada cuando se usa una fuente de alimentación externa (no tiene en cuenta la conexión USB). Se puede proporcionar voltaje a la placa a través de este pin, o en caso de que se esté utilizando una fuente de alimentación externa tomar el valor que está siendo suministrado.

Pines analógicos

Esta placa contiene 6 pines de entrada analógicos. Los elementos que se conecten aquí suelen tener mayor precisión que los digitales pero su uso requiere de una lógica levemente mayor. Más adelante se comentará el uso de un termistor analógico.

2.1.2 Seeeduino

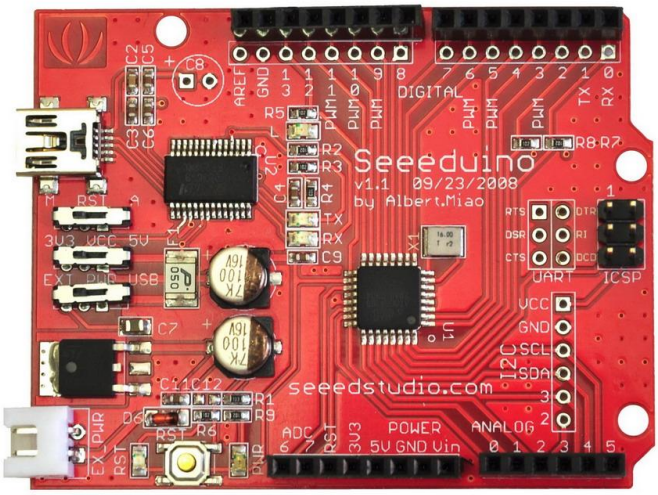

Ilustración 3: Frontal de la placa Seeeduino

La placa Seeeduino ha sido diseñada por terceros, en concreto Seeed. Se basa en el esquema de una Arduino Diecimilia donde los diseñadores han cambiado/añadido ciertos elementos para mejorarla. El fabricante afirma que todos los elementos que sean compatibles en una, lo serán en la otra (a nivel de hardware/software).
Tiene un tamaño de 71x53mm. Su unidad de procesamiento consiste en un microcontrolador ATmega168. Puede ser alimentada mediante USB o alimentación externa y contiene pines tanto analógicos como digitales. De esta placa se van a comentar solo las características que la diferencian de Arduino Uno.

La tabla siguiente resume sus componentes:

Microcontrolador	ATmega168
Voltaje operativo	5V/3.3V
Voltaje de entrada(recomendado)	7-12V
Voltaje de entrada (limites)	6-20V
Pines digitales E/S	14 (de los cuales 6 proporcionan salida PWM)
Pines de entrada analógica	8
Corriente continua para pines E/S	40 mA
Corriente continua para pines de 3.3V	50 mA
Memoria Flash	16 KB (ATmega168) de los cuales 2 KB son para el bootloader
SRAM	1 KB (ATmega168)
EEPROM	512 bytes (ATmega168)
Velocidad del reloj	16 MHz

En la tabla anterior se puede apreciar que el microcontrolador utilizado es más simple que el ATmega328 que utiliza Arduino Uno.
Es posible hacer uso de dos pines analógicos más.
Para alimentarse se puede conectar un USB mini o una fuente de alimentación externa mediante 2 pines Jack JST.
Además añade 3 selectores para cambiar el reset manual/automático, elegir la tensión de funcionamiento de la placa entre 3.3V/5V y seleccionar el modo de alimentación ya sea por USB o externa.

2.2 Software

La plataforma Arduino tiene un lenguaje propio que está basado en C/C++ y por ello soporta las funciones del estándar C y algunas de C++. Sin embargo, es posible utilizar otros lenguajes de programación y aplicaciones populares en Arduino como Java, Processing, Python, Mathematica, Matlab, Perl, Visual Basic, etc. Esto es posible debido a que Arduino se comunica mediante la transmisión de datos en formato serie que es algo que la mayoría de los lenguajes anteriormente citados soportan. Para los que no soportan el formato serie de forma nativa, es posible utilizar software intermediario que traduzca los mensajes enviados por ambas partes para permitir una comunicación fluida. Es bastante interesante tener la posibilidad de interactuar con Arduino

mediante esta gran variedad de sistemas y lenguajes puesto que dependiendo de cuales sean las necesidades del problema que vamos a resolver podremos aprovecharnos de la gran compatibilidad de comunicación que ofrece.

El entorno de desarrollo de Arduino es sencillo e intuitivo además puede descargarse gratuitamente desde su página oficial para distintos sistemas operativos. Ha sido implementado con Processing, un lenguaje similar a Java. Su última versión es la 1.0.2 aunque en el proyecto se ha utilizado la 1.0.1. Es importante remarcar que la placa Arduino Uno solo la podremos utilizar a partir de la versión beta 0021.
Está formado por una serie de menús, una barra de herramientas con botones para las funciones comunes, un editor de texto donde escribiremos el código, un área de mensajes y una consola de texto. En la ilustración 4 se puede apreciar la composición del software de Arduino.

Ilustración 4: Interfaz del software de Arduino

A continuación comentaremos la utilidad de cada área del programa centrándonos solo en lo importante.

Menú

La parte más importante se encuentra en Herramientas. Desde aquí podremos configurar el programa para que pueda comunicarse con la placa Arduino.
Pasando el ratón por Tarjeta aparecerá una lista con los tipos de placa Arduino que el programa comprende. Aquí seleccionaremos Arduino Uno o Diecimilia (para la Seeeduino) dependiendo de con cual estemos trabajando.
En el campo Puerto Serial seleccionaremos el que corresponda a nuestra placa que conectaremos mediante USB. Si utilizamos Windows el puerto tendrá un nombre del estilo COMx pero en Linux será /dev/ttyUSBx donde x es un número. En caso de que aparezcan varios puertos serie y no sepamos cual es el de nuestra placa procederemos a desconectarla, anotamos los puertos que aparecen, reconectamos la placa y volvemos a mirar la lista de puertos. El nuevo puerto que haya aparecido será el de nuestra placa.

Botones comunes

Estos botones son accesos rápidos a ciertas acciones que también están disponibles mediante el menú. Los botones son los siguientes:

- Verificar: comprueba y compila el código.
- Cargar: además de compilar el código lo inyecta en la placa.
- Nuevo: crea un nuevo sketch.
- Abrir: abre un sketch previamente guardado.
- Guardar: almacena en disco los cambios realizados en el sketch.
- Monitor Serial: abre una nueva ventana desde la que podremos comunicarnos bidireccionalmente vía serie con la placa, es decir, podremos leer la información que nos envía o proporcionarla nosotros. La ilustración 5 muestra esta ventana.

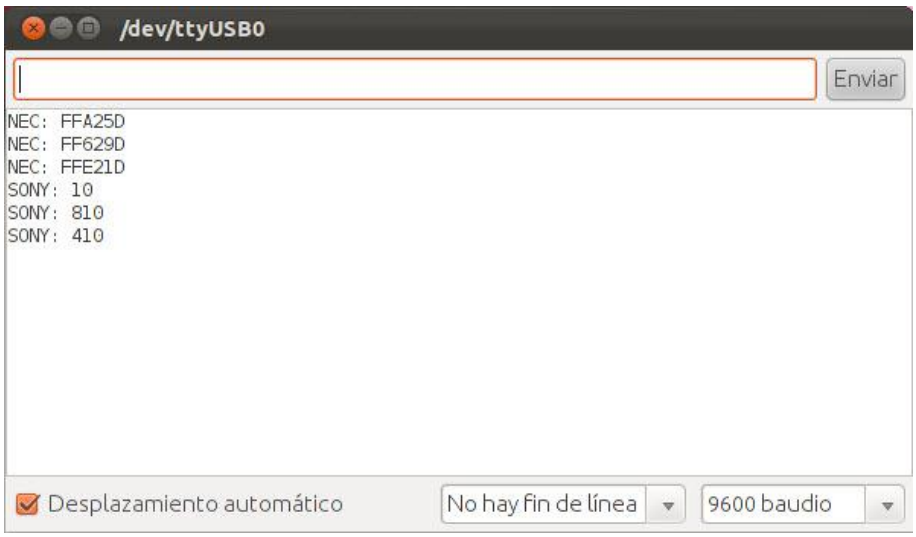

Ilustración 5: Ventana del monitor serie

Editor de texto

En este área escribiremos la implementación (denominada por el programa sketch) para poder cargarla en la placa Arduino. El programa tiene 3 partes. La primera consiste en la inclusión de librerías y la declaración de constantes o variables globales que se podrán utilizar en cualquier función del programa. La segunda es el método *setup()* que es el encargado de inicializar los dispositivos conectados a la placa y será ejecutado solo al iniciar el sistema. La tercera parte consiste en el método *loop()* que ejecutará su código continuamente, es decir, en modo bucle. Aquí es donde se escribirá la lógica de la placa Arduino. Como el lenguaje es muy similar a C es posible crear otros métodos para separar bloques funcionales y dejar ordenado el programa.

Área de mensajes

Muestra la situación del programa al haber utilizado uno de los botones comunes.

Consola de texto

Aquí aparecerán con mayor detalle los eventos del área de mensajes.

3. Dispositivos acoplables a Arduino

Para conseguir las características de un sistema domótico es necesario que además del órgano central que controle el sistema tengamos a disposición sensores que puedan recoger datos sobre la situación de cada habitación de la vivienda. Dependiendo de estos datos el sistema domótico debe ser capaz de comunicarse con los actuadores para mejorar la situación de la vivienda. También deben existir elementos con los que el usuario pueda comunicarse con el sistema y pueda hacer los cambios oportunos manualmente.

Los dispositivos estarán conectados mediante cables o directamente acoplados a la placa Arduino. Algunos de ellos disponen de librerías que deberemos adjuntar al programa para poder usar las utilidades que contengan. Para ello añadiremos la carpeta de la librería en la carpeta *libraries* del entorno de desarrollo de Arduino.
Al principio del código del sketch incluiremos la librería con la línea:
#include <nombreLibreria.h>

Para utilizar los métodos de sensores y actuadores digitales debemos tener en cuenta que solo tenemos dos posibles valores, HIGH representa el nivel alto y LOW el nivel bajo.

