Sistema de control de la iluminación de un hogar a través de Android gobernado por la plataforma Arduino

Ramón Amador Ramos
Plan de estudios: Grado de Tecnologías de Telecomunicación
Ámbito: Ingeniería de Sistemas de Telecomunicación

Área de trabajo final: Arduino

Nombre Consultor: Oriol Jaumandreu Sell
Nombre Profesor responsable de la asignatura: Pere Tuset Peiró

Enero de 2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Sistema de control de la iluminación de un hogar a través de Android gobernado por la plataforma Arduino*
Nombre del autor:	*Ramón Amador Ramos*
Nombre del consultor/a:	Oriol Jaumandreu Sell
Nombre del PRA:	Pere Tuset Peiró
Fecha de entrega (mm/aaaa):	*01/2017*
Titulación:	*Grado de Tecnologías de Telecomunicación*
Área del Trabajo Final:	*Arduino*
Idioma del trabajo:	*Castellano*
Palabras clave	*Arduino, Domótica, Iluminación, Móvil, Gestión, Hogar*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de la aplicación, metodología, resultados y conclusiones del trabajo.*

En la actualidad los Smartphones, las tabletas y otros dispositivos son muy utilizados en lo que es la era del Internet de las Cosas. Prácticamente en todos los hogares se puede encontrar al menos un dispositivo móvil con conexión a Internet.

En muchos círculos, la domótica se considera un lujo que no se encuentra al alcance de todos. Si bien es cierto que la domótica puede considerarse un lujo ya que aporta confort y comodidad en los hogares, este proyecto tiene por objetivo demostrar lo contrario y acercar la domótica a un gran número de viviendas.

Este humilde trabajo académico quiere ser un ejemplo de la facilidad con la que una vivienda puede obtener un sistema domótico adaptado a las necesidades propias y al nivel de vida de una familia. Para ello se construye y diseña un prototipo que controle y gestione el alumbrado mediante una plataforma de desarrollo de software y hardware libre como es el Arduino, incluye el sistema de detección de presencia, la autorregulación de la luz natural y el control a distancia a través de un dispositivo móvil. De esta manera veremos hasta qué punto es posible acercar la domótica a la ciudadanía, y si es factible introducir el confort en la mayoría de hogares.

Abstract (in English, 250 words or less):

Nowadays, Smartphones, tablets and other devices are very used in the internet of things (IoT). Practically in all homes you can find at least one mobile device with an Internet connection. In many circles, the home automation or smart home (also known as domotics) is considered a luxury that is not available to everyone.

While is true that home automation can be considered a luxury that provides comfort in homes, this project aims to prove otherwise and bring automation to a large number of households.

This humble academic work wants to be an example of the ease with which a home can obtain a domotic system adapted to the needs and standard of living of a family. For this, a prototype is built and designed to control and manage the basic lighting of a house through a software development platform and free hardware such as the Arduino, integrating various systems: presence detection, self-regulation of natural light and remote control through a mobile device.

In this way, we will observe to what extent it is possible to bring home automation to the citizenship, and if it is feasible to introduce comfort in most homes.

Índice

Lista de figuras

1. Introducción

1.1 Contexto

La vida viene marcada por los cambios. Los cambios son evolución y las casas también sufren evoluciones: de las antorchas y las velas, a la electricidad, de las señales de humo y el Morse, a Internet. La evolución tecnológica ha permitido mejorar el confort de los hogares, dando paso a electrodomésticos y máquinas que son capaces de realizar tareas cotidianas de manera casi automática. Estos avances no hubieran sido posibles sin la evolución de la tecnología, que mediante el hardware y el software ha permitido regular y automatizar los procesos a gran velocidad.

1.2 Domótica y eficiencia energética

Con la evolución llega la domótica, un término que proviene de las palabras "*domo*" e "*informática*", que se encarga de la integración y la regulación de los sistemas eléctricos y electrónicos, permitiendo que las casas detecten las necesidades y respondan actuando para garantizar el confort, la seguridad y el ahorro energético.

El uso de las tecnologías y de las comunicaciones en las viviendas ha generado nuevas aplicaciones y tendencias basadas en la capacidad de procesar información y en la integración y comunicación entre equipos e instalaciones. De una manera general, un sistema domótico dispondrá de una red de comunicación que permite la interconexión de una serie de equipos domésticos y, basándose en ésta, realizar unas determinadas acciones sobre este entorno. Los elementos de campo (detectores, sensores, captadores, actuadores, etc.) transmiten señales a una unidad central inteligente que tratará y elaborará la información recibida. En función de esta información y de una determinada programación, la unidad central actúa sobre determinados circuitos relacionados con las señales recogidas por los elementos de campo correspondientes. Por lo tanto, son los conceptos como automatizar, control o telecomunicación, que si los aplicamos al ámbito de la vivienda, estamos hablando de *domótica*.

Este trabajo se centra en un uso doméstico en una vivienda. Aunque con la configuración adecuada podría valer para otros inmuebles o supuestos. Haciendo uso de la *domótica* en el hogar, se puede gestionar de forma inteligente la iluminación, climatización, agua caliente sanitaria, el riego, los electrodomésticos, etc. Aprovechando mejor los recursos naturales, utilizando las tarifas horarias de menor coste, reducir la factura energética mientras se gana en confort y seguridad. Aunque no es objeto de este trabajo, la monitorización de consumos energéticos es una manera de hacer consciente al consumidor y ayuda a modificar hábitos que pueden ayudar a ahorrar energéticamente. Desde la climatización hasta el consumo de agua, todos los dispositivos y servicios domésticos se pueden gestionar y automatizar de manera que ayuden a mejorar la eficiencia energética.

1.3 Justificación del Trabajo

Este trabajo se centra en el control de iluminación básico de una estancia, es una de las formas más básica de ahorrar electricidad. El sistema de iluminación que se ha diseñado es un sistema eficiente ya que puede controlar la iluminación en función de la luz ambiente o la presencia de personas, ajustándola a las necesidades de cada momento. Por ejemplo, puede detectar la presencia de personas en zonas de paso, como los pasillos de la vivienda o de zonas comunes de una vivienda y sólo las ilumina cuando es necesario. También incluye un control mediante dispositivo móvil del encendido y apagado de las luces.

1.4 Objetivos del Trabajo

1. Diseñar un sistema gobernado por la plataforma Arduino capaz de controlar un punto de luz.
2. Programar el sistema basado en Arduino.
3. Ajustar los sensores al sistema digital.
4. Accionar circuito de potencia remotamente mediante relés.
5. Implementar una aplicación móvil de control desde el dispositivo móvil.

1.5 Enfoque y método seguido

En el inicio del proyecto se realizará una investigación y recopilación de la información necesaria para la toma de decisiones que afectaran al resto del proyecto.

Esta etapa incluirá la elección de la plataforma particular de entre las diferentes opciones de plataformas tecnológicas basadas en Arduino disponibles en el mercado. Se estudiará qué elementos necesitará el sistema para cumplir con los objetivos, priorizando un diseño modular que facilite las posteriores pruebas sobre el sistema. Terminará con la elaboración de un diseño base de alto nivel que explique cómo interactuarán los diferentes módulos para las funciones que requiere el sistema de acuerdo a los objetivos.

De ser posible se elaborará un módulo con la funcionalidad específica requerida por cada objetivo particular que se comunique con la plataforma.

Se continuará la elaboración del proyecto siguiendo un desarrollo iterativo, que permita realizar pruebas de los diferentes módulos y elementos del sistema a medida que se implementa la funcionalidad necesaria para cumplir los objetivos.

1.6 Planificación del Trabajo

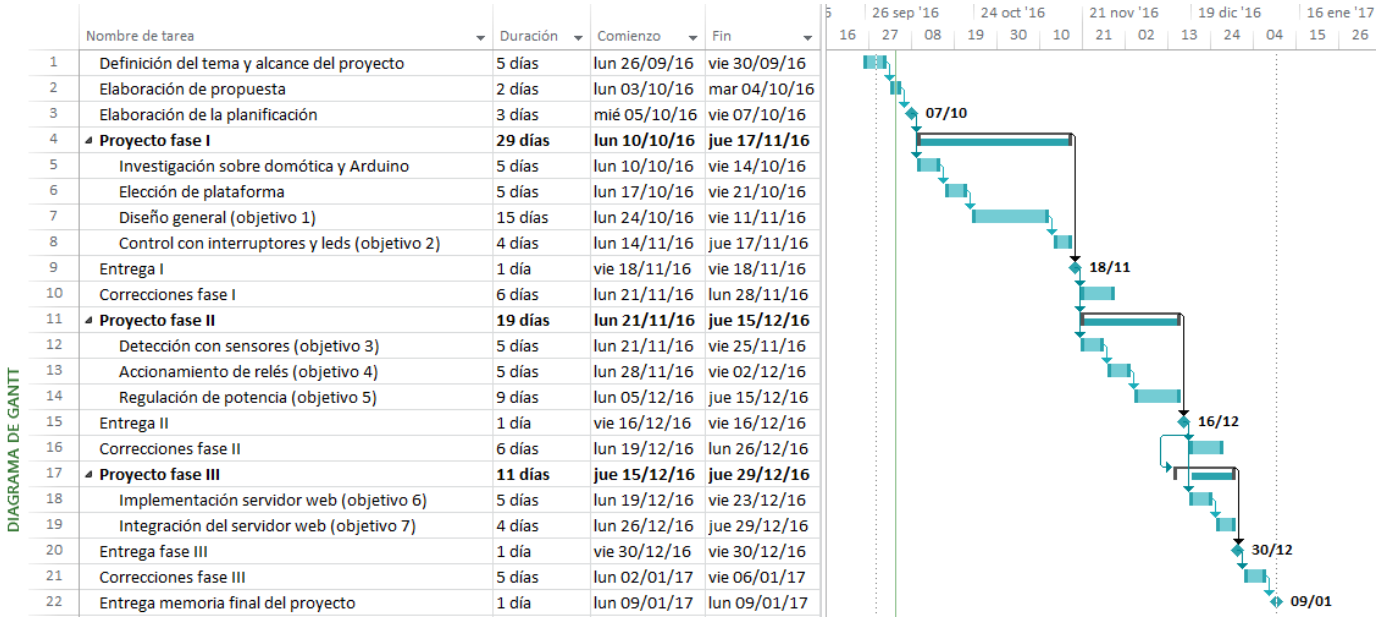
El proyecto requiere la elección de una plataforma particular de Arduino, la cual determinará las características eléctricas de los sensores, componentes e interficies desde y hacia las cuales se pueda comunicar con la plataforma. Determinará las capacidades de cálculo, almacenamiento y respuesta, y la disponibilidad de módulos con funcionalidad extra que permita cubrir las deficiencias en las capacidades de la plataforma a la hora de cumplir los objetivos planteados.

A continuación se requerirá familiarizarse con el entorno de desarrollo de la plataforma, incluyendo IDE, interfaz de programación, en caso de estar disponibles herramientas de diseño y depuración de hardware y de software. Se implementará la funcionalidad más básica controlando interruptores y leds para facilitar el desarrollo del software al minimizar los errores que se puedan introducir mediante el hardware. Al finalizar la

primera fase del proyecto se contará con una base de software funcional sobre la plataforma.

Se proseguirá añadiendo otro tipo de hardware que requiera posiblemente etapas de adaptación de niveles de potencia, de los rangos de señales eléctricas, conversión analógica/digital y/o control de sus señales de activación. Al finalizar la segunda etapa del proyecto la plataforma podrá realizar un control básico de dispositivos de iluminación.

Finalmente, en la tercera etapa del proyecto se incluirá gestión mediante una interfaz Windows y Android para obtener mayor control, comodidad y usabilidad por el usuario de los sistemas controlados por la plataforma.



	Nombre de tarea	Duración	Comienzo	Fin
1	Definición del tema y alcance del proyecto	5 días	lun 26/09/16	vie 30/09/16
2	Elaboración de propuesta	2 días	lun 03/10/16	mar 04/10/16
3	Elaboración de la planificación	3 días	mié 05/10/16	vie 07/10/16
4	**Proyecto fase I**	**29 días**	**lun 10/10/16**	**jue 17/11/16**
5	Investigación sobre domótica y Arduino	5 días	lun 10/10/16	vie 14/10/16
6	Elección de plataforma	5 días	lun 17/10/16	vie 21/10/16
7	Diseño general (objetivo 1)	15 días	lun 24/10/16	vie 11/11/16
8	Control con interruptores y leds (objetivo 2)	4 días	lun 14/11/16	jue 17/11/16
9	Entrega I	1 día	vie 18/11/16	vie 18/11/16
10	Correcciones fase I	6 días	lun 21/11/16	lun 28/11/16
11	**Proyecto fase II**	**19 días**	**lun 21/11/16**	**jue 15/12/16**
12	Detección con sensores (objetivo 3)	5 días	lun 21/11/16	vie 25/11/16
13	Accionamiento de relés (objetivo 4)	5 días	lun 28/11/16	vie 02/12/16
14	Regulación de potencia (objetivo 5)	9 días	lun 05/12/16	jue 15/12/16
15	Entrega II	1 día	vie 16/12/16	vie 16/12/16
16	Correcciones fase II	6 días	lun 19/12/16	lun 26/12/16
17	**Proyecto fase III**	**11 días**	**jue 15/12/16**	**jue 29/12/16**
18	Implementación servidor web (objetivo 6)	5 días	lun 19/12/16	vie 23/12/16
19	Integración del servidor web (objetivo 7)	4 días	lun 26/12/16	jue 29/12/16
20	Entrega fase III	1 día	vie 30/12/16	vie 30/12/16
21	Correcciones fase III	5 días	lun 02/01/17	vie 06/01/17
22	Entrega memoria final del proyecto	1 día	lun 09/01/17	lun 09/01/17

Figura 0. Gráfica de la planificación

1.7 Breve sumario de productos obtenidos

Al finalizar el proyecto se dispondrá de una plataforma Arduino controlada a través de un entorno Windows y un entorno Android que facilitará la gestión de la iluminación de un hogar. Teniendo un mejor control de la iluminación se consigue una vivienda más eficaz y eficiente, con un ahorro energético significativo.

2. Marco tecnológico

Las actuales plataformas de IoT "*el internet de las cosas*" (*Internet of Things en inglés*) provienen de una convergencia tecnológica del desarrollo de ramas tan distintas como lo son informática, microprocesadores, sistemas embebidos, control industrial, redes de ordenadores, interfaz de instrumentos digitales, dispositivos y teléfonos inteligentes, electrodomésticos inteligentes, computación en la nube, automatización del hogar (domótica) y laboratorio de fabricación.

2.1 Aparición de ordenadores

Durante siglos, diversos matemáticos especularon y diseñaron máquinas que pudiesen realizar cálculos, entre ellos se encuentran Charles Babbage con su máquina diferencial (no la pudo terminar de construir, ya que fue extremadamente difícil para una sola persona construir todos los elementos mecánicos de precisión, pero fue reconstruida en 1991). En 1827 publica el primer testimonio de lo que se puede considerar software en su ensayo «On a method of expressing by signs the action of machinery (Sobre un método para expresar mediante signos la acción de la máquina)» en su "Mechanical Notation (Notación Mecánica)". No fue sino hasta el desarrollo de las válvulas eléctricas de vacío que se pudieron construir los primeros ordenadores de utilidad práctica.