En el caso de los analógicos su uso es levemente más complejo pero también más configurable ya que tiene que leerse/escribir un voltaje de 0 a 5 voltios que se representa en 10 bits (lectura) o en 8 bits (escritura), es decir la tensión puede tener 1024 (lectura) o 256 (escritura) valores distintos. Más adelante se hará un ejemplo con un termistor.

3.1 Sensores

Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, movimiento, pH, etc. Una magnitud eléctrica puede ser una resistencia eléctrica (como en un detector de temperatura resistivo), una capacidad eléctrica (como en un sensor de humedad), una tensión eléctrica (como en un termopar), una corriente eléctrica (como en un fototransistor), etc.

Los sensores siempre que estén activados estarán tomando continuamente la situación actual de una habitación y es el servidor o la placa Arduino quien leerá esta información y decidirá cómo actuar. Pueden ser digitales o analógicos.

Los digitales tienen que ser inicializados como pin de salida con el método *pinMode(numeroDePin, OUTPUT)*. Para poder obtener una lectura de los datos

usaremos el método *digitalRead(numeroDePin)*. Los analógicos no requieren esta fase de inicio y para leer lo haremos con *analogRead(numeroDePin)*. Es recomendable asignar a una variable la lectura recibida por los métodos para evitar tener que llamar a la misma función en caso de necesitarse de nuevo. Como los sensores analógicos son algo más complejos que los digitales se tratarán con un ejemplo.

Los sensores que sean responsables de la seguridad de la vivienda deberían avisar del evento mediante un actuador (por ejemplo un timbre o LED) o algún elemento de comunicación (como un correo electrónico o un mensaje de texto al móvil). También podría almacenarse el suceso en un fichero del servidor.
A continuación se describirán algunos sensores a tener en cuenta en un sistema domótico.

Módulo de gas

El detector de gas hace que la vivienda gane en seguridad si cuando detecta un nivel alto de gas (lectura HIGH) el sistema avisa a la persona. Sería importante que el sistema pudiera desconectar la mayor parte de red eléctrica posible de la vivienda.

Ilustración 6: Sensor de gas

Módulo PIR

Otro elemento que interviene en la seguridad cuando no hay nadie en casa es un detector de movimiento. En caso de detectar suficiente movimiento se leerá un nivel alto.

También se puede utilizar para el confort del ser humano. En caso de detectar movimiento en la habitación encender por ejemplo las luces o la calefacción, dependiendo también de la lectura responsable de los dos casos.

Ilustración 7: Sensor de movimiento

Módulo de luz

Este dispositivo es capaz de detectar el nivel de intensidad de luz que hay en la habitación de forma analógica. El sistema leerá el voltaje y en caso de detectar un nivel bajo de luz podría encender las luces de la habitación siempre y cuando se detecte movimiento.

Ilustración 8: Sensor de luz

Módulo de humedad (y temperatura)

Algunos dispositivos son capaces de obtener varias mediciones en el mismo módulo. El módulo de la ilustración corresponde a un DHT11 capaz de representar digitalmente la humedad ambiental medida en % además de la temperatura en Cº. Tiene una precisión decimal y dispone de su propia librería que contiene los métodos para recoger sus mediciones. Este módulo es interesante colocarlo en la zona externa de la casa, como el balcón, la galería o el jardín.

Ilustración 8: Sensor de humedad y temperatura (DHT11)

Módulo de temperatura

En caso de utilizar calefacción o aire acondicionado el sensor de temperatura puede ayudar a reducir el coste de la factura de la luz y acomodar un poco más al ser humano. Cuando se detecte cierto umbral de temperatura podría apagarse/encender o modificar la potencia de la calefacción.

Ilustración 9: Sensor de temperatura

La ilustración 9 muestra un termistor analógico, es decir, un sensor resistivo de temperatura. Su funcionamiento se basa en la variación de la resistividad que presenta un semiconductor con la temperatura.

3.1.1 Programación de sensores analógicos. El ejemplo del termistor

A continuación vamos a mostrar cómo se debe programar el sensor analógico de la Ilustración 9 ya que es ligeramente más complejo que el digital.

En primer lugar debemos obtener la curva característica de la resistencia del sensor, que variará dependiendo de la temperatura. Con ella obtendremos una tabla con el valor de la resistencia y la temperatura que fuerza ese valor, por ejemplo en pasos de 10ºC.

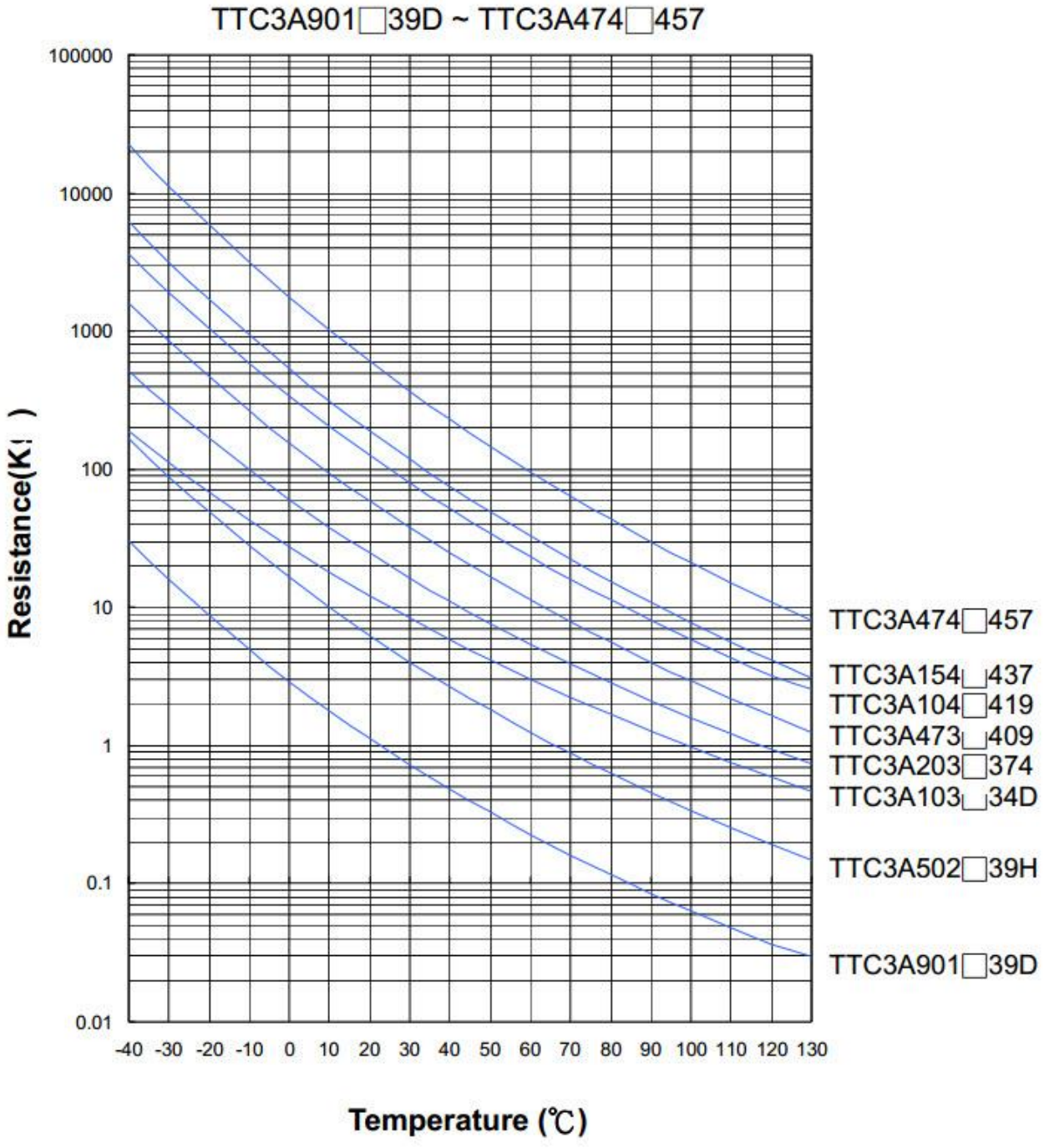

Ilustración 10: Curva característica de la resistencia del termistor analógico

Una vez tenemos la tabla en un programa como Microsoft Excel (Ilustración 11) procederemos a crear un gráfico de dispersión seleccionando los datos de resistencia y temperatura oportunos. Seleccionamos por ejemplo el rango de -20 a 80 ºC y aparecerán los puntos en la gráfica. Ahora hacemos click sobre cualquier punto, botón derecho y Agregar línea de tendencia. Podremos seleccionar el tipo de regresión que deseemos. En nuestro caso será la Polinómica y la pondremos de orden 5 para conseguir una línea de tendencia que se acerque bastante a los puntos (aunque no es exacta). Además seleccionaremos la opción Presentar ecuación en el gráfico. Con todo esto tendremos como resultado el gráfico de la Ilustración 12 cuya finalidad es conseguir la fórmula para poder conocer la temperatura en función del valor de la resistencia en el termistor.

	A	B	C
1	resis	temp	
2	165	-40	
3	90	-30	
4	14	-20	
5	11,8	-10	
6	10,7	0	
7	10	10	
8	6	20	
9	4	30	
10	2,7	40	
11	1,85	50	
12	1,3	60	
13	0,9	70	
14	0,6	80	
15	0,45	90	
16	0,35	100	
17	0,26	110	
18	0,19	120	
19	0,16	130	
20			

Ilustración 11: Tabla con el valor de la resistencia y la correspondiente temperatura del termistor TTC3A502-39H

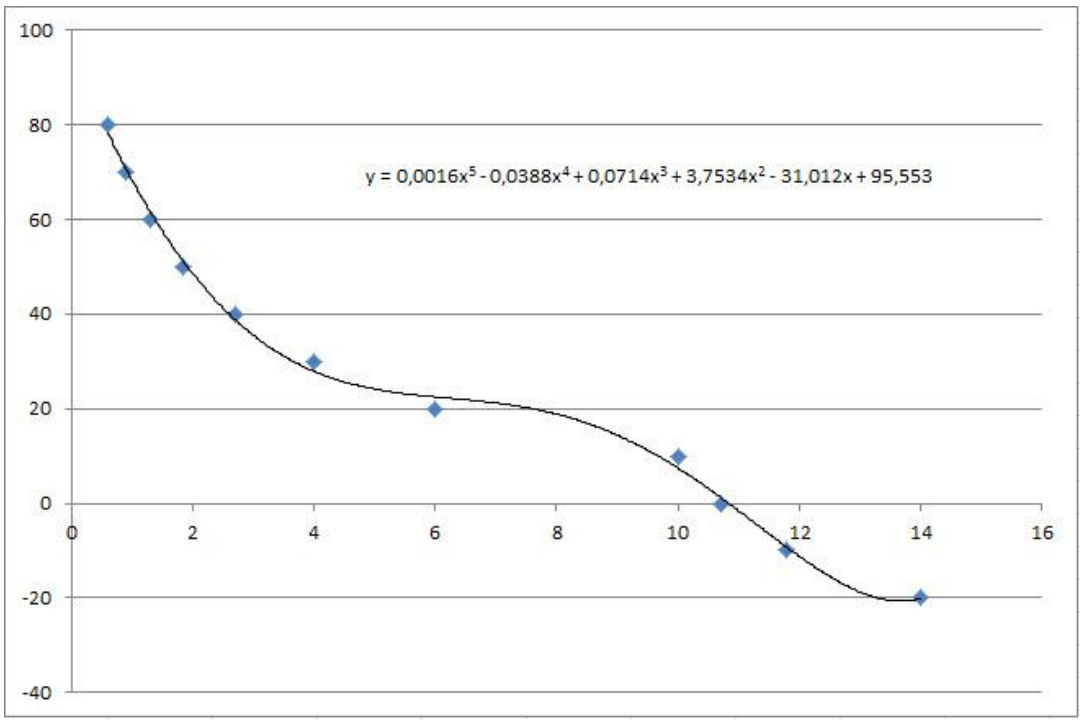

Ilustración 12: Gráfico de dispersión con línea de tendencia y ecuación de grado 5. Eje x -> resistencia(KOhmios) Eje y -> ºC

Acto seguido necesitaremos conocer la intensidad que hace funcionar al termistor y permanecerá prácticamente constante para cualquier temperatura. Para conocer este dato mediante Arduino deberemos recordar la ley de Ohm V = I*R.

Como se ha dicho anteriormente necesitamos hacer una lectura analógica que nos devolverá un valor entero comprendido entre 0 y 1023. Haciendo las pruebas en una habitación a 30ºC (obtenido desde un termómetro de mercurio) la lectura *analogRead(númeroPinAnalógico)* nos devolvía 213. Este valor lo deberemos multiplicar por 5 y el resultado dividirlo entre 1023 para obtener el valor representado en voltios que es 1.05083089 V. Para conocer el voltaje de caída debemos restar el voltaje que suministra la placa (5V) al valor leído desde el sensor con lo que nos queda 3.94916911 V.

Ahora podremos obtener la intensidad mediante la ley de Ohm dividiendo el voltaje de caída entre el valor de la resistencia a 30ºC.
3.94916911 / (4*1000) = 0.000493646138 A
Este valor será prácticamente constante con lo que ya tendremos lo necesario para programar el método del termistor.

En el método leeremos el valor analógico del sensor (valorTermistor), lo pasaremos a voltios (voltaje) y calcularemos el voltaje de caída.
valorTermistor = *analogRead(númeroPinAnalógico)*;
voltaje = valorTermistor * 5.0;
voltaje /= 1023.0;
voltaje = 5.0 - voltaje;

Después obtendremos el valor de la resistencia aplicando la ley de Ohm (dividimos entre mil por ser KOhmios).
resistencia = (voltaje / 0.000493646138) / 1000;

Solo falta conocer la temperatura la cual conseguiremos aplicando la ecuación de grado 5 que obtuvimos anteriormente.
temperatura = 0.0016*pow(resistencia, 5) - 0.0388*pow(resistencia, 4) + 0.0714*pow(resistencia, 3) + 3.7534*pow(resistencia,2) – 31.012*resistencia + 95.553;

3.2 Actuadores

Un actuador es un dispositivo capaz de transformar energía (en nuestro caso eléctrica) en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado. Su función en un sistema domótico va a ser la de cambiar la situación de la vivienda tras un evento ocasionado al hacer por ejemplo una lectura de un sensor que debe ser tratada.

Módulo Relé

Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes. Este tipo de módulos permite activar actuadores como por ejemplo el de una persiana, la puerta del garaje o el de una bombilla.

Ilustración 13: Relé

3.3 Interfaces

Su principal objetivo es ofrecer comunicación entre el sistema y el ser humano. Consisten en elementos visuales/auditivos que avisan de eventos o táctiles para poder causarlos.

Módulo LED

La función de este dispositivo es avisar mediante un diodo emisor de luz la ocurrencia de un evento que puede requerir su atención. Pueden utilizarse de forma digital (encendido/apagado) o de forma analógica si se quiere variar la intensidad de la luz.

Ilustración 14: LED

Módulo timbre

Este elemento es capaz de producir sonidos. Módulo interesante para avisar al ser humano sobre un problema grave en la vivienda dado que la sensibilidad auditiva es mayor que la visual.

Ilustración 15: Timbre

Módulo pulsable

Para poder comunicarnos con el sistema y crear eventos podemos utilizar teclados o botones. Por ejemplo la función de un botón presionado al salir de casa podría ser apagar luces, calefacción y activar el sistema de seguridad.

Ilustración 16: Teclado y Botón

Módulo visualizador

Con él podremos conocer la situación de la vivienda y de la ocurrencia de eventos que nos envíe el sistema en forma de texto sobre una pantalla.

Ilustración 17: Pantalla LCD

3.4 Comunicadores

Este apartado abarca el conjunto de elementos que permiten la comunicación entre distintas placas Arduino y el servidor o incluso con electrodomésticos del hogar. El medio por el que circula la información puede ser por aire (modulación de ondas electromagnéticas) o físico (por cable) teniendo sus ventajas e inconvenientes. Normalmente estos dispositivos tendrán a nuestra disposición librerías con funciones ya implementadas que nos facilitará su manejo.

Si el medio es el aire el sistema total va a ser más barato puesto que evitamos tener que cablear las habitaciones, además de esto conseguimos que sean más estéticas. En cambio las transmisiones son menos seguras y puede haber problemas por el ruido ocasionado de otros elementos que utilizan el aire como forma de comunicación. Además los obstáculos que haya entre emisor y receptor van a reducir la distancia de transmisión.

Por parte de los sistemas que utilizan cables para enviar datos debemos tener en cuenta su coste de instalación además de estudiar si nos gustaría estéticamente ese cableado en la habitación. Pero esto puede suplirse sabiendo que las transmisiones serán más robustas y seguras.

Módulo Ethernet

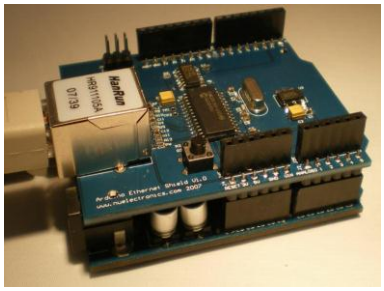

Ilustración 18: Ethernet Shield

Es una placa que se acopla encima de la Arduino y permite establecer conexiones a internet mediante el estándar Ethernet que utiliza el protocolo TCP/IP. Podemos conectarla a un router utilizando un cable RJ45 y le asignará una dirección IP. Con esta dirección podremos abrir conexiones entre el servidor y la placa o de placa a placa para enviar flujos de datos. Hay distintos chips y cada uno utiliza sus propias librerías. En nuestro caso hemos trabajado con el chip 28J60 que gasta las librerías *etherShield.h* y *ETHER_28J60.h*

Es importante tener en cuenta que en el código de la placa la configuración de la Ethernet Shield ha de ser correcta. Debemos asignarle la dirección IP que le proporcione el router en caso de que utilice DHCP. También hay que poner una dirección MAC única para que el router conozca los distintos dispositivos conectados. Además podremos abrir un puerto mediante el que escuchará peticiones. Esta configuración se ha de realizar en la función *setup(),* es decir en la fase de inicio de la placa.

La función principal de la Ethernet Shield va a ser leer peticiones, en nuestro caso HTTP (puerto 80). Las peticiones en HTTP tienen el siguiente formato: GET /ruta_del_objeto HTTP/1.1 Para tomar la petición nos basaremos en la función *serviceRequest()* que nos devolverá una cadena con toda la parte del protocolo ya tratada, es decir, obtendremos sólo la ruta del objeto.

Módulo Wi-Fi

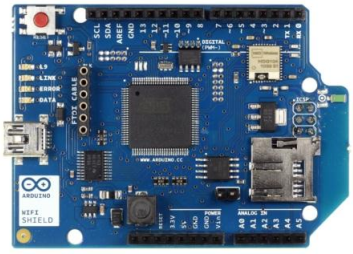

Ilustración 19: WiFi Shield

Si deseamos utilizar el protocolo TCP/IP pero queremos evitar tener que cablear la habitación podemos utilizar este módulo también acoplable a la Arduino. Aunque utiliza otra librería los métodos son equivalentes al del módulo Ethernet. La frecuencia de la señal ronda los 2.4GHz.

Módulo XBee

Este elemento se comunica de forma inalámbrica utilizando ZigBee que es un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radiodifusión digital de bajo consumo. Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y que necesiten un bajo consume. Utiliza unas frecuencias comprendidas entre 865MHz y 2.4GHz

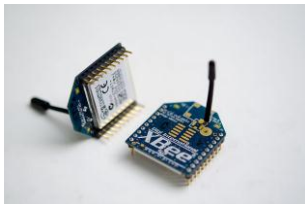

Ilustración 20: Módulo XBee

Módulo Bluetooth

Se denomina Bluetooth al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, que requieren corto alcance de emisión y basados en transceptores de bajo costo. Opera mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz. Su uso es adecuado cuando puede haber dos o más dispositivos en un área reducida sin grandes necesidades de ancho de banda.