En 1937 se diseñó el primer ordenador digital electrónico automático. Fue construido por el profesor de matemática y física de la Universidad Estatal de Iowa John Vincent Atanasoff con la ayuda del estudiante graduado Clifford Berry. Fue diseñada sólo para resolver sistemas de ecuaciones lineales por lo que no era totalmente automática y fue probada satisfactoriamente en 1942.

2.2 Primer ordenador completo

Los esfuerzos de investigación e innovación de las guerras mundiales indujeron la creación de los primeros superordenadores.

El 5 de Junio de 1943 el Comando de Investigación y Desarrollo de la Armada de Estados Unidos encargó el diseño y construcción de la ENIAC (Electronic Numerical Integrator And Computer) con un costo total de 487.000 usd, equivalentes a 6.816.000 usd en 2016 a la escuela

de ingeniería eléctrica de la Universidad de Pensilvania. La máquina consumía 150 kilovatios constaba de 17468 válvulas eléctricas y 1500 relé. En su construcción se realizaron 5 millones de soldaduras realizadas a mano. Sólo contenía 20 sumadores-acumuladores con contadores de 10 dígitos decimales en anillo. Su diseño ineficiente se basó en las máquinas sumadoras electromecánicas, pero permitió encadenar las operaciones cableadas en los módulos de instrucciones («Function Tables», equivalentes a ROM) sin intervención de operaciones mecánicas y con ello acelerar la velocidad de cálculo 1000 veces. Fue la primera máquina Turing-completa, es decir, el primer ordenador universal, aunque se usó poco en esta modalidad debido a la limitada memoria de 20 números de 10 dígitos. La combinación de velocidad, memoria y reprogramabilidad permitió realizar muchos más cálculos que los que se acostumbraba en cada problema. En la elaboración de tablas de balística, ENIAC calculó trayectorias que le tomaron a humanos 20 horas, en sólo 30 segundos, incrementando 2400 veces la velocidad. Fue también utilizada en los cálculos de investigación para la elaboración de la bomba atómica.

2.3 Aplicación de la tecnología del ordenador al ámbito civil

Después de la guerra se construyeron muchos súper-ordenadores principalmente en universidades de USA e Inglaterra y surgieron importantes fabricantes, como las empresas constituyentes de IBM que provenía esencialmente de elaborar tarjetas para el control horario de los obreros y sumadoras electromecánicas.

La computación avanzó enormemente en el plano teórico, pero no había forma de extender su aplicabilidad debido a las voluminosas, costosas y calientes válvulas de vacío.

En 1926 ya Julius Lilienfeld patentó un transistor de efecto de campo, con la idea de sustituir a la válvula de vacío, aunque no pudo construir un dispositivo funcional en ese tiempo.

En 1947 se construye el primer transistor práctico en forma de transistor de punto de contacto, por los físicos John Bardeen, Walter Brattain y William Shockley. El transistor revolucionó la electrónica y abrió el

camino para dispositivos más económicos y pequeños incluyendo radios, calculadoras y ordenadores.

2.4 Primeros sistemas embebidos de control

La siguiente gran cadena de innovaciones fue impulsada por la carrera espacial de la guerra fría. La NASA (National Aeronautics and Space Administration)

Uno de los primeros sistemas embebidos modernos reconocibles fue el Ordenador de Guía del Apolo (Apollo Guidance Computer), desarrollado por Charles Stark Draper en el laboratorio de instrumentación del MIT. Utilizó el nuevo desarrollo de circuitos integrados monolíticos para reducir el tamaño y el peso. Posteriormente, un temprano sistema embebido producido en masa fue el ordenador de guía Autonetics D-17 para misil nuclear LGM-30 Minuteman.

La NASA contrató numerosos proyectos de desarrollo a Motorola, quien además incorporó la tecnología desarrollada en la elaboración de numerosos productos de consumo civil.

Los sistemas embebidos han caído de precio desde su aparición hasta el día de hoy a la vez que han incrementado drásticamente su poder de cómputo y funcionalidad.

2.5 Primer microprocesador

La mejora en la tecnología de integración produjo el primer microprocesador totalmente integrado en un único chip en 1971. Fue el Intel 4004 elaborado bajo pedido del fabricante de calculadoras japonés Busicom Corporation, con el objetivo específico de reducir costes de incluir los diferentes elementos constitutivos de una unidad CPU en varios chips. El diseño fue elaborado con Rubilith, es decir, planos a escala con máscaras del contenido del CPU con una cinta adhesiva de color rojo elaboradas a mano.

2.6 Aparición del microcontrolador

A principios de los 80, los componentes del sistema de memoria y manejo de entrada y salida ya habían sido integrados en el mismo chip que el procesador, conformando el microprocesador. Los microprocesadores encontraron aplicaciones donde un ordenador de propósito general hubiera sido muy costoso.

Un microcontrolador es un sistema integrado que incluye una CPU y hardware adicional que se usa frecuentemente en sistemas embebidos. Un microcontrolador de costo comparativamente reducido puede ser programado para cumplir el mismo rol que un número grande de componentes separados. Aunque el sistema resultante es generalmente más complejo que una solución tradicional, la mayor parte de la complejidad queda contenida dentro del propio microcontrolador. El sistema así diseñado requiere de muy pocos componentes adicionales y la mayoría del esfuerzo del diseño está ahora enfocado en el software. El prototipado y prueba de software puede ser más rápido si lo comparamos con el diseño y construcción de un nuevo circuito que no utiliza un procesador embebido.

En 1978 la Asociación Nacional de Fabricantes de Ingeniería publicó un "estándar" para microcontroladores programables. Esto propició la caída de los precios de los microprocesadores y microcontroladores e hizo factible reemplazar componentes analógicos costosos basados en diales tales como potenciómetros y condensadores variables con botones arriba/abajo o diales de lectura con microprocesador, incluso en productos de consumo.

2.7 Aparición del ordenador personal

A finales de los años 70 y principios de los 80 se desarrollan los primeros ordenadores personales de la mano de fabricantes como Altair, MITS (Microinstrumentation and Telemetry Systems – Sistemas de Micro instrumentación y Telemetría) y surgen empresas como Apple, Microsoft, Atari, Commodore, Sinclair Research Ltd ante la negativa de IBM de satisfacer las necesidades de cómputo para el hogar. A la par continúan los desarrollos de superordenadores ahora para ámbito comercial, incluyendo los avances de la Digital Equipment Corporation con la VAX, la japonesa NEC y muchas otra que surgen a medida que se comprueba el potencial de cómputo para la industria y el hogar. Los primeros ordenadores personales eran a penas «cajas de luces» como las ilustradas en las famosas películas antiguas de ciencia ficción y la primera temporada de Star Trek. Totalmente insoportables para el usuario doméstico, pero sirvieron de entrenamiento para los creadores

de las que serían las primeras compañías especializadas en ordenadores personales. Curiosamente, el procesador de los primeros ordenadores con monitor era el mismo que el utilizado en las últimas cajas de luces. Sólo faltaba un poco de creatividad y conocimiento de la señal del televisor para incrementar enormemente la capacidad gráfica y la usabilidad.

2.8 Establecimiento de la industria del ordenador personal

Ocurre una verdadera explosión del mercado cuando confluyen en el ordenador personal el modem para conectividad a equipos y sistemas de información a través de las líneas telefónicas, el disco flexible que permite introducir y almacenar datos y programar de una forma práctica y flexible, los lenguajes de alto nivel que facilitan la elaboración de programas y los paquetes de software para oficina que incluyen la hoja de cálculo o spreatsheet. Destacan en este período el Apple II y el Commodore 64 por su enorme éxito comercial. Hubo muchos intentos de «clones» que produjo tensiones por temas de patentes y que no se solucionó hasta que en la década siguiente IBM elabora el estándar IBM-PC al cual evolucionó la gran mayoría de la industria del ordenador personal.

Destaca fuera de la línea de los ordenadores IBM-compatible las versiones del Commodore Amiga A1200 y A4000 a finales de 1992. La plataforma se hizo particularmente popular por los videojuegos y las demostraciones de programación, pero también consiguió un rol preponderante en los negocios de producción de video de escritorio y control de programa televisivo. Se construyó con ella el asequible sistema de edición de video "Video Toaster" en conjunto con electrónica y equipo de grabación de video. La habilidad nativa de reproducción simultánea de múltiples muestras de sonido digital, la hicieron una plataforma popular para los primeros software "tracker". También se elaboraron dispositivos para permitirle controlar instrumentos musicales digitales.

2.9 Sistemas de control industrial basados en procesadores

En el ámbito industrial se desarrollan los sistema de control lógico programable (PLC) de fabricantes como Siemens, ABB, Echelon, Omron

y otros. Estos sistemas en lugar de seguir la vía de la programación de lenguajes de alto nivel, buscan un sistema mucho más básico y fácil de usar, basado en los diagramas eléctricos escalera que sustituyen. Se automatizan y computarizan una enorme cantidad de procesos industriales y se desarrollan nuevos métodos de manufactura mucho más precisos con el control numérico por ordenador (CNC).

2.10 Reinvención del hardware y el software

Los mismos avances tecnológicos ahora pasan a producir un ciclo de retroalimentación donde las nuevas tecnologías ya no se diseñan a mano. Las mismas tecnologías de software de la generación anterior se rediseñan incorporando el paradigma de la programación orientada a objetos posibilitando la creación de herramientas avanzadas de diseño asistido por ordenador (CAD), fabricación asistida por ordenador (CAM) y se crean herramientas de prototipado y pruebas mucho más avanzados para poder emprender la explosión en la complejidad de los nuevos sistemas, como lo son los ICE (In-Circuit Debugger – depurador en circuito), los FPGA (Field Programmable Gate Array – arreglo de puertas de campos programables) y otros. En el ámbito industrial se desarrollan los sistemas SCADA (Supervisory Control And Data Adquisition).

2.11 Aparición de Internet

Los acelerados avances científicos, tecnológicos y económicos llevan a la aplicación civil de la red de ordenadores ARPANET. Inicialmente concebida como un sistema de soporte de control descentralizado para el ejército de Estados Unidos en caso de un ataque nuclear, incorpora a los departamentos de investigación de las principales universidades de Estados Unidos. Hasta el momento, las redes de datos entre ordenadores se basaban en el establecimiento de conexiones a través de las mismas conexiones telefónicas que se utilizaban para conversaciones de voz consistentes de una serie de circuitos conmutados. En la cultura informática de las universidades se crearon comunidades de voluntarios que prestaban sus ordenadores y líneas telefónicas para actuar como servidores de mensajería similar al correo electrónico durante las horas de la madrugada. El servicio atendía

llamadas de corta duración provenientes de equipos de desconocidos con los que se entregaban y recibían los mensajes correspondientes a dicha persona. Se establecía y actualizaba manualmente pasando de voluntario a voluntario una red jerárquica de números de teléfono por región para intercambiar mensajes a mayor distancia. ARPANET creó la tecnología de conmutación de paquetes que sustituyó el establecimiento de las conexiones telefónicas tradicionales estableciendo las bases de Internet. Se presentan 4 hitos importantes en el uso de Internet desde el punto de vista de este proyecto:

1) El descubrimiento en el ámbito público de los sistemas de encriptación de llave pública en 1976 (anteriormente se había descubierto en el ámbito clasificador por el gobierno de Estados Unidos), por Whitfield Diffie y Martin Hellman, que posteriormente permitió nuevos sistemas de encriptación que son la base del comercio electrónico y el acceso a recursos en la nube.

2) La creación de la World Wide Web en 1989 concebida por Tim Berners Lee, que facilita la organización, interconexión y visualización de grandes cantidades de información a través de páginas web.

3) El lanzamiento de GNU/Linux en 1991 saca a la luz la capacidad de la colaboración voluntaria masiva en el desarrollo de software libre y se constituye en la principal plataforma de desarrollo de las tecnologías abiertas.

4) La creación del primer Smartphone (teléfono inteligente) de amplia aceptación en 1999 por la japonesa NTT DoCoMo, inicia la convergencia de la tecnología celular, internet y la funcionalidad antiguamente asociada a un ordenador personal.

2.12 Situación tecnológica actual

Nos encontramos cerca del final de la expansión de la microelectrónica, que creció durante todo este tiempo de forma exponencial como describió Gorgon Moore el 19 de abril de 1965. Siendo co-fundador de Intel, observó cómo la sinergia de los avances en microelectrónica permitían empaquetar cada vez más transistores, trazando una recta en una gráfica logarítmica de densidad de transistores vs tiempo. La pendiente de esta recta es la que permitió la famosa declaración sobre la

duplicación de la cantidad de transistores cada 18 meses, declaración que ha sido corregida en algunas ocasiones a 12 meses o a 24 meses, dependiendo de las condiciones económicas y de descubrimientos tecnológicos de cada período.

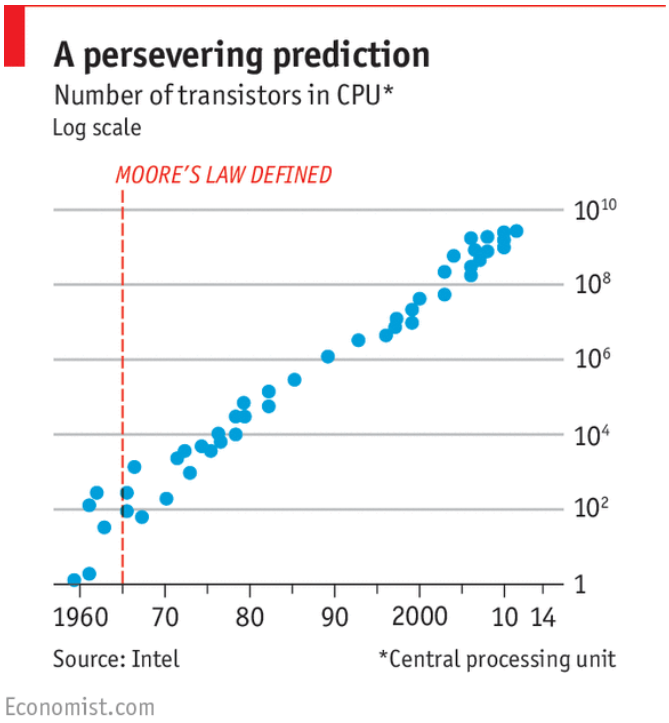


Figura 1. Evolución número transistores

La gráfica es analizada en un artículo the The Economist: http://www.economist.com/blogs/economist-explains/2015/04/economist-explains-17

El artículo entra en el análisis de los factores económicos que arrojan luz para afirmar que está próximo un cambio de tendencia. Los costos de producción suben exponencialmente al intentar continuar con los avances en miniaturización. No hay progresión tecnológica previsible, ni razones de valor añadido al producto que puedan justificar el nivel de inversión económica que se requiere para mantener el actual ritmo en el campo de los transistores más allá de pocas generaciones tecnológicas.

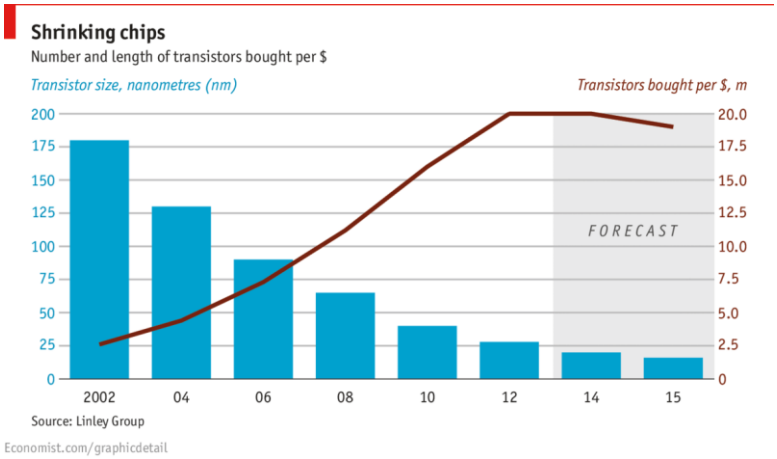


Figura 2. Evolución coste transistores

La gráfica refleja cómo se han incrementado los costes de miniaturización en los últimos procesos de fabricación de transistores con ancho de canal de 14 nm en los últimos años.