Ilustración 21: Módulo Bluetooth

Bluetooth tiene la ventaja de simplificar el descubrimiento y configuración de los dispositivos, ya que éstos pueden indicar a otros los servicios que ofrecen, lo que redunda en la accesibilidad de los mismos sin un control explícito de direcciones de red, permisos y otros aspectos típicos de redes tradicionales. El WiFi utiliza el mismo espectro de frecuencia que Bluetooth con una potencia de salida mayor que lleva a conexiones más sólidas.

Módulo infrarrojos

Otra forma de comunicación sin medio físico es la utilización de rayos infrarrojos. Este tipo de comunicación consigue tener menos interferencias debido a la mayor frecuencia del espectro electromagnético en la que trabaja. Normalmente no es un método de comunicación entre placas Arduino sino que es utilizado para contactar con los electrodomésticos del hogar.

Ilustración 22: Emisor y Receptor Infrarrojos

Durante el proyecto se ha trabajado con los dispositivos de la ilustración 22 a la vez de con un mando que envía códigos al pulsar un botón y mediante el cual podemos crear

eventos. Han sido utilizadas dos librerías, la *NECIRrcv* que consigue descodificar la secuencia que envía el módulo emisor utilizando el protocolo NEC y la *IRremote* que consigue descodificar hasta cuatro tipos de protocolos distintos, NEC, Sony, RC5 y RC6 (estos dos últimos de la marca Philips). Esto se consigue porque cada protocolo utiliza una composición del mensaje distinta. Por ejemplo el protocolo NEC utiliza 24 bits por mensaje (o 6 números en hexadecimal) mientras que el de Sony solo necesita 12bits (o 3 números en hexadecimal) pero requiere una ráfaga de 3 mensajes seguidos en un corto espacio temporal para que el receptor acepte dicho mensaje (aunque luego esto a veces no se cumpla).

Puerto Serie

Un puerto serie o puerto serial es una interfaz de comunicaciones de datos digitales, frecuentemente utilizado por computadores y periféricos, donde la información es transmitida bit a bit enviando un solo bit a la vez, en contraste con el puerto paralelo que envía varios bits simultáneamente.

Ilustración 23: Cables para conectar un puerto serie

Es otro método de comunicación cableada que podemos usar para la comunicación entre un PC y una placa, o entre placas. El puerto serie consiste en un canal de recepción y otro de transmisión haciendo que el punto de recepción de un dispositivo esté conectado con el de transmisión del otro dispositivo. Todas las placas Arduino disponen de al menos un puerto serie compuesto por los pines digitales 0(rx) y 1(tx). Al conectar un cable de USB de la placa al ordenador estaremos aprovechando este puerto serie en el que debemos configurar la velocidad de datos en bits por segundo (baudios) para la transmisión de datos.

Para iniciar el puerto serie y establecer los baudios utilizaremos la función *begin(speed)* de la librería *Serial*. Se recomienda establecer el puerto serie a 9600 baudios. Esta fase se hace en la función de configuración de la placa *setup()*.

La placa Arduino tiene un buffer de recepción de datos de 128 bytes. Con la función *available()* podremos conocer cuántos caracteres (bytes) hay disponibles en el buffer. En caso de que haya datos en el buffer la función *read()* devolverá el primer carácter (byte) guardado en el puerto serie y lo retirará del buffer. En caso de que no haya datos devolverá un -1.

Es posible crear nuevos puertos serie con la librería *SoftwareSerial.h* en la que tendremos que especificar los pines de recepción y transmisión de datos.

3.5 Servidor

Es el órgano principal del sistema y va a ser el encargado de que el usuario pueda controlar los distintos elementos del hogar resolviendo sus peticiones. Apoyándonos en el servidor y en otros dispositivos podríamos controlar el sistema incluso fuera de casa, por ejemplo desde el móvil utilizando una aplicación, enviando un mensaje de texto o simplemente desde un navegador de internet.

Ilustración 24: Placa Raspberry Pi

Como se ha comentado en la introducción existen varias alternativas para utilizar como servidor. Podemos utilizar un ordenador personal muy simple, pero esto va a encarecer el sistema además de desaprovechar su capacidad, además de que ocuparía más espacio que otras opciones.

Una alternativa consiste en la placa computadora de bajo coste Raspberry Pi desarrollada en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. El diseño incluye un System-on-a-chip Broadcom BCM2835, que contiene un procesador central (CPU) ARM1176JZF-S a 700 MHz (el firmware incluye unos modos "Turbo" para que el usuario pueda hacerle overclock de hasta 1 GHz sin perder la garantía), un procesador gráfico (GPU) VideoCore IV, y 512 MB de memoria RAM. El diseño no incluye un disco duro o una unidad de estado sólido, ya que usa una tarjeta SD para el almacenamiento permanente; tampoco incluye fuente de alimentación o carcasa. Soporta varios sistemas operativos Unix como Debian, Fedora o Arch Linux. Su precio ronda los 30€.

Otra opción consiste en utilizar una placa Arduino que actúe como servidor pero esto va a limitar mucho las opciones de configuración del sistema domótico.

4. Arquitecturas de comunicación

Una vez conocido como funciona Arduino y los distintos dispositivos que nos pueden ayudar a construir el sistema domótico debemos abordar la distribución de las placas y cómo va a ser estructurada su comunicación. Se van a ofrecer un total de tres arquitecturas: centralizada, distribuida y mixta.

A continuación vamos a describir la distribución de las placas que servirán para comentar los tres casos. Esta decisión puede variar dependiendo del diseño de las habitaciones de cada vivienda. En nuestro sistema vamos a instalar una placa Arduino por cada habitación de la casa y separadas en dos niveles. El primer nivel puede constituir el conjunto del recibidor, comedor, cocina y dormitorio. El segundo nivel puede estar formado por la terraza, galería y baño. En el primer caso las placas estarán conectadas directamente con el servidor mediante un router con Ethernet, WiFi. Serán las encargadas de pasar el mensaje al segundo nivel en caso de que vaya dirigido a una de sus placas por ejemplo utilizando Bluetooth, XBee o Puerto Serie. Cada placa del primer nivel se encargará de la comunicación con otra del segundo nivel, esto es, estarán enlazadas por ejemplo la placa de la cocina y la de la galería.

La capacidad de decisión que realiza el servidor puede verse alterada dependiendo de la arquitectura de comunicación escogida. El servidor en todos los casos va a ser el responsable de comunicar al resto de placas eventos globales como puede ser el apagado de todas las luces de la vivienda. Por otro lado, estarán los eventos normales que serán los introducidos por el usuario, o por decisión de la placa en caso de que los valores obtenidos de los sensores no sean óptimos.

En caso de que un dispositivo tenga que tratar con dos protocolos de comunicación distintos ha de ser capaz de hacer las traducciones oportunas. Esto en el servidor puede realizarse apoyándonos por ejemplo en el lenguaje XML.

Antes de pasar a explicar las tres arquitecturas hay que hacer un apunte sobre las imágenes que se van a utilizar. Cada habitación (o placa Arduino) va a tener instalados sensores y actuadores como se puede ver en la Ilustración 25. Las habitaciones que no tengan las franjas (Habitación 1) van a ser placas esclavas, es decir, su función va a ser la de acatar órdenes del servidor o de las placas maestras que son las que contienen las franjas (Habitación 2) y serán las responsables de comprobar la situación de la habitación y ordenar una acción en consecuencia.

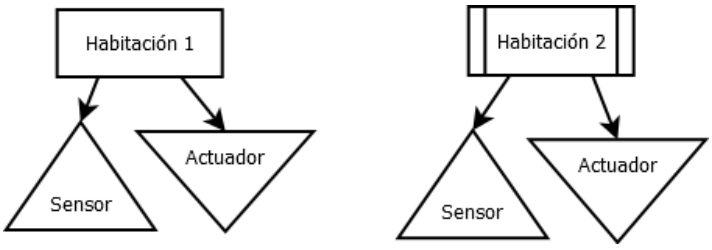

Ilustración 25: Sensores y Actuadores en la habitación

4.1 Centralizada

Este tipo de arquitectura va a tener instalada toda la lógica del sistema en el servidor que va a actuar como maestro. El resto de placas serán esclavas, es decir su programa solo va a conocer la ejecución de órdenes y la comunicación entre placas. La Ilustración 26 muestra esta situación.

Es por ello que todo el sistema fallará en caso de que caiga el servidor. Este efecto se puede paliar añadiendo un servidor de reserva que tendrá una copia de la funcionalidad del servidor principal y se encargará de preguntar cada cierto tiempo si está activo, en caso de no recibir respuesta reemplazaría al principal. Además esta solución aumentaría el coste del sistema.

Este sistema va a hacer que la mayor parte del flujo de información se sitúe en la parte superior del esquema pudiendo sobrecargarla reduciendo su eficacia.

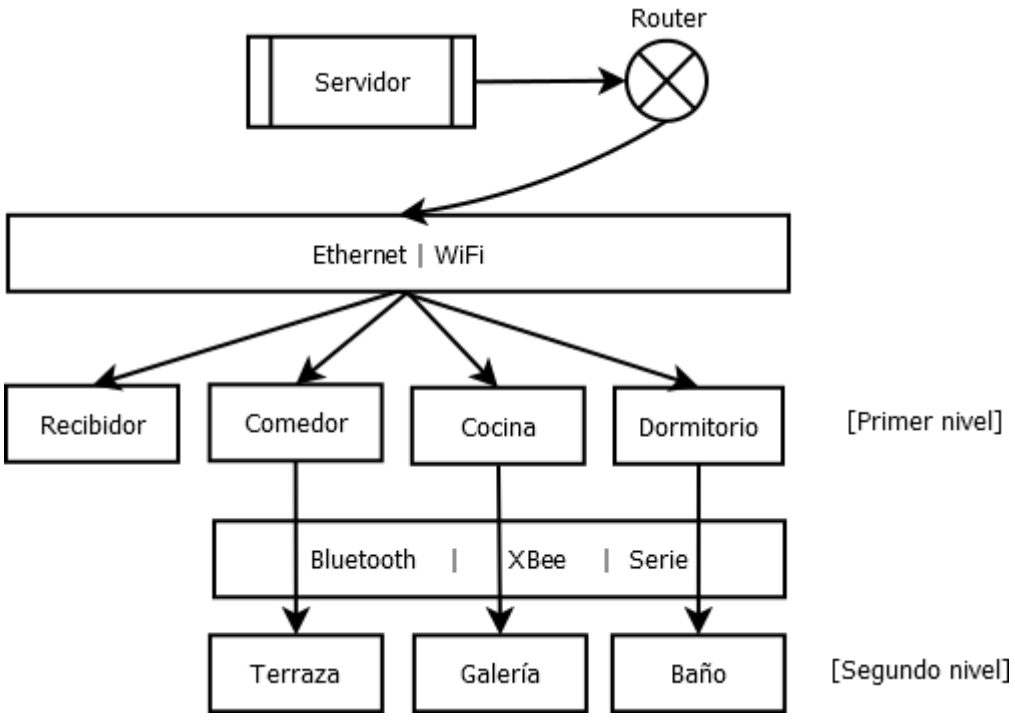

Ilustración 26: Esquema de la arquitectura centralizada

El servidor se encargará de enviar periódicamente peticiones a las habitaciones para comprobar su situación. El servidor estará a la escucha de mensajes de las habitaciones, si el usuario ha generado algún evento se enviará la orden a las placas correspondientes. En caso de que se el mensaje sea un valor de un sensor comprobará si está dentro de los límites correctos, si no es así generará una orden que enviará hacia la habitación.

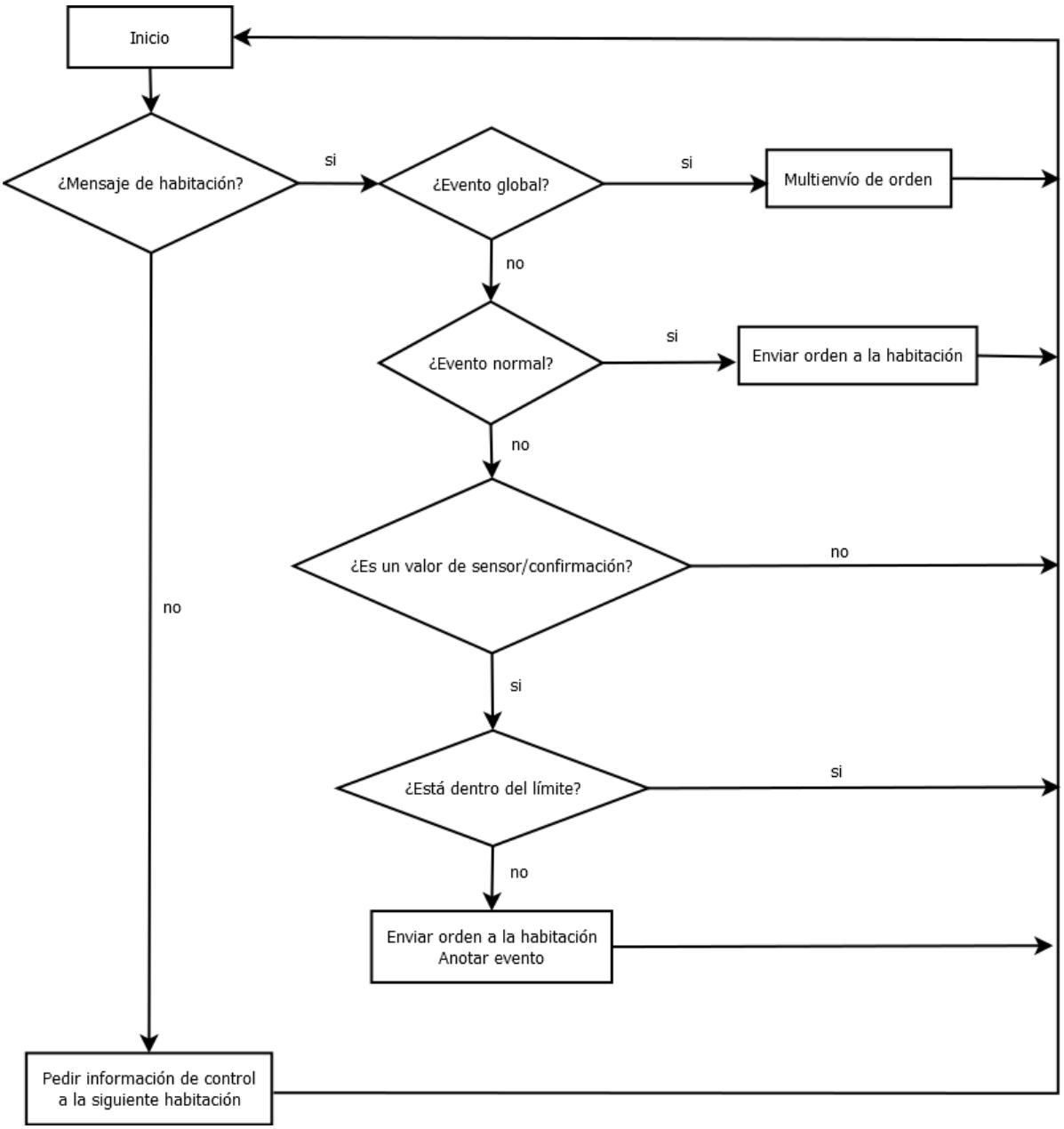

Ilustración 27: Función del servidor en la arquitectura centralizada

La función de la placa será muy simple, estará escuchando por si llega algún mensaje si es del servidor primero comprobará que va dirigido a ella, si no es así lo pasará al siguiente nivel. En caso de que sea para ella ejecutará sus órdenes o comprobará el estado de los sensores y le enviará sus valores.

Después comprobará si el usuario ha introducido algún evento manualmente, por ejemplo pulsando un botón.

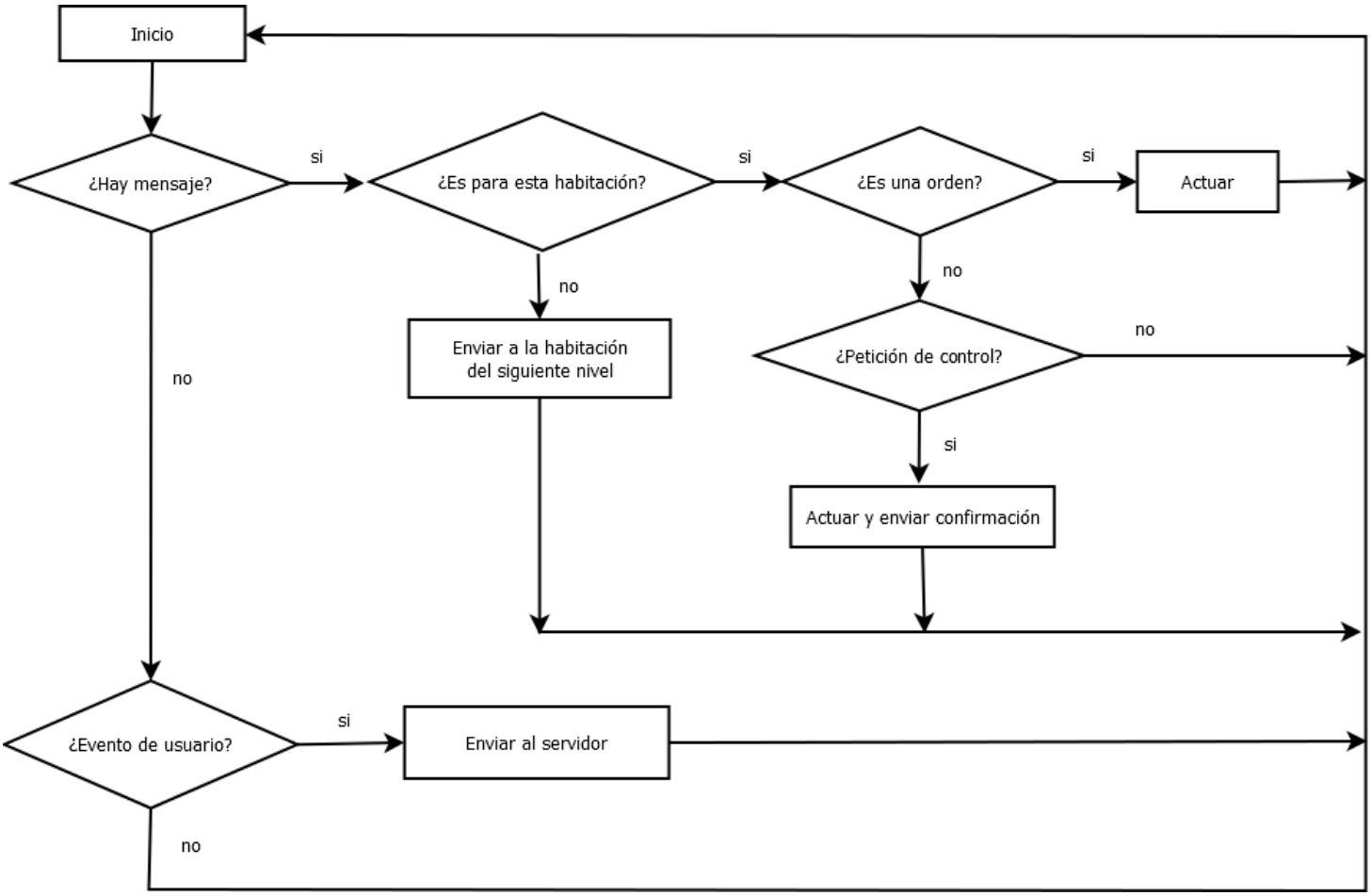

Ilustración 28: Función de la placa en la arquitectura centralizada

4.2 Distribuida

Esta forma de comunicación va a tener a todas las placas como maestras por ello el servidor va a tomar un papel mucho menos decisivo sobre el hogar. Cada habitación va a tomar sus propias decisiones sobre cómo mejorar su situación. En caso de que el usuario introduzca un evento global será el servidor el que difunda la orden. También anotará los eventos importantes como puede ser una anomalía en el nivel de gas, o una activación del PIR cuando no hay ningún familiar en casa.