Las tecnologías sustitutivas del transistor como la computación cuántica, se encuentran muy lejos de alcanzar aplicación práctica a bajo costo.

Más del 95% de los procesadores que se fabrican actualmente no están destinados al mercado de ordenadores, sino a electrónica de consumo y este mercado sigue creciendo especialmente en el área de teléfonos inteligentes con previsiones del 7% anual con 1500 millones de teléfonos fabricados para 2016 de acuerdo a Gartner Group:

http://www.gartner.com/newsroom/id/3270418

Se espera que continúen los desarrollos tecnológicos en el área de arquitectura y diseño de circuitos con diseños híbridos que se adapten a múltiples tipos de procesamiento como las unidades de procesamiento general que pueden incorporar procesamiento de cómputo tradicional y de gráficos como las GPCPU del fabricante AMD.

2.13 Teléfonos inteligentes

La mayoría de nuevos diseños de teléfonos inteligentes incluyen procesadores multinúcleo y capacidad multimedia. Es especialmente evidente la facilidad de convergencia de las tecnologías de teléfonos inteligentes con las tecnologías del «Internet de las Cosas» o IoT, por su portabilidad, conectividad y eficiencia energética. Donde se diferencian estos mercados es en el coste y en la complejidad de procesamiento.

Los requerimientos de procesamiento en IoT son muy inferiores a los de aplicaciones móviles y navegación web «responsive», es decir, aplicaciones interactivas y aplicaciones que se adaptan a diferentes tamaños de pantalla con gran atractivo visual.

2.14 Sinergia entre teléfonos inteligentes e IoT

Lo más destacado de la convergencia tecnológica móvil para el punto de vista de este trabajo, no es que los dispositivos IoT asuman las funciones de los teléfonos inteligentes. Así como los primeros ordenadores y consolas de videojuegos se valieron de los televisores para reducir costos y poder brindar una interfaz cómoda y muy funcional, se puede predecir que en un futuro próximo la mayoría de dispositivos y electrodomésticos en el hogar dispondrán de una aplicación móvil desde la cual brinden información y permitan su configuración y control, sin incurrir en ningún coste añadido más allá del necesario para brindarles conectividad.

2.15 Control de un dispositivo eléctrico sencillo del hogar

Muchos dispositivos eléctricos del hogar han evolucionado para incorporar microcontroladores y ofrecer mayor rendimiento o comodidad en su uso. Otros, como ocurre con un sistema sencillo de iluminación, han permanecido ajenos a todos los sistemas de control y comunicación de bajo costo que han invadido el hogar.

Enfrentaremos estos dispositivos que presentan un funcionamiento muy sencillo a la tecnología moderna para controlarlos cómoda y económicamente.

A día de hoy los mayores pedidos son de un microprocesador basado en ARM Cortex53 de 4 núcleos a 1.2GHz equivalente al del Raspberry Pi 3 en el distribuidor DigiKey tiene un costo de 23 euros la unidad. En cambio el procesador de AVR ATMega328P a 20MHz tiene un costo de 3,75 euros la unidad, el mismo precio con el que se puede obtener la memoria microSD de 2GB Toogo en Amazon España, necesaria para almacenar el sistema operativo Raspian completo con interfaz de escritorio en caso de querer utilizar toda la funcionalidad de la plataforma Raspberry Pi 3.

El AVR ATMega328P es el microcontrolador que utiliza el sistema Arduino UNO revisión 3. Puede ser competitivo respecto al Raspberry Pi a pesar de tener una frecuencia de reloj 60 veces menor, 1 sólo núcleo y procesar de a 8 bits en lugar de a 64 bits porque en su rango de aplicación no se requiere un sistema operativo multitarea, no se requiere manejo de memoria externa y en general no se requieren las funcionalidades adicionales de la plataforma Raspberry Pi 3. La mayoría de los sistemas embebidos no incorporan una interfaz gráfica de usuario. Como se mencionó anteriormente sobre la sinergia entre los teléfonos inteligentes e IoT, ni siquiera es necesario generar una señal de video si se comunican los datos necesarios a una aplicación móvil para brindar una interfaz gráfica de usuario.

En ambos casos se requiere un dispositivo controlador de Ethernet 10/100 para otorgarle conectividad.

En el caso del Ethenert Shield de Arduino, el controlador es el Wiznet W5100 cuya primera versión salió al mercado en diciembre de 2006. Al día de hoy la última versión disponible en DigiKey se puede obtener al mayor por menos de 4 euros. Este controlador empaquetado en un chip permite comunicarse a Internet programando sólo los aspectos de la capa de aplicación que configuran y comunican los sockets. Todas los detalles del funcionamiento interno de los sockets, la capa de red, de enlace de datos y manejo de la señal física que soportan el funcionamiento de Internet de acuerdo al modelo OSI (siglas de "Open Systems Interconnection model" - "modelo de Interconexión de Sistemas Abiertos") son manejas por el chip y quedan ocultas para el resto del sistema.

Muchas aplicaciones que se han desarrollado actualmente en Raspberry Pi en su versión actual o en anteriores utilizando Python, se pueden trasladar a Arduino Uno R3. Python es un lenguaje moderno multiparadigma que se ejecuta generalmente bajo un intérprete. El proceso de interpretación de un lenguaje de programación es un proceso lento. Es común lograr aceleraciones mayores al orden de magnitud al trasladar una aplicación en un lenguaje interpretado a un lenguaje compilado. Arduino utiliza la cadena de herramientas de compilación de

GNU para soportar los lenguajes C y C++. En los principios de C++ los diseñadores de sistemas embebidos no soportaban dicho lenguaje, porque el código generado por los primeros compiladores era más pesado de lo necesario. Hoy en día los compiladores de C++ están altamente optimizados y no se puede pasar por alto el aumento de la productividad y mantenibilidad que brinda la programación orientada a objetos en el desarrollo de software.

3. Tecnología del hogar

3.1 Base eléctrica

La llegada de la electricidad tuvo su mayor impacto en la calidad de vida de las personas a través de los numerosos aparatos a los que dio vida en los hogares. Hace ya varias generaciones cuando la principal fuente de energía para realizar las labores domésticas, como hacer la colada, era el trabajo manual. Los primeros aparatos eléctricos, aunque carentes de toda «inteligencia» como podríamos entenderla hoy en día, lograron reducir en más de un 50% el tiempo requerido para atender las necesidades del hogar. Además de la maquina lavadora de ropa, llegaron la aspiradora, la plancha eléctrica, la licuadora y otros en una primera etapa. Con la mejora de los procesos de fabricación y la electrónica vino una segunda etapa de aparatos con sistemas de control básico incluyendo el refrigerador, el secador de cabello, el horno microondas, el radio sin baterías, el televisor y con este último se produjo un estallido en el desarrollo de numerosos aparatos, muchas veces innecesarios, gracias a la economía de consumo que creció alrededor de la difusión masiva de publicidad.

La llegada de los microcontroladores de bajo precio transformó todos estos dispositivos para hacerlos funcionar más «inteligentemente». La máquina lavadora de ropa muestra opciones de ciclos de lavado, temperatura, cantidad de detergente y suavizante adecuados para cada tipo de ropa. El refrigerador con heladera incluye ciclo anti escarcha. El microondas incluye funciones para descongelar, hacer pizzas y palomitas de maíz. El televisor sintoniza los canales automáticamente, recuerda en qué canales hay una señal útil para mostrar al usuario y

hasta recuerda los canales favoritos. En un hogar promedio de clase media es possible encontrar 40 microcontroladores "escondidos" en todos los aparatos eléctricos y electrónicos.

Mientras se producía la explosión tecnológica alrededor de los ordenadores y de internet, los demás aparatos y tecnologías de la vida cotidiana fueron mejorando paulatinamente sin mayores cambios, a excepción de aquellos relacionados con la telefonía y los medios con cualquier combinación de elementos auditivos y visuales.

Cuando ocurre la llegada de los teléfonos inteligentes, todos los medios ya tenían en alguna proporción un componente digital, los ordenadores se hicieron tan competentes como cualquier aparato analógico de electrónica de consumo para manejar cualquier tipo de recurso multimedia e internet de banda ancha penetró a la mayoría de la población de los países desarrollados.

3.2 Domótica

La disposición de todas estas tecnologías despertaron el interés por conseguir soluciones comerciales que explotaran las posibilidades de una etapa de automatización del hogar. Grandes compañías tecnológicas comenzaron a investigar ideas que eran sólo cosa de aficionados a la electrónica y al DIY (siglas en inglés del "Do It Yourself" que se refiere a los "manitas" que les gusta construir cosas por sí mismos).

Se puede consultar la siguiente etimología y definición de domótica en su versión en inglés consultada en noviembre de 2016 siguiendo el enlace:

http://www.smart-homes.nl/Domotica.aspx?lang=en-US

"La palabra domótica es una contracción de la palabra latina para hogar «domus» y las palabras informática, telemática y robótica.”

Definición de los autores del sitio web:

La integración de tecnología y servicios para una mejor calidad de vida"

Para introducirnos en el contexto y significado actual de domótica se acude a algunas ideas resumidas y traducidas del libro "Inside the Smart Home" (en español "Dentro de la Casa Inteligente"), Springer 2003 Dr. Richard Harper",

http://www.springer.com/la/book/9781852336882
que incluye un capítulo sobre un estudio de caso realizado por Orange sobre la aplicación de las tecnologías de punta en la electrónica de consumo de 1999 a un hogar experimental que fue habitado por 3 familias durante un período de hasta 2 semanas.
Del primer capítulo "Inside the Smart Home: Ideas, Possibilities and Methods" (en español "Dentro de la Casa Inteligente: Ideas, Posibilidades y Métodos").
"...
Después de todo, no es como el lugar de trabajo donde obtienes una planificación, mantenimiento y – lo más importante de todo – soporte técnico. Las familias, después de todo, no están estructuradas como las organizaciones. Para hacer las cosas aún peor, «los usuarios» van desde bebés hasta pensionistas de avanzada edad. Y finalmente, los usuarios del hogar son difíciles de estudiar en cualquier forma: ¿qué familia podría querer observadores pasando alrededor todo el día y (quizá) toda la noche?" página 2.
En la misma página señala como principales funciones dentro del hogar a ser mejoradas o interconectadas a un servidor: iluminación, calefacción, aire acondicionado, seguridad, sistemas audio-visuales, cortinas y baños. En el capítulo sobre el caso de estudio se incorporan funciones de seguridad, monitorización de la salud, comunicación multimedia, teleconferencia, educación online, utilización integrada de pantallas para ordenador, reproducción de películas y sintonización de canales, y otras menos exitosas como la bañera, la lavadora y la nevera.
En la siguiente página marca como una de las metas de un adecuado diseño que la ubicación de los dispositivos interactivos necesita estar relacionada con los patrones de uso del espacio en el hogar, pero que los usuarios prefieren tocar el control físico de volumen en el equipo de sonido de alta fidelidad a utilizar el «stylus» (lápiz electrónico) en la pantalla de los PDAs (siglas en inglés de asistente digital electrónico, es como un teléfono inteligente sin la función de teléfono).
En la página 4 señala la conveniencia de los comandos de voz y a la vez el estado decepcionante y poco útil de la tecnología probada.

El resto de los capítulos incluyen diversos análisis relacionados con los aspectos sociales del uso de la tecnología, un recuento de diversas tecnologías hoy en día obsoletas, un capítulo dedicado al tema de la discapacidad y cómo abordarlo con tecnologías de automatización.

El último capítulo del libro trata sobre los desaciertos de los fabricantes de tecnologías para la automatización del hogar y cómo la competencia por tratar de imponer estándares cerrados ha obstaculizado el progreso efectivo.

La adopción de las tecnologías de automatización del hogar posiblemente sea mucho más efectiva si se sustenta sobre estándares abiertos y flexibles que permitan mejoras progresivas a la tecnología y la integración entre dispositivos y controles de fabricantes distintos a muy bajo costo.

3.3 Clasificación de hogares inteligentes

Resulta útil definir un contexto para estudiar problemas específicos y sus posibles soluciones. En el mismo capítulo sobre el caso de estudio se explican los siguientes tipos de hogares inteligentes:

1. Los que contienen objetos inteligentes. Contienen aparatos y objetos individuales, autosuficientes que funcionan de forma inteligente.
2. Los que contienen objetos inteligentes, intercomunicados. Contienen aparatos y objetos que funcionan inteligentemente por sí mismos e intercambian información entre ellos para incrementar la funcionalidad.
3. Hogar conectado. El hogar posee redes internas y externas, permitiendo el control interactivo de sistemas y acceso a servicios e información de dentro y de fuera del hogar.
4. Hogar aprendiz. Se graban los patrones de uso y los datos acumulados se utilizan para anticipar las necesidades de los usuarios. Ver por ejemplo, la Casa Adaptativa (Mozer, 1998), la cual aprende los patrones de uso de iluminación y calefacción.
5. Hogar alerta. Las actividad de las personas y objetos dentro del hogar son registradas constantemente y esta información se utiliza

para anticipar las necesidades de los usuarios. Ver por ejemplo, la Casa Sensible (Kidd et al., 1999).

4. Plataforma Arduino

4.1 Inicios Arduino

Figura 3. Placa Arduino UNO

La primera versión de Arduino fue lanzada en 2005, se basó en un microcontrolador de 8 bits Atmel AVR, sin puerto de comunicación USB. El objetivo fue proveer una forma de bajo costo a los novatos y profesionales para crear dispositivos que interactuaran con el ambiente utilizando sensores y actuadores. Los ejemplos más comunes de dispositivos para novatos incluyen robots simples, termostatos y detectores de movimiento.

Estos sistemas disponen de un conjunto de terminales de entradas y salidas que pueden servir de interfaz a varias placas de expansión denominadas shields (palabra del inglés que significa escudo) y a otros circuitos. Muchas placas se pueden direccionar individualmente a través de los terminales correspondientes al bus I2C por lo cual se pueden apilar varias placas una sobre otra y usarlas paralelamente. Los microcontroladores destinados para esta plataforma en precargado en la memoria Flash un programa denominado "bootloader" (del inglés cargador de arranque) y un programa controlador de comunicaciones en el caso de una placa que disponga del microcontrolador adicional para gestionar la conversión del protocolo de comunicaciones RS232 (reducida a rango de voltaje de 0 a 5V) hacia el protocolo USB con el

que se suele conectar a un ordenador host para recibir el programa de usuario. Antes de 2014 las placas oficiales de Arduino habían usado los microprocesadores de la serie megaAVR de Atmel: ATmega8, ATmega168, ATmega328, ATmega1280, ATmega2560. A partir de octubre de 2012, se incorporaron nuevos modelos de placas de desarrollo que usan microcontroladores Cortex M3, ARM de 32 bits. ARM y AVR no son plataformas compatibles en cuanto a su arquitectura y por lo que tampoco lo es su set de instrucciones, pero se pueden programar y compilar bajo el IDE predeterminado de Arduino sin ningún cambio.