Esta forma de comunicación será más segura en caso de que falle el servidor dado que cada habitación puede actuar por separado. Además repartirá el flujo de información por todo el sistema.

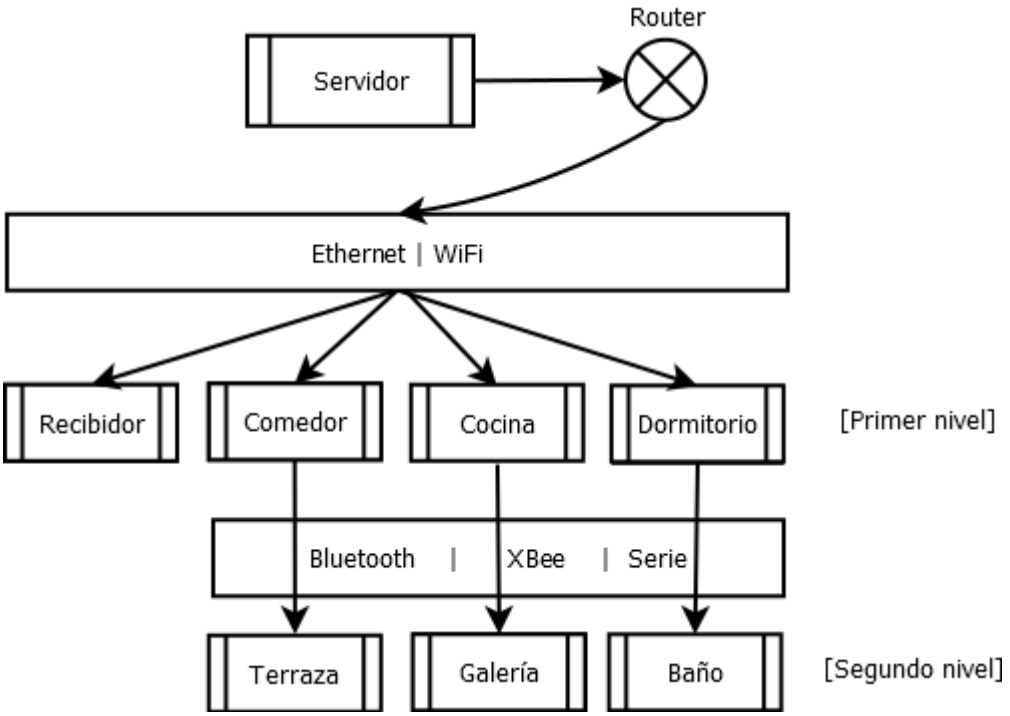

Ilustración 29: Esquema de la arquitectura distribuida

Como se ha comentado, en este caso el servidor pierde mucho poder con respecto a la arquitectura centralizada. Su función va a ser de “caja negra” para eventos problemáticos y como comunicador de órdenes globales.

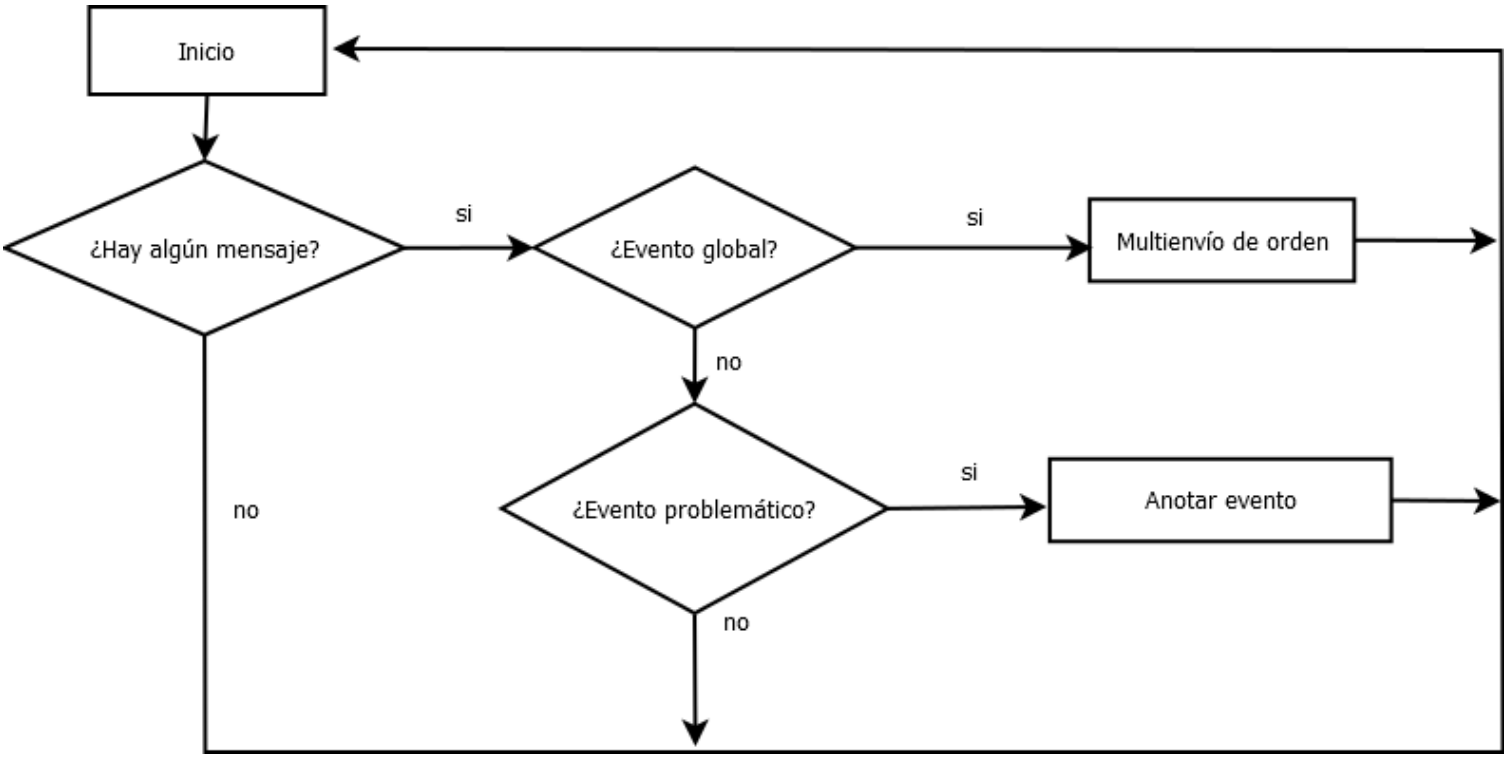

Ilustración 30: Función del servidor en la arquitectura distribuida

La función de la placa va a permitir que tenga el dominio sobre la habitación. Gran parte del control que poseía el servidor en la arquitectura centralizada pasa a dividirse por cada habitación de la vivienda. En caso de que haya algún evento importante (global o problemático) se avisará al servidor.

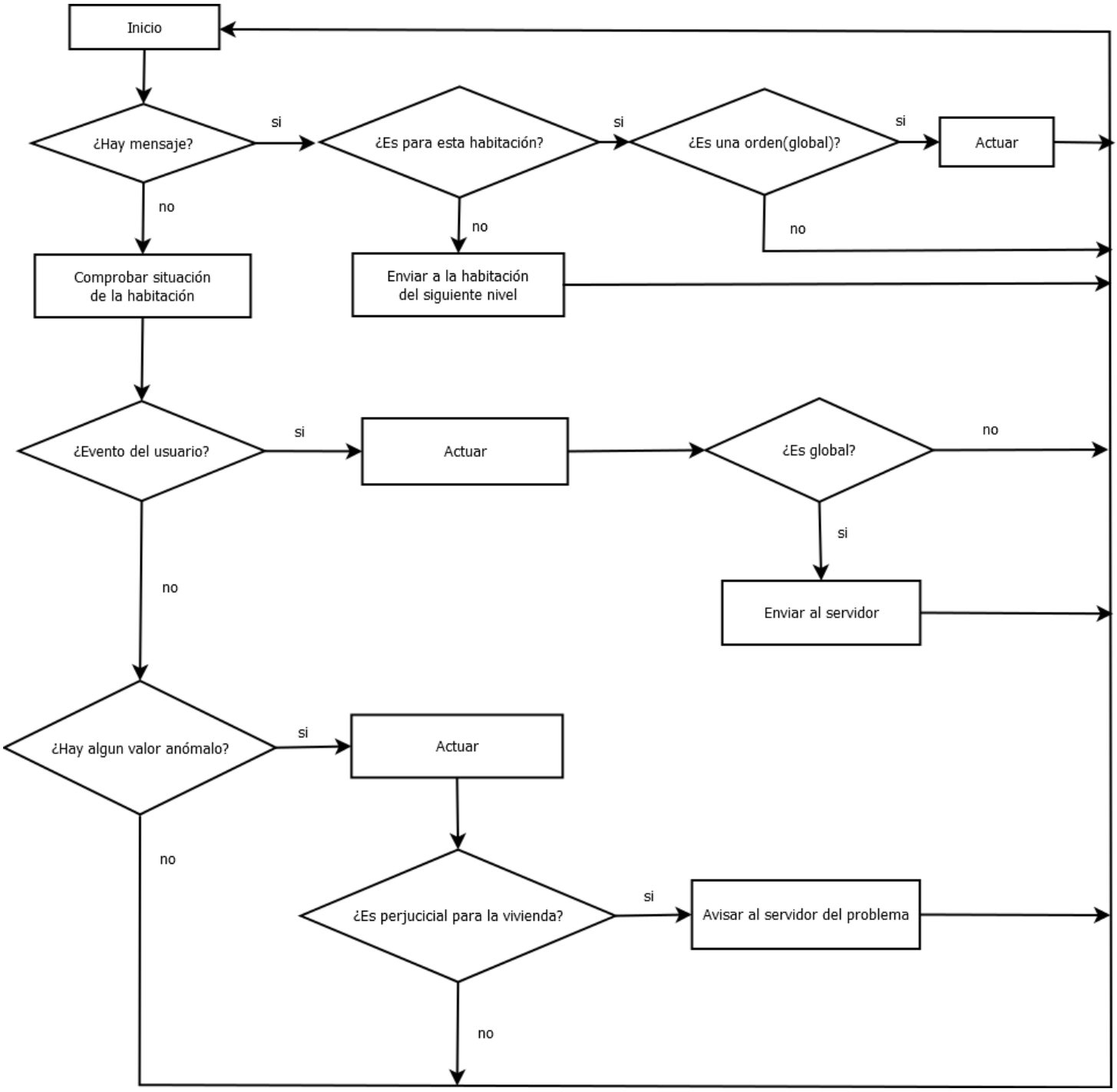

Ilustración 28: Función de la placa en la arquitectura distribuida

4.3 Mixta/Híbrida

Esta arquitectura va a ser una mezcla de las dos anteriores. Es decir tendremos ciertas placas que serán maestras que pueden ser las del primer nivel y actuarán como servidor para las esclavas que se situarán en el siguiente nivel.

Dependiendo del nivel en el que estemos adoptaremos las propiedades (ventajas y desventajas) de la arquitectura centralizada o distribuida. Es por ello que en este caso no se van a replicar las ilustraciones con la función del servidor y de la placa.

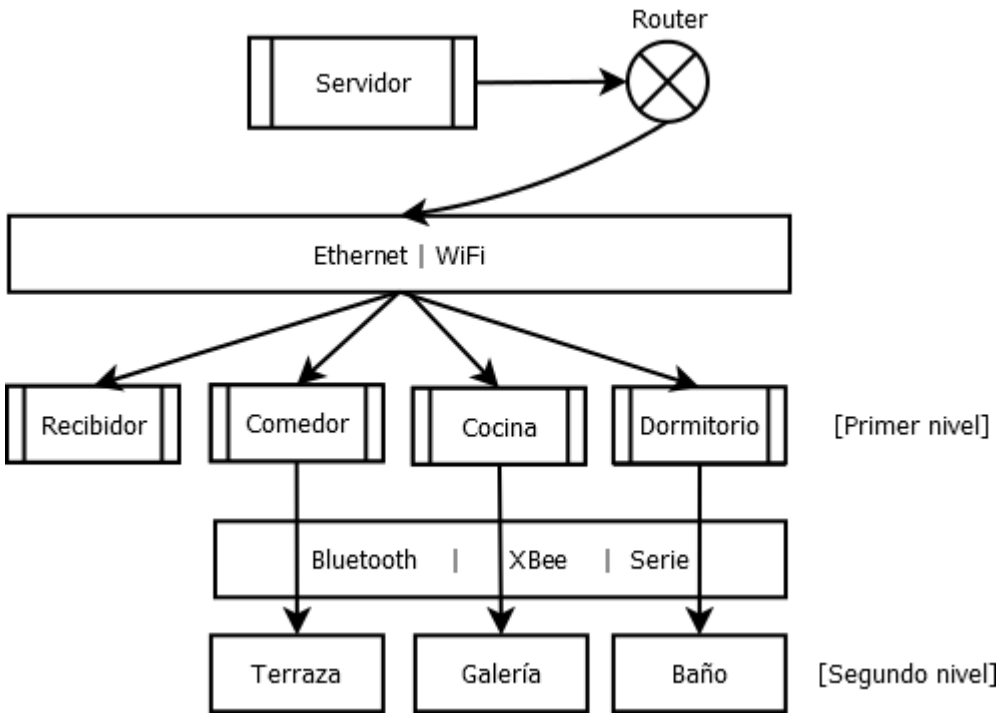

Ilustración 30: Esquema de la arquitectura mixta/híbrida

5. Prototipo de servidor domótico

En este último punto vamos a proponer un modelo sencillo de servidor domótico utilizando placas Arduino y otros dispositivos acoplables. Se comentarán el montaje del sistema y su funcionamiento intentando demostrar el porqué de cada elección.

5.1 Montaje

Se ha escogido una arquitectura de comunicación centralizada en el que el órgano central, es decir el servidor, es una placa Arduino en vez de un PC. El porqué de esta decisión es que pensamos que la placa tiene las características necesarias para que cumpla con garantía el funcionamiento del sistema. Disponemos de dos placas la Arduino Uno y la Seeeduino con lo cual resulta más fácil hacer la elección de cual actuará como servidor, esto es la placa Uno dado que tiene más capacidad de procesamiento. Simularemos que cada una de ellas está en una habitación distinta. El servidor en el recibidor dado que es la primera placa que vemos al entrar y la última al salir de la vivienda y por ello tendrá la capacidad de gestionar eventos a la otra habitación. La Seeeduino recibirá por ejemplo el nombre de comedor.

Como elementos acoplables hemos escogido un Ethernet Shield que instalaremos en la placa servidor y conectaremos al router con un cable RJ45. Esto nos permitirá comunicarnos con el sistema desde cualquier dispositivo conectado a la red local del router.

Otro método de comunicación utilizado es el puerto serie. Deberemos tener el puerto serie del recibidor conectado directamente con un PC (mediante USB) desde el cual podremos hablar con el sistema y a la vez alimentarlo. También hay un puerto serie dedicado para las comunicaciones entre el recibidor y el comedor. Para crear este puerto serie se han reservado los pines 2 y 3 del recibidor para recibir y transmitir respectivamente mediante los cables. Por el lado del comedor hemos utilizado los pines reservados por defecto, es decir, los pines 0 y 1. Es importante tener en cuenta que el pin de transmitir de una placa debe ir conectado al pin de recibir de la otra y viceversa.

La placa del comedor puede ser alimentada de distintas formas pero se ha escogido aprovechar la que genera el USB del recibidor. Debemos conectar el pin GND de ambas placas entre sí, además del pin de +5V. En caso de querer alimentar la placa del comedor mediante USB deberemos modificar el código del programa para que no utilice los pines 0 y 1.

Para simular el encendido y apagado de las luces de cada habitación se han utilizado leds. En caso de querer hacerlo más real puede sustituirse el led por un relé y una bombilla. Por parte del comedor aprovechamos el led integrado en la placa (pin 13). En cambio el led integrado del recibidor no lo podemos utilizar ya que lo hace el Ethernet Shield por ello conectaremos uno externo.

Además, en el comedor hemos añadido un módulo capaz de medir la temperatura y la humedad. Se trata del modelo Keyes DHT11 comentado en el punto 3.

La siguiente ilustración muestra el montaje final de nuestro sistema domótico simple:

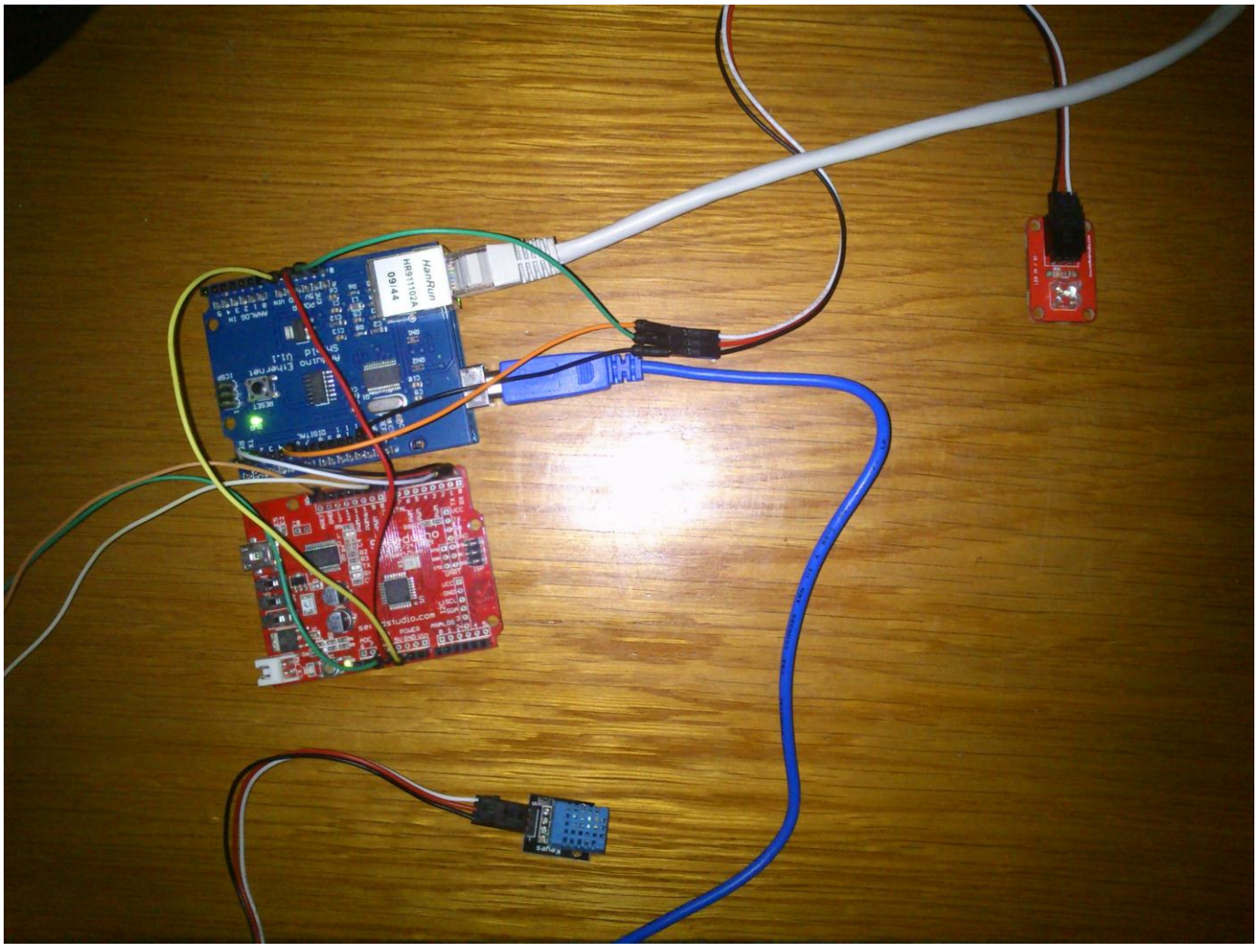

Ilustración 31: Montaje del prototipo de un sistema domótico utilizando Arduino

5.2 Funcionamiento

Parte del modo de trabajar del sistema está comentado en el código de las dos placas recibidor.ino y comedor.ino que se pueden encontrar en los archivos adjuntos del proyecto.