Las placas Galileo de Intel implementan una tecnología basada en su arquitectura x86 con el nuevo Quark SoC X1000. Requirió traducir las librerías de Arduino para darles soporte. Son placas con una gran cantidad de periféricos añadidos, gigabytes de memoria principal, soporte de un bus mini-PCIe. Presenta las conexión para ser compatible con los shields de Arduino.

4.2 Entorno de desarrollo de Arduino

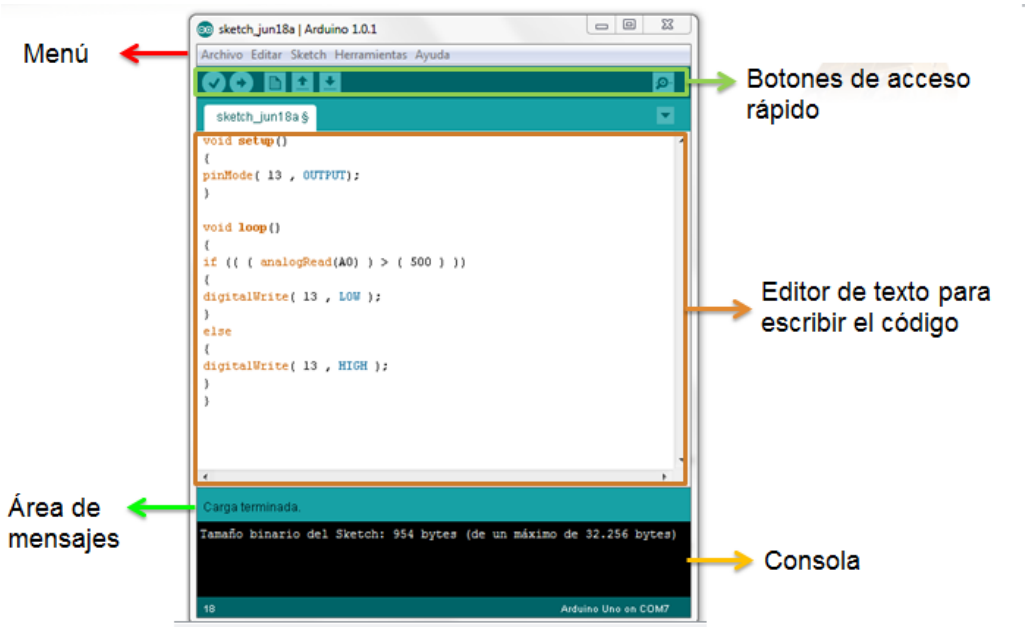


Figura 4. Entorno de desarrollo de Arduino

La plataforma Arduino provee herramientas de software adaptadas para facilitar el desarrollo, incluyendo la cadena de herramientas de compilación de GNU y un ambiente de desarrollo integrado (siglas en

inglés IDE) basado en el lenguaje de programación Processing con cambios de sintaxis al estilo de C/C++. No soporta todas las características de C++. El nombre de los archivos de código fuente es sketch. Existe una modificación en la estructura de un programa en un archivo sketch que lo hace distinto al C y C++ estándares. La función "main" ya viene predefinida en las librerías de Arduino, donde gestiona el inicio del bootloader y de las de la comunicaciones. Hay dos «puntos de enganche» entre el software producido por el usuario y la funcionalidad de software precargada, son las funciones llamadas «setup» y «loop». La primera se ejecuta una sola vez desde que se reinicia el microcontrolador y segunda «loop» se ejecuta y se repite mientras cada vez que se termina de ejecutar la última instrucción de la función.

Processing proviene de Java, en consecuencia lo primero que se requiere instalar es una versión suficientemente reciente de la máquina virtual de java.

Los programas compilados con Arduino (salvo en las placas con CorteX M3) se enlazan contra AVR Libc por lo que tienen acceso a algunas de sus funciones. AVR Libc es un proyecto de software libre con el objetivo de proporcionar una biblioteca de ambiente de tiempo de ejecución de C de alta calidad para utilizarse con el compilador GCC sobre microcontroladores Atmel AVR. Se compone de 3 partes: avr-binutils, avr-gcc, avr-libc.

La mayoría del lenguaje de programación Arduino está escrita con constantes y funciones de AVR y ciertas funcionalidades sólo se pueden obtener haciendo uso de AVR.

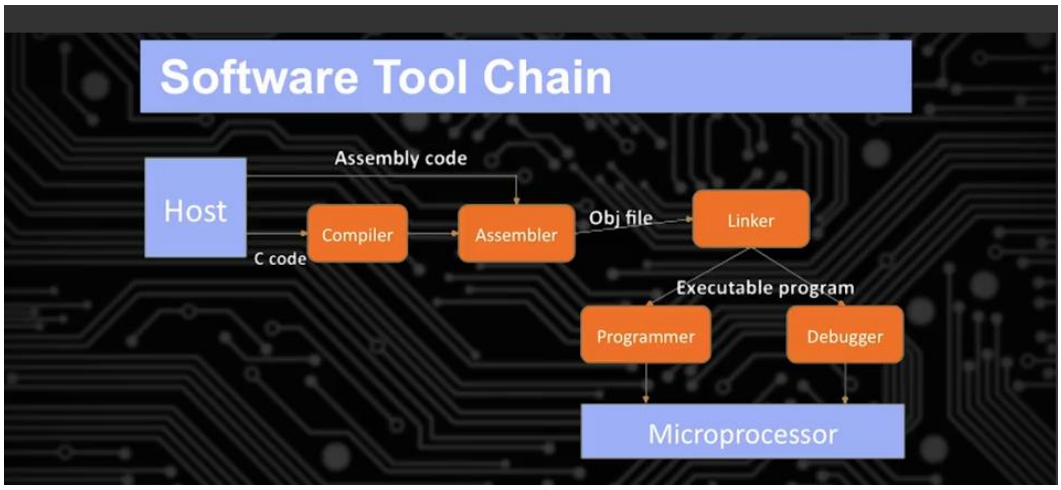


Figura 5. Gráfica AVR

El proceso de compilación es casi el mismo que el proceso con las herramientas de compilación de GNU tomando en consideración lo mencionado sobre AVR Libc y el uso de arvdude en caso de utilizar el puerto de ICSP (In-Circuit Serial Programming) o la comunicación con el bootloader.

El proceso es el siguiente:

1) El usuario edita un sketch en el ordenador host, definiendo al menos las funciones setup y loop.

2) El preprocesador expande todas las macros, carga definiciones de símbolos y reúne el contenido que necesite de todos los archivos enlazados.

3) El compilador revisa toda la estructura sintáctica y comprueba que todas las definiciones y referencias se corresponden con las llamadas de uso. Realiza optimizaciones en el código en una representación intermedia.

4) El código intermedio pasa al ensamblador donde se traduce en código objeto

5) El código objeto pasa al enlazador que se encarga de localizar los diferentes componentes del programa y las librerías que referencia en un espacio de memoria y luego resolver la direcciones de las referencias.

Ahora obtenemos el programa ejecutable que en caso de quererlo cargar en un microcontrolador con bootloader debemos asegurarnos de

que el campo serial.download_rate dentro del archivo preferences del IDE coincide con el valor configurado del bootloader. Los primeros Arduinos tuvieron un bootloader configurado a 9600 baudios. Los actuales están configurados a 19200 baudios. Dependiendo de la configuración, el bootloader puede requerir entre 1KB y 2KB de memoria Flash. Para el Arduino UNO R3 esto no es tan significativo porque posee 32KB de memoria Flash. Hay que asegurar que dentro de los 8 segundos después de reiniciar el microcontrolador, éste no reciba ninguna señal al puerto serial mientras se ejecuta el bootloader.

Si se quiere eliminar el bootloader o cargar el programa en un microprocesador que no lo contiene, se requiere un adaptador para conectar al puerto de programación ICSP y se requiere convertir el archivo con el arvdude antes de mandarlo a este puerto. Es posible programar un Arduino con bootloader para que realice las funciones de adaptador ICSP y cargue un bootloader en un microcontrolador sin bootloader.

El ambiente de ejecución del microcontrolador resulta distinto al del ambiente estándar de ejecución de un ordenador estándar. Esta diferencia produce problemas a la hora de realizar la optimización de compilación en el compilador de C++ de gcc. Especialmente se debe cuidar el consumo de memoria, más allá de lo que se haría en un ordenador, no sólo porque en el ordenador generalmente sobra memoria, sino también porque no está disponible la optimización automática del compilador como aquella que remueve del ejecutable el código no utilizado o no accesible desde ningún punto de entrada de ejecución al programa.

Se tiene 3 espacios de memoria distintos:

- Flash (32KB)
- EEPROM (1KB)
- SRAM (2KB)

Dentro del chip del microcontrolador ATMega328. La memoria SRAM es la más rápida, tiene una vida útil prácticamente ilimitada y por defecto es donde el compilador almacena todas las variables. Su principal inconveniente es que se destruye su contenido cuando el

microcontrolador se queda sin energía. Los otros 2 tipos de memoria son no volátiles, con tiempo de retención superior a 90 años, aunque son memorias con una vida reducida en número de ciclos de escritura, siendo 10.000 para la Flash y 100.000 para la EEPROM. La Flash es donde se encuentra el bootloader y donde éste carga el sketch que le transfiere el Arduino IDE por el puerto serial, se puede escribir en ella durante el tiempo de compilación como parte del espacio de memoria que el compilador organiza con su variable Progmem y aunque su vida es limitada, de ser necesario se puede escribir durante la ejecución del programa escribiéndola por bloques. La EEPROM se encuentra en su propio espacio de memoria que se encuentra protegido por fusibles virtuales. Se deben activar en secuencia para evitar errores de falsas escrituras cuando falla la energía que alimenta al microcontrolador. No debe escribirse habitualmente en ella, está pensada para escribirse sólo durante raras ocasiones, como establecer valores de configuración, o mantener un registro de datos de escritura muy poco frecuentes.

4.3 Herramienta de depuración de Arduino

Resulta muy ineficiente y difícil la simulación de microcontroladores de una arquitectura distinta a la propia del ordenador donde se quiere realizar dicha simulación. Existen soluciones de co-simulación, como la aportada por *Visualmicro*, donde se instrumenta el código fuente de un sketch para reportar el estado de ejecución del programa. El software transforma el código fuente en una versión de depuración que permite detener la ejecución ante determinados eventos y realizar tareas de depuración y diagnóstico, modificar valores en memoria y en registros del microcontrolador mientras se mantiene pausada la ejecución del código principal y retomar la ejecución como si se tratase de la ejecución normal del programa. El control y la información para realizar estas tareas fluye por el puerto de comunicaciones serial. Se puede adivinar que la primera limitación de este entorno de depuración es no poder utilizar simultáneamente el puerto serial del microcontrolador mientras se realiza la depuración. El segundo inconveniente de este sistema es el tiempo de procesamiento que requieren las instrucciones “invisibles” adicionales que se ejecutan para posibilitar la depuración. Dadas las

limitaciones de depuración del Arduino IDE resulta recomendable comprar este software para cualquier proyecto de una complejidad similar o superior al actual.

Sin embargo se utilizará otra herramienta de desarrollo, el *Circuit Explorer* de la *Suite Autodesk Circuits*. Es un entorno de emulación integrado online. Resulta significativamente lento, pero usable, debido a que la emulación no requiere emerger los comportamientos que queremos estudiar de los circuitos a partir de la simulación dinámica de características de voltaje *versus* corriente de cada componente interconectado. Por el contrario, describe comportamientos predefinidos para las conexiones más comunes. Algunas veces estos comportamientos predefinidos resultan totalmente descorrelacionados con la operación normal que se espera de los circuitos. No resulta adecuado en situaciones como en la simulación de circuitos de filtro de frecuencias.

Este enfoque de emulación lo emplea también la herramienta comercial *Virtual Breadboard*. Esta herramienta podría ser mejor que la *Suite Autodesk Circuits* ya que no está limitada a su uso en línea.

4.4 Consideraciones sobre el diseño en el entorno Arduino

Debido a la dificultad de incorporar el Arduino en un sistema de simulación de circuitos eléctricos y electrónicos (tanto por requerir un uso avanzado de software de simulación de señales mixtas digital/analógicas, como por el costo de las licencias de dicho software) el trabajo intentará limitar el procesamiento de señales analógicas y uso de módulo compatibles a los disponibles en el emulador.

La placa de desarrollo a utilizar es la Arduino UNO R3 por su facilidad de interconexión a módulos compatibles con Arduino de tipo “*Shield*” ya mencionados y también a componente de prototipado de tecnología DIP “*through hole*” mediante cables de tipo “*jumper*” a un *protoboard*. Esto no significa que un producto comercial que se pueda desarrollar en un futuro utilice estas tecnologías. Fácilmente se podrán reducir costos, tamaño y consumo de energía si al tener el prototipo de un diseño basado en módulos de Arduino se realiza un módulo “*barebone*” personalizado con tecnología SOIC que integre los elementos mínimos

utilizados. El consumo eléctrico típico de la placa Arduino UNO R3 es de 45 mA. Si por ejemplo, el producto final fuese a operar por baterías, se podría reducir a menos de una cuarta parte utilizando una versión de bajo consumo y de menor tamaño, y prescindir de la fuente de alimentación, debido al amplio margen de voltajes en los que puede operar. En otro caso, se podría por ejemplo diseñar una placa que integre el chip de red de Ethernet del módulo de Ethernet sin necesidad de tanto volumen de conexiones ni de los elementos de la controladora del puerto de tarjetas microSD.

Las posibilidades de reducción de costos, tamaño y consumo energético dependen hasta cierto punto de una utilización efectiva y eficiente de los recursos disponibles. Los microcontroladores tienen modos de ahorro energético, donde se desconecta la energía de la mayoría de los periféricos y se paraliza el procesador hasta la llegada de una interrupción externa o generada internamente. Ejemplo de uso de esta funcionalidad incluye la instalación de sensores remotos donde sólo se requiere realizar una determinada medición cada hora. En dichas situaciones se programa el microcontrolador para salir del estado de hibernación cada hora, realizar la medición, guardar el valor y volver a hibernación. Otra posibilidad de reducción del consumo energético es reducir la frecuencia de reloj, en caso típico de utilización de esta característica, de 16MHz (frecuencia máxima del ATMega328 20MHz) a 1MHz, la cual depende directamente de que el tiempo de respuesta requerido para una aplicación sea mucho mayor que el tiempo que le toma al microcontrolador ejecutar todas las tareas programadas.

Otro objetivo deseable que a veces puede estar en contra del objetivo de la utilización más eficiente de los recursos de hardware es que el software sea reutilizable, inteligible y adaptable.

Durante las primeras etapas de desarrollo, se puede dar prioridad a la inteligibilidad del software. A medida que se vaya entendiendo y probando se puede refactorizar para adaptarlo a funcionar con otros elementos de software y finalmente, si la aplicación lo requiere optimizarlo.

Se hará uso de librerías libres ampliamente probadas para desarrollo en Arduino con el fin de acortar el desarrollo y la depuración de elementos de funcionalidad que no aportarían mejoras significativas si se desarrollaran por propia cuenta.

Si la extensión temporal de este proyecto fuese lo suficientemente amplia cabría la posibilidad de desarrollar una librería abierta, ponerla a disposición de la comunidad de desarrollo de Arduino para recibir contribuciones e integrarla en otros proyectos.