Como la comunicación en el sistema se basa en una arquitectura centralizada debemos recordar que las órdenes van a pasar siempre por el servidor aunque no vayan dirigidas específicamente a él.

El sistema es capaz de reconocer mensajes con el formato habitación/dispositivo/valor donde habitación representa al [recibidor, comedor, evento salirDeCasa] el dispositivo puede ser [led, temperatura, humedad] y el valor un numero entero que se pasará al dispositivo solo en caso de que sea de tipo escritura (*digitalWrite/analogWrite*), por ejemplo el led. Como el sensor DHT solo puede leer podremos omitir el campo valor en caso de querer obtener la temperatura o humedad.

El recibidor va a ser capaz de encender/apagar el led, ejecutar el evento salirDeCasa que apaga todos los leds de las habitaciones y además actuar de servidor. Esto último va a consistir en trasladar los mensajes destinados al comedor (el evento salirDeCasa también lo es) y presentar la información al usuario ya sea por el software de Arduino o por un navegador de internet. El bucle principal del recibidor va a buscar mensajes por el puerto serie del USB (*Serial*) que está conectado al PC, por el puerto serie al comedor para obtener sus respuestas (*puertoSerieAComedor*) y por Ethernet con el protocolo HTTP. Si existe algún mensaje actuará en consecuencia, esto es, si la orden es para el recibidor la ejecutará y si es para el comedor se la enviará por el puerto serie reservado a esta comunicación (*puertoSerieAComedor*).

Por la parte del comedor podremos encender/apagar el led, leer la temperatura, la humedad y hacer que se ejecute el evento salirDeCasa que apagará el led. El código del comedor va a consistir en leer el puerto serie, actuar y enviar respuesta de confirmación al recibidor.

Tenemos dos opciones para hablar con el sistema:

- La primera consiste en el envío de información a través del puerto serie del PC utilizando el software de Arduino. Si el mensaje va destinado al recibidor se mostrará separado por los tres campos habitación(o evento), dispositivo y valor. En caso de que vaya dirigido al comedor se imprimirá también la respuesta con un valor ok si ha sido capaz de efectuar la operación sobre el led o con el valor de la temperatura/humedad pedido.

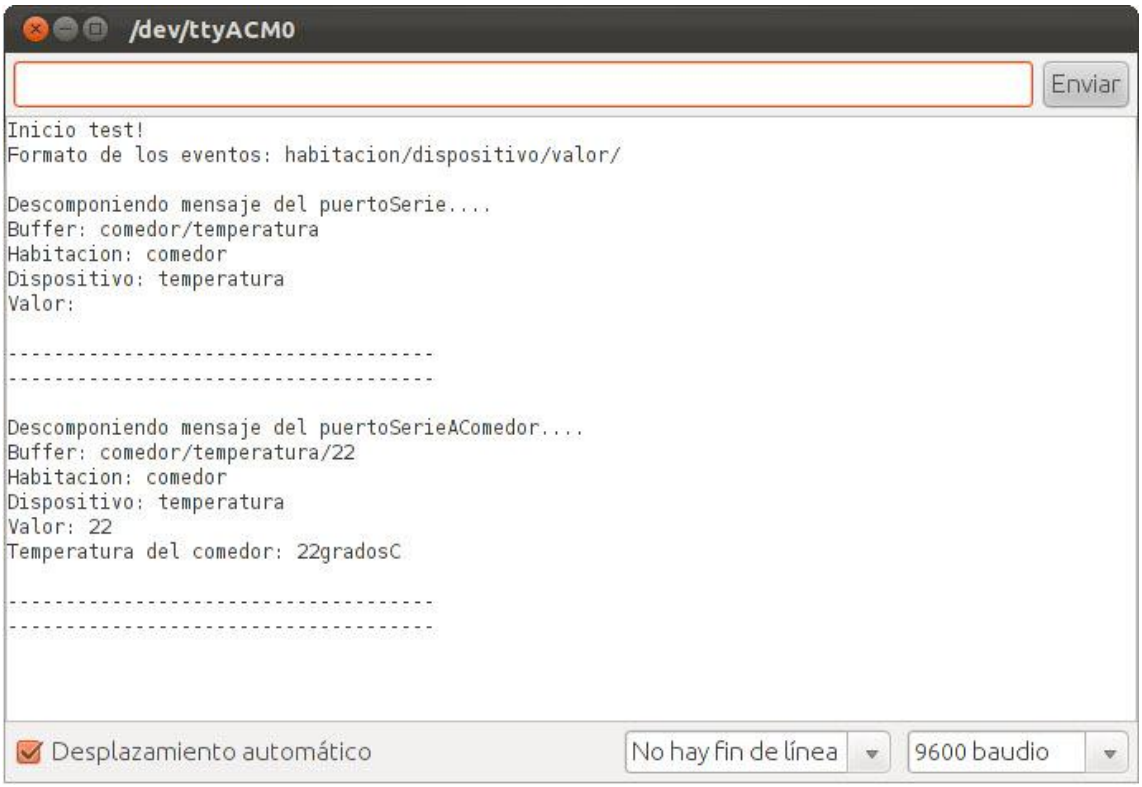

Ilustración 32: Comunicación con el sistema utilizando el software de Arduino

- La segunda trata del protocolo HTTP que utilizaremos desde un navegador de internet. Es importante tener en cuenta que en el código del recibidor la configuración de la Ethernet Shield ha de ser correcta. Debemos asignarle la dirección IP que le proporcione el router en caso de que utilice DHCP. También hay que poner una dirección MAC única para que el router conozca los distintos dispositivos conectados. Es posible configurar el puerto aunque se ha dejado por defecto el de HTTP, es decir el 80.
  Para poder tratar con el sistema debemos indicar en la dirección web la IP del servidor, el puerto (si es el 80 no es necesario incluirlo) y el mensaje con un formato correcto.
  En este caso se imprimirá el mensaje y su respuesta en HTML y podremos visualizarlo en el navegador. También lo podremos ver por el puerto serie que lee el software de Arduino.
  Como puede observarse la respuesta que aparece en el navegador proporciona la información justa sin presentar una interfaz elaborada. Esto es debido a que el envío de la página HTML producida por la placa Arduino tiene una determinada capacidad y en caso de superarla no llegaría a enviarse al navegador.

Ilustración 33: Comunicación con el sistema utilizando un navegador de internet

6. Conclusión

Este proyecto me ha dado la oportunidad de aprender mucho sobre el mundo de la domótica y un poco de electrónica general. Aunque originalmente el proyecto se basaba en construir un sistema domótico con una placa Arduino y un servidor que trabajase sobre Linux, he decidido desviarme un poco y he cambiado el servidor por otra placa Arduino. Con esto he conseguido ver que aunque se puede hacer, reduce las posibilidades de ampliación del sistema.

En todo caso, se ha podido demostrar que es posible instalar un sistema domótico apoyándonos en la plataforma Arduino, con un coste muy inferior al que se utiliza en las viviendas de lujo, a cambio de dedicarle un poco de tiempo.

Según avanzaba en el desarrollo del proyecto se me hacía más necesario probar los distintos dispositivos que podía instalar. Es importante tener las herramientas adecuadas para trabajar dado que en algún momento del proyecto me ha fallado algún dispositivo y tras comprobarlo a nivel físico con un tester eléctrico se detectaron los fallos, aunque fue remplazado y se pudo continuar sin problemas.

A la hora de programar los distintos elementos ayuda mucho realizar de antemano un esquema con las funciones que necesitamos y no alterarlo ya que un proyecto cada vez va haciéndose más grande y tener que cambiar una cosa que a priori parece insignificante puede dar mucho trabajo adicional.

Agradezco a ambos directores la libertad que me han otorgado para construir el sistema domótico que mejor me pareciese, y sobre todo, la resolución de todas las dudas que he tenido.

Para finalizar me gustaría comentar que estoy contento por haber escogido este proyecto y en un futuro próximo es posible que haga algún proyecto personal utilizando Arduino y Raspberry Pi ya que forman un conjunto muy potente para crear sistemas domóticos.

7. Anexos

Código

comedor.ino: programa de la placa Arduino esclava

recibidor.ino: programa de la placa Arduino maestra

Datasheet

dht.pdf: ficha técnica de los tipos de sensores DHT

dht11.pdf: ficha técnica del sensor DHT11

Seeeduino.pdf: ficha técnica de la placa Seeeduino

TTC03 Thermistor.pdf: ficha técnica del termistor analógico

Librerías

DHT

ETHER_28J60

etherShield

IRremote

NECIRrcv

Otros

códigos mando IR.txt: códigos de los botones de un mando NEC y otro Sony

Electronic Bricks Vol1.1 (pack sensores).pdf: conjunto de dispositivos acoplables a Arduino

gráfico dispersión resis_temp.slsx: documento Microsoft Excel con el gráfico de dispersión del termistor analógico

Referencias

Comunicación serie entre Arduinos:
http://www.youtube.com/watch?v=FiDaNkuwgQM

Datasheet Ethernet Shield:
http://www.nuelectronics.com/estore/index.php?main_page=project_eth

Información sobre sensores:
http://www.ladyada.net/learn/sensors/index.html

Librería IRremote:
http://www.pjrc.com/teensy/td_libs_IRremote.html

Librería NECIRrcv:
http://www.sherkhan.net/blogs/frikadas/?p=331

Página oficial de Arduino:
http://www.arduino.cc/

Tutorial Ethernet Shield 1:
http://www.instructables.com/id/Arduino-Ethernet-Shield-Tutorial/

Tutorial Ethernet Shield 2:
http://bildr.org/2011/06/arduino-ethernet-pin-control/

Tutorial Ethernet Shield 3:
http://arduino.cc/forum/index.php/topic,6595.0.html#0