El uso de librerías sin tener habilitadas opciones avanzadas de optimización del compilador puede llevar rápidamente a agotar la memoria Flash disponible del microcontrolador. La primera estrategia de control sobre el consumo de la memoria Flash consiste en hacer seguimiento de las librerías utilizadas y en caso de sustituir una librería por otra o dejar de utilizar las funciones de la librería, comentar o eliminar la inclusión de dicha librería en el código del sketch. Algunas librerías son especialmente largas, como las que se utilizan para manejar el módulo de expansión (*Shield*) de WiFi que consume casi la mitad de la memoria Flash disponible. En un caso extremo, si se necesita espacio y se utiliza sólo una parte de la funcionalidad que aporta una librería, es posible editar la librería para hacer una versión reducida que incluya sola la funcionalidad utilizada.

El otro tipo de espacio de memoria que más frecuente y encarecidamente se puede necesitar optimizar es el relativo a la SRAM, donde por defecto el compilador ubica todas las variables, sean globales, de pila (*stack*), o reservada de memoria de la pila. Se dispone de una función que reporta la memoria disponible aunque esta limita a reportar sólo el espacio contiguo de memoria. Se debe limitar la profundidad de las secuencias de llamadas a funciones y especialmente las llamadas recursivas. No se cuenta con la optimización típica derivada de la programación funcional relativa a reciclar el marco de pila (*stack frame*) de las llamadas recursivas en cola. Una forma de optimización bastante eficaz consiste en limitar las variables al tamaño mínimo necesario para contener los valores que el programa puede introducir en

ellas, aunque dependiendo de la ocasión puede producir errores de desbordamiento difíciles de depurar.

Una optimización que reduce el incremento de la profundidad de la pila y a la vez acelera la velocidad de ejecución del programa consiste en la incrustación de llamada de funciones en línea. Un ejemplo de esta técnica aplicada es la versión optimizada de la librería *TimerOne* de control del temporizador

https://github.com/PaulStoffregen/TimerOne/blob/master/TimerOne.h

No se utilizará esta versión de la librería, porque como se puede evidenciar en fragmentos comentados del código de la librería, el código generado resulta muy difícil de depurar.

El compilador dispone de una macro “F()” para ubicar en espacio de memoria Flash (*Progmem*) valores constantes que normalmente ubicaría en el espacio de memoria SRAM. Esta técnica resulta especialmente conveniente con la asignación de valores de cadenas de caracteres (*String*).

Otras técnicas de optimización como la generación de “tablas de salto” (*jump tables*), requieren de la manipulación de código ensamblado en coordinación con el código generado por el compilador y se encuentran fuera del alcance del proyecto.

El tamaño del nombre de las variables no tiene ninguna influencia ni sobre el espacio de memoria del código generado, ni sobre la velocidad de ejecución del mismo. Al tratarse de código compilado todos los nombres de variables son reasignados por el compilador a direcciones de memoria, apuntadores y referencias de su tabla de símbolos interna. Se puede por tanto, asignar el nombre de las variables a según convenga, tan largos como se necesite expresar un código suficientemente descriptivo para hacer el código inteligible y tan corto como se permita resumir en pocos caracteres sin perder la idea conceptual que encierra la variable a modo de reducir la cantidad de texto que es necesario leer para comprender el código.

Resulta de particular importancia la utilización de las recién mencionadas referencias. En lenguaje C el paso de parámetros de llamadas a funciones y la devolución del correspondiente valor de

retorno se realizaba siempre por valor. Con el uso de apuntadores se consiguió que el valor a pasar fuese sólo el de una dirección de memoria y no el contenido completo de cual fuera la variable o estructura a donde apuntaba éste. Sin embargo, el uso de apuntadores resulta más complejo y difícil de depurar. En C++ las referencias constituyen una solución que aporta las ventajas del uso de apuntadores pero con mayor apoyo del compilador para lograr mantener el código correcto. Utilizando referencias a variables complejas se reduce el tiempo de ejecución y el tamaño del crecimiento de la pila.

5. Primer programa en Arduino

Un problema frecuente en el uso de interruptores es el rebote. El rebote de contacto ocurre en interruptores y relés mecánicos. Cuando los puntos de contacto entre las superficies golpean uno contra otro, la elasticidad produce rápidos pequeños rebotes que se repiten hasta que se disipa la energía de los rebotes y se produce un contacto en reposo. Los circuitos digitales suelen ser tan rápidos que pueden malinterpretar varios de estos rebotes como múltiples acciones de parte del usuario de activaciones repetidas del interruptor. Una opción tradicional para solucionar el problema de rebote consiste en utilizar un circuito de filtro con un condensador y una resistencia para producir una curva con una pendiente lo suficientemente lenta de forma que cuando el voltaje cruce el umbral de detección las superficies de contacto ya se encuentren en reposo. Es una opción eficaz, económica y simple. Sin embargo, queriendo poder reducir al mínimo el hardware utilizado para permitir aplicaciones compactas, se hará uso de la resistencia interna de pull up (de subida).

En los circuitos lógicos correctamente diseñados, todas las entradas deben estabilizarse en un valor lógico conocido. Dependiendo de la familia lógica las entradas en determinados valores pueden introducir mayor consumo eléctrico y ruido. Por ejemplo, en la familia CMOS de alta velocidad, la componente de potencia más importante depende de la frecuencia de reloj cuando opera con entradas cambiantes. Una entrada “flotante” que se acople capacitivamente a pistas de energía o de

señales de potencia puede estar introduciendo transiciones en el circuito CMOS que le hagan multiplicar el ruido y el consumo de potencia. El microcontrolador ATMega328 está diseñado (como es común en microcontroladores que operan a 5V) para interconectarse con circuitos de familias lógicas de TTL (del inglés de lógica de transistor a transistor). Una entrada TTL estándar en nivel lógico “1” opera asumiendo una fuente de corriente de 40 uA, y un nivel de voltaje por encima de 2,4V. El valor máximo de una resistencia de pull up dentro del margen de tolerancia de las salidas lógicas para esta situación sería de 50 kOhm, donde el voltaje que alimente a la resistencia no podría bajar de 4,4V. En cambio, una entrada TTL estándar en nivel lógico “0” opera asumiendo una fuente de corriente de 1,6 mA, y un nivel de voltaje por debajo de 0,8V. El valor máximo de una resistencia de pull down (de bajada) dentro del margen de tolerancia de las salidas lógicas para esta situación sería de 500 Ohm, donde el voltaje que alimente a la resistencia tendría que ser obligatoriamente 0V. Se requiere consumir una corriente mucho menor para mantener una entrada de un circuito TTL en niveles bajos. El ATMega328 en su hoja de datos señala un rango de entre 20 kOhm y 50 kOhm. Utilizando el multímetro (téster) entre la entrada con la resistencia de pull up activada y tierra obtenemos una corriente de 0,14 mA, correspondientes a un valor de resistencia de pull up aproximado de 36 kOhm.

La función habilitada por defecto en el Arduino IDE millis() cuenta los milisegundos transcurridos desde que inició por última vez el microcontrolador. Presenta un desbordamiento cada 50 días, así que es poco frecuente que se pueda perder una entrada del usuario al interruptor en ocasión de ocurrir dicho desbordamiento mientras se espera a que terminen los rebotes del interruptor. En caso que ocurriese, el usuario tendría que presionar el interruptor nuevamente, lo que se considera una incomodidad menor e infrecuente.

5.1 Control de led mediante interruptor

Se realizará el primer programa en Arduino que incluye lectura de interruptor sin condensador externo. La ya mencionada resistencia de pull-up no puede evitar cambios repetitivos y aleatorios mientras las

superficies de contacto del interruptor sufren rebotes. Sin embargo, es posible diseñar un programa que espere a que los rebotes se detengan antes de tomar una decisión sobre la activación del interruptor. Un usuario común tiene un tiempo de respuesta de accionamiento de botones superior a 50 milisegundos por lo cual este tiempo suele ser imperceptible visto desde el punto de vista de interactividad del usuario.

El código está desarrollado y comentado en inglés como es estándar en el desarrollo de software en Arduino. La razón de la elección del idioma inglés es facilitar la comunicación entre numerosos colaboradores de distintas nacionalidades. A principios de 2016 ya se han vendido más de 1 millón de módulos de desarrollo para Arduino oficiales sin contar el número de módulos vendidos por fabricantes que no están afiliados a Arduino pero que realizan módulos con las mismas características siguiendo fielmente los diseños abiertos. La mayoría de los módulos oficiales se han vendido en Europa y Estados Unidos, pero un gran número no determinado de módulos no oficiales se han vendido en Asia.

Los nombres de funciones y clases generalmente seguirán la escritura en camello, término que proviene del inglés “*CamelCase*”, con pequeñas excepciones relativas a resaltar alguna palabra clave o al utilizar siglas. Los nombres de las variables generalmente seguirán la misma convención exceptuando que la primera letra estará en minúscula. Los nombres de macros y constantes definidas en el preprocesador estarán en mayúsculas.

5.2 Código Control de led mediante interruptor

```
// El terminal 13 tiene un LED conectado en la mayoría
// de las placas Arduino

int led = 13;
bool ledON = true;
int button = 2;          // Conexión botón o interruptor

unsigned long lastTime = millis();
bool buttonLastRead; // Última lectura botón
bool buttonState;            // Estado reportado botón
bool buttonAck;              // Reconocimiento cambio estado botón
```

```
int debounceDelay = 50;        // 50 milisegundos tiempo estándar
                               // espera anti rebote

// la rutina setup (configurar) se ejecuta una vez después de presionar
// el botón de reset (reinicio)
void setup() {
        // inicializar el terminal digital como salida
    pinMode(led, OUTPUT);
        // la resistencia de subida de aprox. 36 kOhm
        // mantiene la entrada
        // en un estado conocido cuando el botón
        //no se encuentra presionado
    pinMode(btn, INPUT_PULLUP);
    buttonLastRead = digitalRead(button);
    digitalWrite(led, ledON);
}
    // la rutina loop (bucle) se ejecuta una y otra vez
void loop() {
    if (buttonLastRead != digitalRead(button) {
        lastTime = millis();
    } else {
        If (millis() – lastTime > debounceDelay) {
            buttonState = buttonLastRead;
            buttonAck = false;
        }
    }
    If (buttonAck == false) {
        if (buttonState == true) {
            ledON = ! ledON;
            digitalWrite(led, ledON);
        }
        buttonAck == true;
    }
}
```

Este programa sencillo, mantiene el registro de los cambios del valor de lectura de la entrada y “cronometra” el tiempo transcurrido desde el

último cambio hasta el momento actual. Sólo cuando transcurre un período superior a debounceDelay sin efectuarse cambios, se realiza la actualización de la variable que indica el estado del botón.

La parte final del programa usa la variable buttonAck (abreviación de *button* + *acknowledgement*, que significa reconocimiento del botón) para controlar cuándo es que la parte “activa” o de salida del programa accede al valor de la variable que refleja el estado del interruptor. Con el procesador en operación a la frecuencia de reloj de 16 MHz, si se cambiase el estado del LED cada vez que se ejecuta la función loop (bucle) mientras esté en true (activada) la variable del estado del botón, el estado del LED alternaría entre encendido y apagado probablemente miles de veces y como consecuencia el tiempo aleatorio de desactivación del botón tendría el efecto de dejar el LED con similar probabilidad en encendido que en apagado. Al poner a falso la variable buttonAck se controla que se cambie el estado del LED entre encendido y apagado una única vez por detección de cambio del estado del botón después de descartar (o “filtrar”) los rebotes.

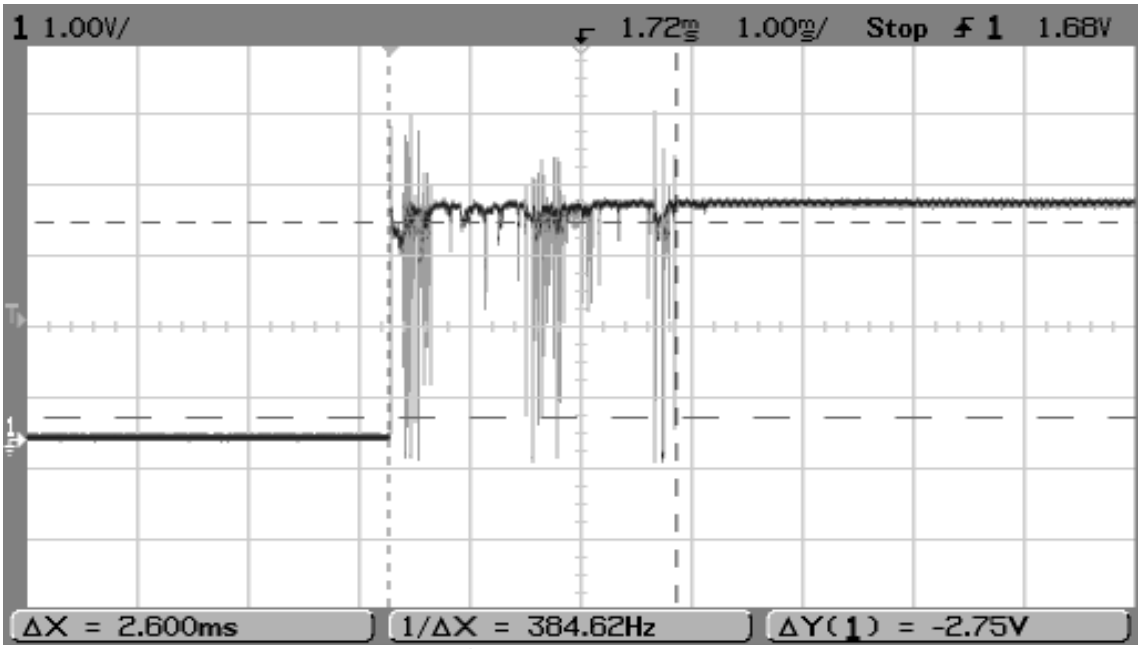


Figura 6. Ejemplo típico de rebote de interruptor

La figura presenta una captura de pantalla de un osciloscopio que representa el voltaje en el terminal que no está conectado a tierra en un circuito típico de un interruptor en serie con una resistencia a la tensión

de operación de un circuito lógico que opera a 3,3V. Se observa dentro de los 3 ms posteriores a su activación, al menos 7 rebotes que llevan el nivel de voltaje debajo del umbral lógico para el 0, “low” (voltaje bajo se corresponde con un valor lógico “0” en los circuitos de lógica positiva), otros muchos rebotes de menor amplitud, donde diversos factores eléctricos del circuito producen cambios en el voltaje que quizá también hayan cruzado el umbral lógico del “0” pero durante un tiempo tan corto que no se puede apreciar en la configuración actual del osciloscopio.

Como todas las técnicas, conviene pensar sobre las ventajas y las desventajas de su aplicación. El código expuesto es una adaptación extendida para facilitar la explicación sobre su funcionamiento de la técnica sugerida en la página de Arduino sobre interruptores. La principal desventaja es que depende totalmente de que se ejecute frecuentemente el código en la función de loop() y la siguiente en importancia es que ocupa 14 bytes en la escasa memoria SRAM. Si la aplicación requiere en un futuro realizar extensos cálculos dentro de la función de loop() para brindar otra funcionalidad o por cualquier razón se demora más de 50 ms entre las sucesivas ejecuciones de la función de lazo el código comienza a perder el registro de todas las activaciones. Sin duda, ningún usuario estaría complacido con un sistema que responde aleatoriamente.

Una alternativa al problema del retraso en las repeticiones de llamada a la función loop consiste en utilizar interrupciones. Las interrupciones se componen de una señal de interrupción y de una rutina de interrupción.

Durante la operación normal del CPU dentro del microcontrolador, mientras está activo se encuentra realizando el ciclo de ejecución de instrucciones:

1) Busca la instrucción a la que apunta el contador del programa.
2) Busca los operandos y/o las banderas de condiciones de la evaluación de resultados que requiere la instrucción.
3) Ejecuta el cálculo asociado a la instrucción.
4) Evalúa las condiciones del resultado.
5) Determina la nueva dirección que contendrá el contador del programa y el nuevo estado de ejecución de la instrucción actual.

6) Guarda el resultado y/o las banderas de condiciones de la evaluación del resultado.

7) Si la instrucción requiere más operaciones regresa al paso 2, si la instrucción actual ha terminado regresa al paso 1.

No todas las instrucciones siguen todos los pasos.

Las interrupciones tienen la propiedad de que su presencia es temporal y puede pasar “desapercibida” por un programa con el cual no comparte información. Un mecanismo común para lograr la no alteración del programa en ejecución consiste en mantener en espera a la interrupción hasta que finaliza la instrucción actual y se ha calculado cuál es la instrucción a donde correspondía continuar en la ejecución del programa de no haber sido activada una interrupción. En este momento no se vuelve al paso 1 sino que se ejecuta una instrucción especial de activación de interrupción, externa al programa que se encontraba en ejecución, que guarda en memoria todo lo relacionado con el estado de ejecución del programa que incluye el contenido de las variables del CPU que son necesarias para atender a la interrupción como la posición de la pila, se busca en una tabla de direcciones llamada tabla de vectores de interrupción, la fila correspondiente al generador de la interrupción actual, se llama al procedimiento, función o método que esté guardado en dicha fila y cuando termine de realizar su tarea, llama a una instrucción especial de regreso de interrupción que recupera todo lo relativo al estado de ejecución del programa que fue guardado en la fase de activación.

Un manejador de interrupción es un procedimiento, método o función definido con cualquier nombre definido para realizar alguna o algunas de las diferentes tareas que requiere el programa en relación a un generador de interrupciones para cumplir con sus funciones. Los manejadores de interrupción en Arduino no devuelven ni reciben valores. Como todo procedimiento, método o función queda definido exclusivamente por su dirección de memoria asignada después del proceso de compilación y enlace. Esta dirección es la que se utiliza en todas las llamadas que se realizan dentro del programa para ejecutar dicho procedimiento, método o función. El programa llama a la función

attachInterrupt(digitalPinToInterrupt(pinNumber) interruptHandler, condition) para guardar la dirección del manejador de interrupción “interruptHandler” en la dirección de memoria del “vector de interrupción” que indica la función “digitalPinToInterrupt()” y para guardar los valores de “condition” en la dirección o direcciones de memoria de registros de control de la interrupción. Los vectores de interrupción son posiciones de memoria preasignadas en hardware para colocar las direcciones de los manejadores de interrupción. Se establece durante el diseño del hardware del CPU qué tipo de interrupción y qué interrupción va en cada posición de memoria de la tabla de vectores de interrupción. Cada posición en la tabla de vectores de interrupción está indicada en cada conexión de hardware generador de interrupciones correspondiente. En Arduino UNO R3, el terminal 2 del ATMega328 está cableado al vector de interrupción 0 (INT0), con lo que sabemos que si se encuentra habilitada la interrupción 0 (INT0) y se aplica una señal de nivel lógico 1 al terminal 2, el hardware de interrupciones activará la instrucción especial de llamada de interrupciones ya explicada justo después de que se termine de ejecutar la instrucción actual, guardando todas las variables del CPU relacionadas con el estado de ejecución del programa, realizará la llamada al vector de interrupción interruptHandler porque su dirección estará guardada en la posición (INT0) y después que termine la ejecución del interruptHandler este llamará a que se ejecute la instrucción especial de restauración del estado de ejecución del programa.

En las tablas de manejadores de interrupciones la dirección que corresponde con el tipo de interrupción que ha ocurrido (suele haber más de 1 dirección de manejador de interrupción aunque puede repetirse la misma).

5.3 Diseño de un interruptor para IoT

Siguiendo la filosofía del “*Internet of Things*”, las “cosas” deben estar en capacidad de ofrecer una interfaz “natural” que les permita ser operadas como si se tratase de las antiguas cosas sin internet ni avanzado poder computación a las que reemplazan.

El usuario debería poder utilizar un interruptor integrado a un dispositivo IoT como se encuentra acostumbrado a usar los interruptores tradicionales. Y como se vio en el caso de uso sobre la familia en la casa “del futuro” los usuarios son especialmente exigentes en este tema cuando se encuentran arrastrando los pies para avanzar en total oscuridad.

El interruptor ideal que se puede diseñar tomando en consideración las conclusiones del caso de uso mencionado y las interpretaciones expuestas en el marco teórico consistiría en un interruptor deslizante con sus posiciones convencionales superior para encendido e inferior para apagado que se pueda manipular físicamente. Las funciones añadidas no deberían interferir con esta interfaz. Por tanto, la solución ideal aunque inexistente en el mercado actual, es un interruptor deslizante con bobina que pueda accionarse electromecánicamente desde cualquier interfaz nueva, a modo de su activación o desactivación por medios electrónicos no rompa la correspondencia entre la posición mecánica y el estado de funcionamiento. Entrando en los detalles de funcionalidad donde varios usuarios activaban algunos dispositivos accidentalmente o pensando que nadie más los estaba utilizando, pudiera utilizarse un color distinto para cada usuario registrado. El interruptor reflejaría con el color el emisor de la orden que permanece activa y pudiese discernir entre cada usuario que accede físicamente a activarlo midiendo la capacitancia de contacto de cada uno. Puede haber reglas en el sistema de control del interruptor como no permitir que determinado grupo de usuarios cambie el estado del interruptor dentro de un tiempo dado desde que ha sido activado o desactivado por otro usuario de forma manual o electrónica.

Las funcionalidades añadidas como las relacionadas a sensores de niveles de luz pueden resultar muy convenientes pero en determinados casos pueden resultar un obstáculo. Un ejemplo de uso incluye la situación en que un usuario se dispone a realizar una tarea que requiere una excepcional atención al detalle, como lo puede ser enhebrar una aguja disponiendo de un nivel medio de luz natural. El sistema de control de iluminación inteligente puede encontrarse encendido y detectar al

usuario pero el nivel de luz natural puede estar indicando que no es necesario activar las luces. En este caso convendría que el sistema alterne entre modo “manual” e “inteligente”. Si el sistema de capacitancia de contacto fuese incorporado para la identificación del usuario, pudiese también utilizarse el tiempo de contacto para señalar el cambio de modo. Por ejemplo tocar el interruptor durante más de 3 segundos puede ser una señal conveniente para cambiar el modo de funcionamiento y poder realizar las tareas que requieren mucha iluminación, así como disfrutar de una siesta.

6. Sistema de control de iluminación inteligente

En esta etapa de desarrollo iterativo se busca implementar la funcionalidad básica del sistema de control de iluminación inteligente. Debido al uso del emulador *Autodesk Circuits* se utilizará un módulo compatible con el modelo de módulo integrado PIR (“Passive Infrarred” del inglés para sensor de detección de infrarrojos pasivo)

Figura 7. Sensor PIR Passive Infrarred

y un sensor de luminosidad de sulfuro de cadmio identificado en el emulador como LDR (“Light-Dependent Resistor”), en el prototipo final se utilizará el siguiente modelo con potenciómetro incluido, LDR Shield.

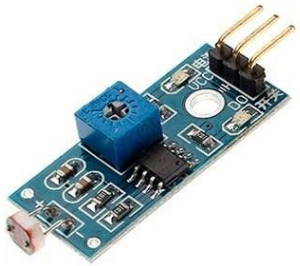


Figura 8. Sensor LDR

Los sensores PIR son de tipo pasivo porque no emiten ninguna radiación hacia el objeto a detectar. Están constituidos generalmente por los siguientes 3 componentes:

1) Una cubierta de plástico blanco que en realidad es transparente al infrarrojo sirviéndole de protección física, filtro de luz y hasta de arreglo de lentes multifacetados de tipo fresnel.

2) Un sensor piro eléctrico especialmente sensible a la luz infrarroja y que suele estar dividido en 4 secciones.

3) Y la electrónica que adecúa la señal del sensor y compara con valores pre-calibrados para realizar la detección.

La principal desventaja de este sensor es que se trata de un sensor de movimiento, por lo que puede causar inconvenientes cuando la persona realice actividades muy estáticas. Reportando erróneamente su ausencia.

Para un control domótico del hogar se requiere una cobertura completa utilizando un gran número de estos sensores. Conviene reemplazarlos por sensores mejor configurados para también detectar situaciones estáticas y si no hay posibilidad de hacerlo, en proyectos futuros desarrollar un módulo diseñado para dicha tarea.

El sensor LDR ha resultado ser un dispositivo muy fácil de usar sin mayor electrónica añadida que la resistencia de pull up y el conversor analógico a digital interno del ATMega328.

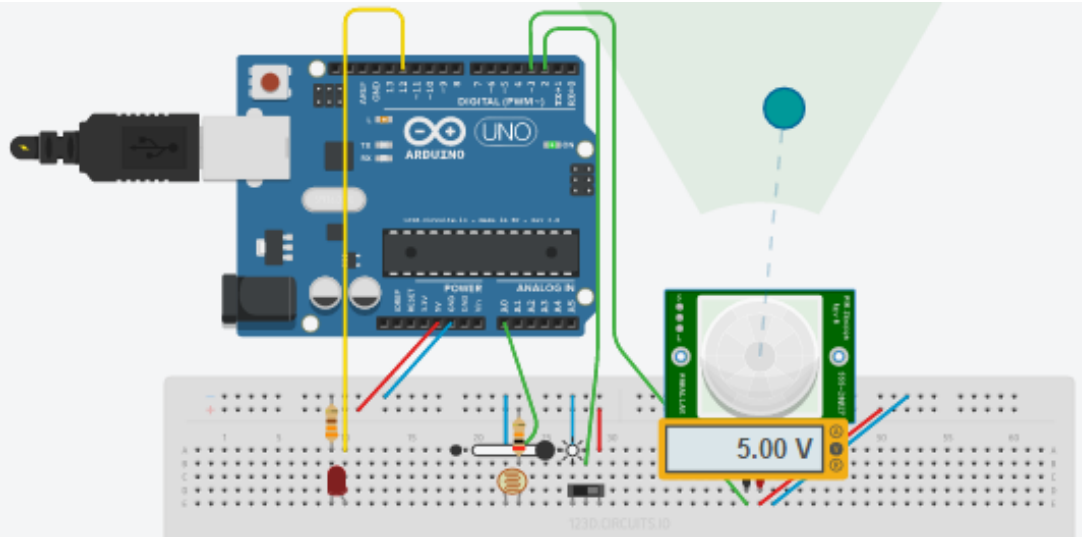


Figura 9. Emulación de sistema de control

Este circuito implementa toda la lógica necesaria sobre los sensores relacionados con la administración eficiente de la energía. Se ha utilizado el módulo PIR estándar disponible en la biblioteca de componentes del emulador y se corresponde bien con el módulo PIR de Keyestudio ks0052. Este módulo encapsula toda la lógica y el hardware de conversión analógica a digital de sensor de iluminación infrarroja, es compatible con TTL trabajando en un rango de voltajes entre 3,3V y 6V.

El componente PIR estándar disponible en la biblioteca de componentes provee además de herramientas visuales para emular el campo de visión del dispositivo y la emulación de un objeto “cálido en infrarrojo” para desplazar con el ratón y observar la activación de la alarma del módulo. La alarma del módulo tiene un tiempo de persistencia propio durante el cual se mantiene activa a pesar de retirar el objeto emulado del campo de visión del sensor.

Se ha decidido programar un tiempo de retención en software independiente del tiempo de retención del módulo ks0052.

El LDR también se encuentra en la biblioteca de componentes y al activarlo con el ratón mientras está andando la “simulación” (ya se ha explicado que se trata en realidad de una emulación) aparece un control de barra deslizante con un punto de selección y a los extremos de la barra un círculo obscuro y un círculo blanco radiante para escoger entre distintos niveles de luminosidad.

El interruptor de encendido y apagado del sistema de iluminación inteligente utiliza una barra deslizante sin ningún tipo de condensadores. No posee retroalimentación. La forma en que se ha programado el Arduino y tipo de interruptor hacen que sea innecesario un control específico para eliminar rebotes.

El siguiente circuito a considerar es el de regulación de potencia para el sistema de luces. Para este caso de uso se ha conseguido unos relés de estado sólido al mismo precio que los correspondientes relés electromecánicos. Si bien tienen menor capacidad de corriente, ésta es más que suficiente para los requisitos del sistema de iluminación. A diferencia de los electromecánicos, tienen una durabilidad y velocidad de respuesta mucho mayores.

La regulación de la potencia se obtiene mediante el cambio del ciclo de trabajo de una forma cuadrada que enciende y apaga la forma de onda sinusoidal de corriente alterna común para el consumo eléctrico doméstico. El circuito se mantiene sincronizado con la fase para mantener valores consistentes a lo largo de la operación de regulación de potencia. En el circuito emulado se ha colocado un transistor npn TIP120 y un relé electromecánico en sustitución de los relés de estado sólido utilizados.

Los relés de estado sólido utilizados son los Omron de cruce por cero. La función de cruce por cero produce que el relé se encienda cuando la fuente de energía de la carga AC se acerca a 0V para suprimir el ruido generado cuando la corriente de carga sube repentinamente.

Hay 2 tipos de ruido:

- ruido sobre las líneas de tensión
- ruido emitido hacia espacios abiertos

La función de cruce por cero es efectiva contra ambos tipos de ruido.

Las lámparas y equipos similares producen una corriente de entrada muy grande cuando se encienden.

El hecho de que el dispositivo opere sólo en cruces por cero implica también una limitación en cuando a su funcionalidad. A 50Hz 220VA se produce un cruce por 0 sólo cada 10ms. El circuito de cruce por cero no es perfecto y puede haber un margen de -20V a 20V. Por lo expuesto en este párrafo el control de intensidad de luz para bombillas incandescentes se verá reducido a sólo 4 niveles: 100%, 50%, 33% y 25%, ya que intentar controlar con más niveles puede terminar en incómoda sensación de luz destellante.

Para el seguimiento de la fase de los cruces por 0 se ha utilizado un par de optoacopladores 4N35. El 4N35 consiste en un fotoled entre los terminales 1 y 2 que soporta hasta 60mA y un fototransistor NPN con terminales: 4 emisor, 5 base y 6 colector. El objetivo principal del optoacoplador es aislar eléctricamente 2 circuitos, especialmente para reducir riesgos de cortocircuitos. Se emulará el circuito de potencia con un generador de funciones sinusoidal a 50Hz y 31,11Vp de amplitud, una décima parte del voltaje correspondiente a 220Vrms o 311,1Vp. Las

únicas conexiones del sistema de control de iluminación inteligente a las líneas de 22Vrms de la fuente de alimentación son a través de 2 resistencias en serie de 100 kOhm (en el circuito real representarán resistencias de 1 MOhm, limitando la corriente en el fotodiodo 4N35 a 155,5 uA) y el circuito de potencia del relevador.

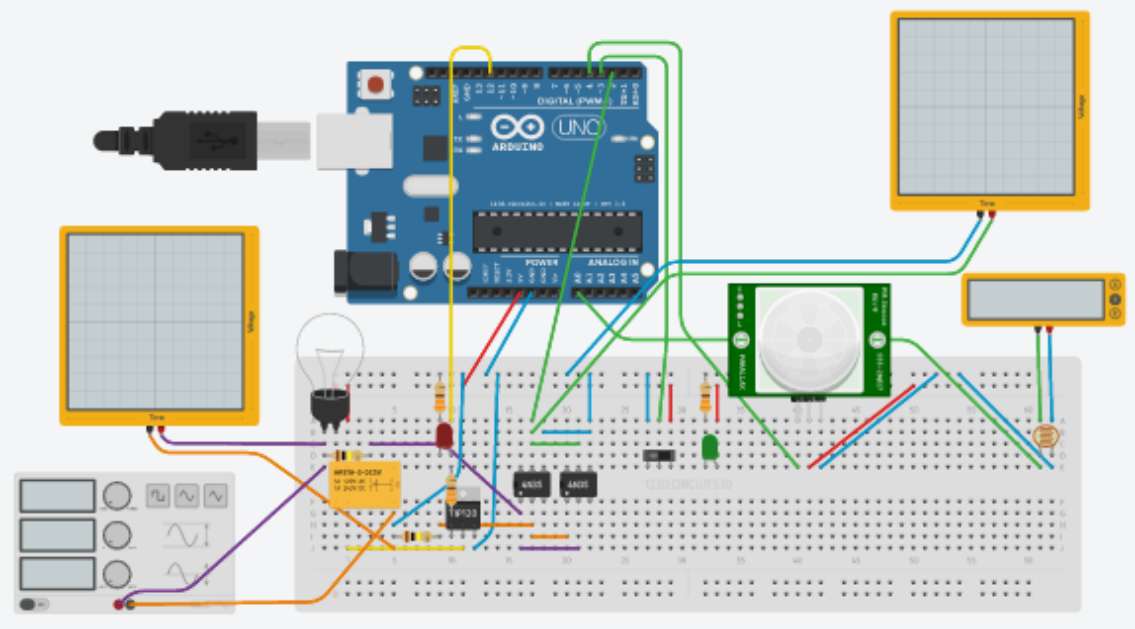


Figura 10. Emulación de sistema de control de iluminación inteligente con etapa de potencia en modo inactivo.

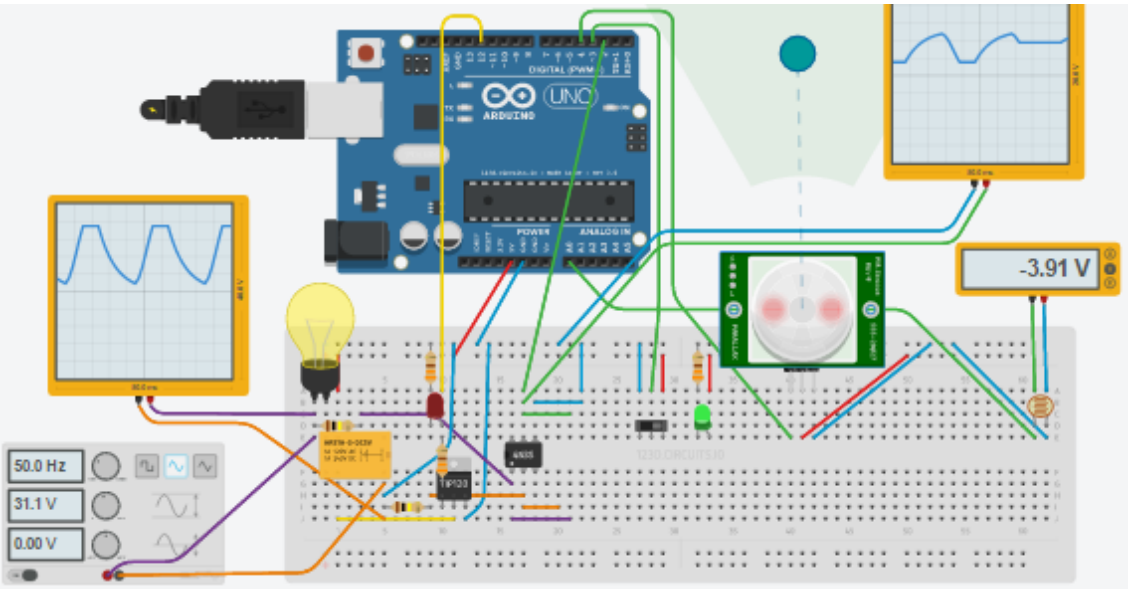


Figura 11. Emulación de sistema de control de iluminación inteligente con etapa de potencia en modo activo.

Este es el código fuente que implementa el control del sistema emulado:

```
int led = 13;
int alarmSignal = 12;
bool alarmON = false;
const int turnOffDelay = 1000;
int timeTurnOff;
int crossingSineSignal = 2;
bool crossingSineActive = false;
int crossingSineThreshold = 20;
int ldr = A0;
int ldrVal;
int ldrThreshold = 50;
int btn = 3;
int btnVal;
int pir = 4;
bool pirVal = false;
int intensityDivider = 4;
int intensityCounter = 0;

void setup() {
        // Uso del Monitor Serial para calibración y reporte
    Serial.begin(9600);
    pinMode(led, OUTPUT);
    pinMode(alarmSignal, OUTPUT);
// resistencia interna de carga para fototransistor optoacoplado
    pinMode(crossingSineSignal, INPUT_PULLUP);
    pinMode(btn, INPUT_PULLUP);
        // sensor pull up
    pinMode(ldr, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(crossingSineSignal), crossingSine,
RISING);
}
void loop() {
    btnVal = digitalRead(btn);
    digitalWrite(led, btnVal);
    pirVal = digitalRead(pir);
    ldrVal = analogRead(A0);
    if(alarmON) {
```

```
            if(millis() > timeTurnOff || ldrVal <= ldrThreshold) {
                alarmON = false;
            }
        } else {
            digitalWrite(alarmSignal, false);
        }
        If (btnVal && ldrVal > ldrThreshold && pirVal) {
            timeTurnOff = millis() + turnOffDelay;
            alarmON = true;
        }
    }
    void crossingSine() {
        Serial.println(millis(),DEC);
        If (alarmON) {
            intensityCounter++;
            if(intensityCounter >= intensityDivider) {
                intensityCounter = 0;
                digitalWrite(alarmSignal, true);
            }else {
                digitalWrite(alarmSignal, false);
            }
        }
    }
```

En la emulación del sistema de control de iluminación inteligente con etapa de potencia en modo activo, se ha utilizado sólo 1 de los optoacopladores para facilitar la emulación. Se ha utilizado el puerto Serial de Arduino IDE que es un puerto Serial virtual simulado a través de una conexión USB y el controlador 16U2 en la placa Arduino que conecta a los terminales 0 y 1 del ATMega328.

Se observa el reporte de los tiempos de interrupción, de aproximadamente cada 20ms, en concordancia con utilizar sólo 1 de los optoacopladores, con lo cual se detectan las subidas sólo en la parte positiva de la función seno de la forma de onda de la entrada que representa la alimentación AC de la etapa de potencia.

Debido a la importancia de la sincronización del tiempo en lo referido anteriormente sobre el relé de estado sólido de cruce por cero, se ha

utilizado la característica de interrupción para poder actuar sobre esta señal con la máxima velocidad posible y generar una respuesta adecuada a tiempo.

La variable intensityCounter se utiliza para llevar una cuenta de los ciclos (semiciclos si se utilizan los 2 4N35) que ha estado desactivado el relé. La variable intensityDivider es la que determina qué fracción de entre todos los ciclos corresponde al relé en estado encendido.

7. Comunicación Arduino-Android

Una vez implementado y en funcionamiento del prototipo diseñado hasta este punto, se dispondrá a establecer una comunicación entre la plataforma Arduino y el software de control, que podrá ser desde un sistema operativo como Windows o desde dispositivos móviles basados en Android. Como no es el objetivo de este trabajo diseñar una aplicación propia para Android, se utilizarán herramientas gratuitas para control y gestión de estas comunicaciones.

7.1 Comunicación serie Arduino

Un puerto es el nombre genérico con que se denomina a los interfaces, físicos o virtuales, que permiten la comunicación entre dos ordenadores o dispositivos. La comunicación por el puerto serie es muy importante porque gran parte de los protocolos utilizados actualmente son serie y además muchos dispositivos de comunicación inalámbrica usan la comunicación serie para hablar con Arduino como los módulos bluetooth y los módulos Xbee. Esta comunicación serie es la que se usa generalmente para comunicar el Arduino con el Ordenador.

Para manejar el puerto serie en Arduino, se debe leer a fondo la referencia de Arduino:

http://arduino.cc/en/Reference/Serial

Todas las placas Arduino tienen al menos un puerto serie disponible en los pines digitales 0 (RX) y 1 (TX) compartido con el USB. Por lo tanto no es posible usar estos pines como entradas/salidas digitales.

El Arduino mega dispone de tres puertos adicionales Serial1 on pins 19 (RX) and 18 (TX), Serial2 on pins 17 (RX) and 16 (TX), Serial3 on pins

15 (RX) and 14 (TX). Estos pines no están conectados al interfaz USB del Arduino.

Prácticamente todas las placas Arduino disponen al menos de una unidad UART. Las placas Arduino UNO y Mini Pro disponen de una unidad UART que operan a nivel TTL 0V / 5V, por lo que son directamente compatibles con la conexión USB, que será la que se utilice en este trabajo.

El código de la librería Serial se encuentra en el core del IDE de Arduino y las funciones más importantes que se deben conocer para manejar el puerto serie son: begin(), read(), write(), print() y available())

- **begin()** – establece la velocidad de la UART en baudios para la transmisión serie, también es posible configurar el número de bits de datos, la paridad y los bits de stop, por defecto es 8 bits de datos, sin paridad y un bit de stop.
  https://www.arduino.cc/en/Serial/Begin
- **read()** – lee el primer byte entrante del buffer serie.
  https://www.arduino.cc/en/Serial/Read
- **write()** – escribe datos en binario sobre el puerto serie. El dato es enviado como un byte o serie de bytes.
- **print()** – imprime datos al puerto serie como texto ASCII, también permite imprimir en otros formatos.
  https://www.arduino.cc/en/Serial/Print
- **available()** – da el número de bytes (caracteres) disponibles para leer en el puerto serie, son datos que han llegado y se almacenan en el buffer serie que tiene un tamaño de 64 bytes.
  https://www.arduino.cc/en/Serial/Available

Los puertos serie de los microcontroladores tienen un buffer que se va llenando hasta que se van leyendo con la función read() hasta vaciarse, es una pila FIFO. El tamaño del buffer serie en el Arduino Uno es de 64 bytes, cuando se llena ese buffer el resto de elementos recibidos se pierden.

El monitor de puerto serie es una pequeña utilidad integrada dentro de IDE Standard que nos permite enviar y recibir fácilmente información a

través del puerto serie. Su uso es muy sencillo, y dispone de dos zonas, una que muestra los datos recibidos, y otra para enviarlos. Estas zonas se muestran en la siguiente imagen.

Figura 12. Interfaz serie IDE Arduino

Para un primer paso en la comunicación necesaria se añadirá el siguiente código:

```
if (Serial.available()>0){
        if(Serial.read()=='1')
          {
            digitalWrite(2, HIGH);
          }
        else
          {
            digitalWrite(2, LOW);
          }
}
```

Al recibir un 1 por el puerto serie se activará la INT0, interrupción 0, pin digital 2, para cualquier otro valor no se activará.

7.1 EventGhost

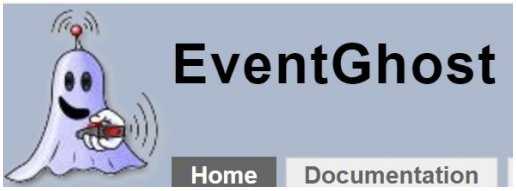


Figura 13. Logo EventGhost

EventGhost, software gratuito y de código abierto, es una herramienta avanzada, de automatización fácil de usar y extensible para Windows, y gratuita. Puede utilizar diferentes dispositivos de entrada como los mandos a distancia por infrarrojos o inalámbrica para activar macros, que por su parte toman el control de un ordenador y su hardware asociado. Una macro es un conjunto de sentencias a realizar. Este software se utilizará para controlar la placa Arduino y disponer de una comunicación serie.

Web EventGhost:

http://www.eventghost.org/

Manual EventGhost:

http://www.eventghost.org/mediawiki/index.php?title=Short_Manual

Este programa se configura para crear una serie de tareas y poder controlar la iluminación al recibir un comando a través del teclado, por voz o por un mensaje enviado desde un dispositivo móvil con Android.

Para el objetivo de este trabajo se detallan la acciones realizadas en este software de una forma abreviada y sin entrar en muchos detalles, ya que cada software que se va a utilizar para la comunicación es amplio en sus posibilidades, configuración y manejo:

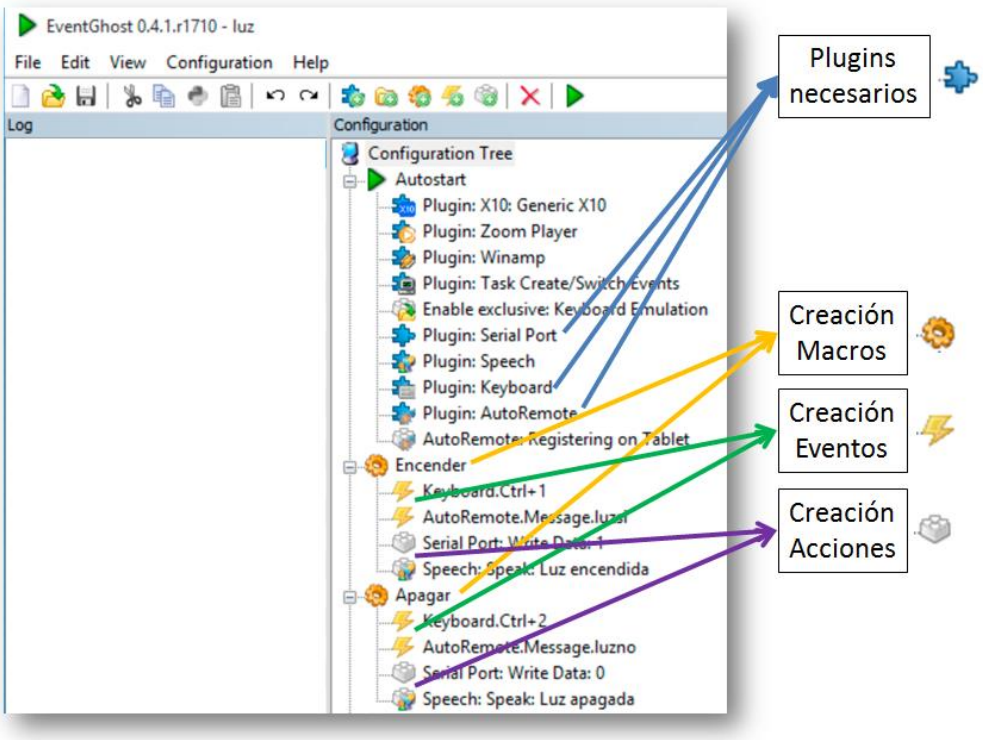


Figura 14. Interficie EventGhost

Añadir plugins:

Se requieren los siguientes plugins en EventGhost

- Serial Port, para la comunicación con la placa Arduino a través del puerto serie adecuado (COM6) y poder enviar las acciones a realizar para el control de la iluminación.

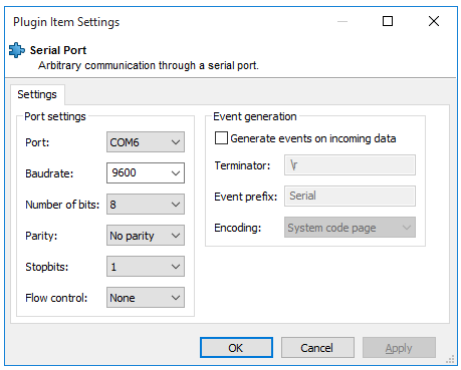


Figura 15. EventGhost, plugin Serial Port

- Speech, este plugin es un valor añadido para que se oiga una voz a través del ordenador de un texto introducido.

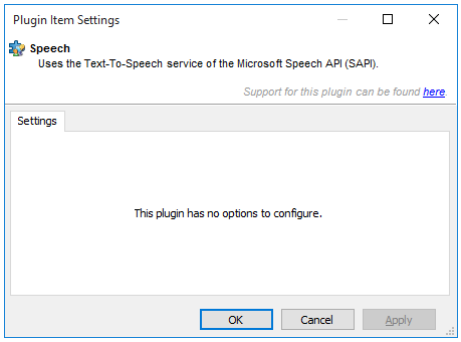


Figura 16. EventGhost, plugin Speech

- Keyboard, a través de un evento de teclado se podrá mandar órdenes para realizar tareas.

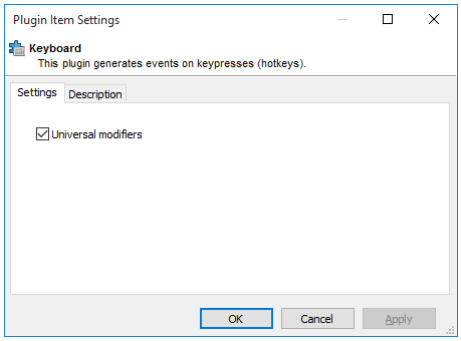


Figura 17. EventGhost, plugin Keyboard

- AutoRemote, el plugin necesario para la comunicación vía internet entre el dispositivo móvil y el ordenador.

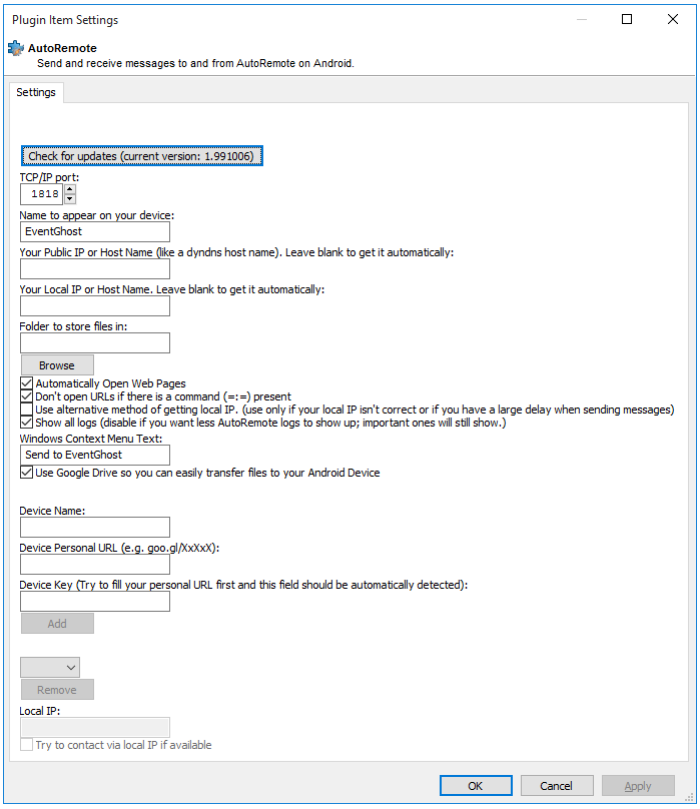


Figura 18. EventGhost, plugin AutoRemote

Creación macro Encender/Apagar:

Las 4 sentencias que contiene la macro Encender son: enviar un 1 a través del puerto serial, una locución diciendo “luz encendida”, la activación a través del teclado con las tecla Ctrl+1 o al recibir un mensaje “luzsi” desde la app móvil Autoremote. La macro Apagar es una copia configurando otros parámetros como son: enviar un 0 a través del puerto serial, una locución diciendo “luz apagada”, la activación a través del teclado con las tecla Ctrl+2 o al recibir un mensaje “luzno” desde la app móvil Autoremote.

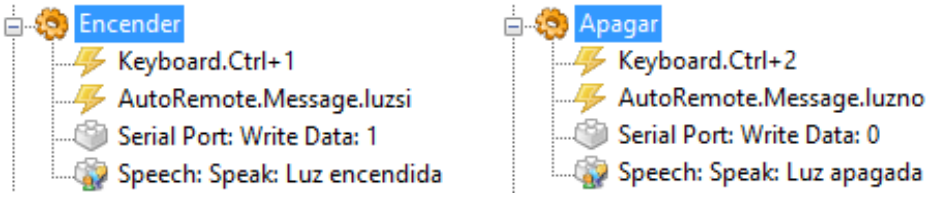


Figura 19. EventGhost, macros Encender / Apagar

7.3 Autoremote lite

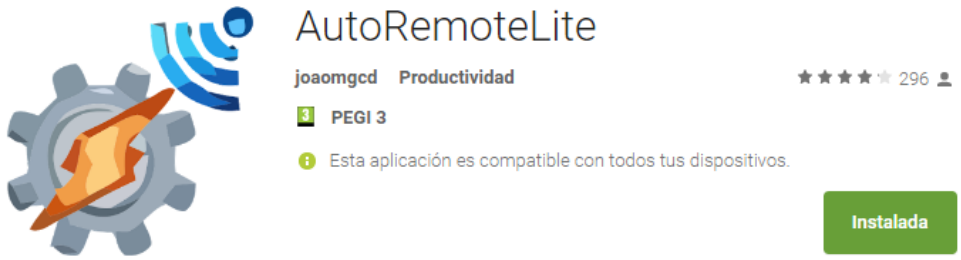


Figura 20. Autoremote lite en GooglePlay

La aplicación Autoremote es un sistema de control remoto muy flexible, que le da al usuario un nivel de control total. Por ejemplo, se puede controlar un ordenador conectado a internet desde un dispositivo con Android o controlar un dispositivo Android con diferentes servicios web (como Facebook o Twitter).

Con esta app se puede gobernar un dispositivo a distancia para decidir el qué funciona y el cómo funciona. Una herramienta muy potente que se utilizará para comunicar un dispositivo móvil con el ordenador al que está conectado la placa Arduino del proyecto.

Las limitaciones de la versión Lite es el envío como máximo en la comunicación de 2 caracteres, que para la comunicación a realizar es más que suficiente.

PlayStore:

https://play.google.com/store/apps/details?id=com.joaomgcd.autoremote.lite

Web:

https://joaoapps.com/autoremote/

Se utiliza esta aplicación para conectar el dispositivo Android con el sistema Windows del ordenador que controlará las acciones creadas en EventGhost. Los pasos a seguir son:

- Instalar app
- Registrarse
- Obtener un código único de conectividad
- Configurar el plugin Autoremote en EventGhost con los datos obtenidos. Para este paso es necesario conocer datos de nuestro equipo que actuará como servidor (IP, puerto, etc.)

Una vez realizados estos pasos, ya existe comunicación entre móvil y ordenador, están conectados.

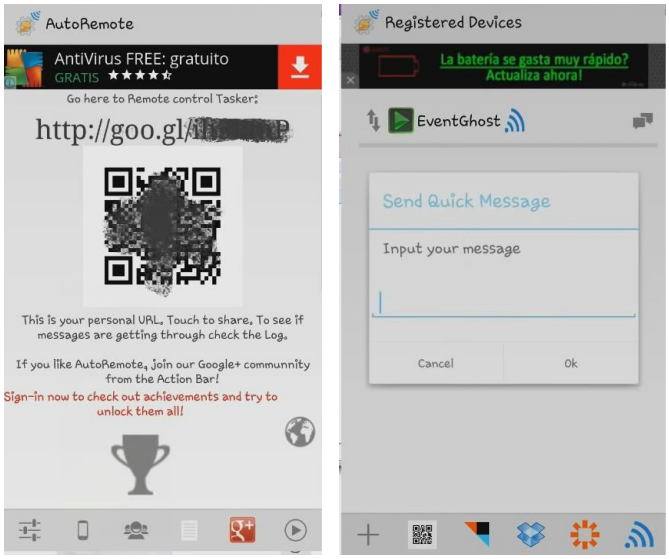


Figura 21 Comunicación Autoremote – Ordenador con EventGhost

7.4 Tasker

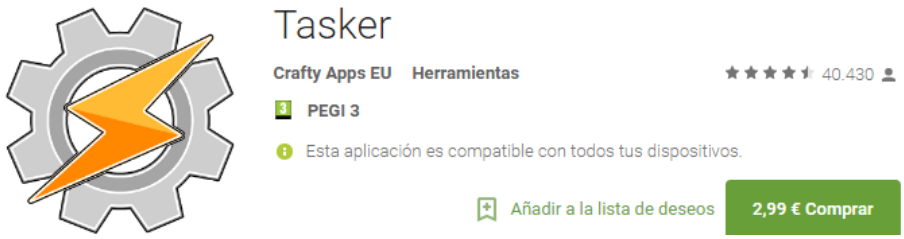


Figura 22. Tasker en GooglePlay

Tasker es una aplicación de pago con versión de prueba para Android que realiza tareas (conjuntos de acciones) basadas en contextos ya sean de aplicación, hora, fecha, ubicación, evento o gesto, en perfiles definidos por el usuario o en widgets de pantalla de inicio con clic o temporizador.

Este concepto simple extiende profundamente el control sobre un dispositivo Android y sus capacidades, sin necesidad de "root" o de costosas aplicaciones creadas a medida.

PlayStore:

https://play.google.com/store/apps/details?id=net.dinglisch.android.taskerm

Web:

https://tasker.dinglisch.net/

Los pasos a seguir para la configuración de unos botones de control de la iluminación en el dispositivo móvil son los siguientes:

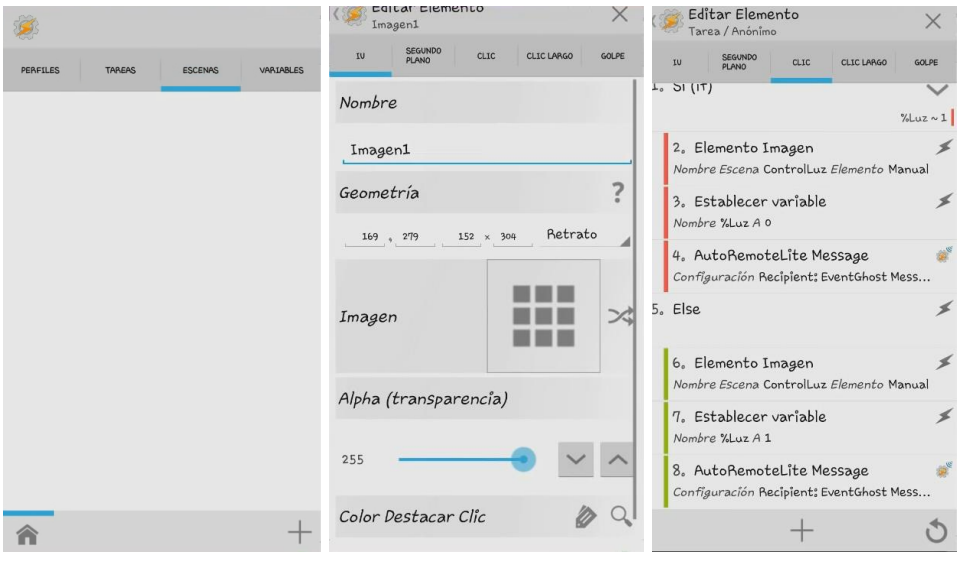


Figura 23. Configuración Tasker

Se crea inicialmente una escena para insertar los diferentes elementos de control, cada elemento es una imagen diseñada exclusivamente para este proyecto.

Figura 24. Iconos botones Android

Como se ve en la anterior figura 23, cada imagen de la figura 24 será un botón asociado a una programación personalizada, en cada caso se enviará un mensaje diferente, que a través de Autoremote llegará al programa EventGhost del ordenador para realizar una acción, dicha acción estará incluida en la macro correspondiente.

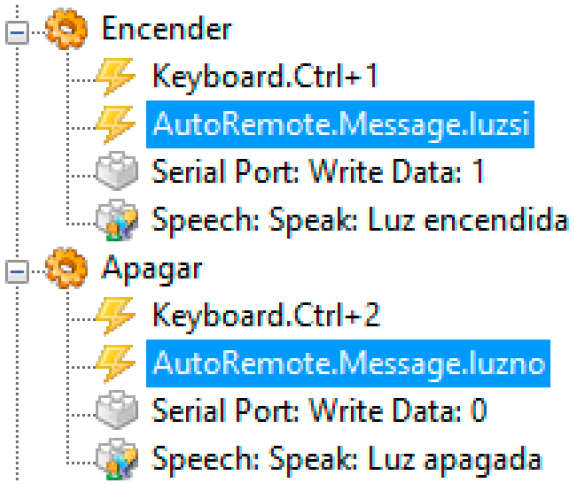


Figura 25. EventGhost recibe mensaje de Tasker vía AutoRemote

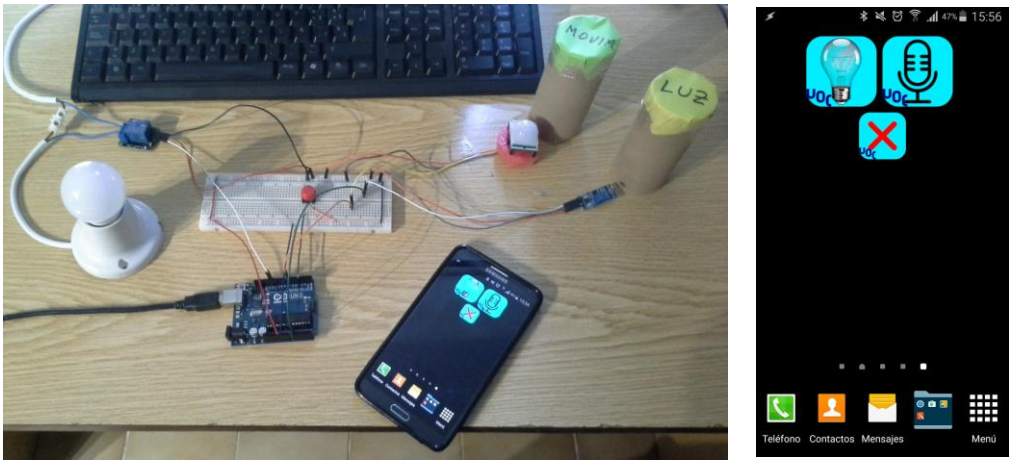


Figura 26. Imagen del prototipo+ iconos Android

A continuación se relacionan unos enlaces privados con el mismo contenido para la visualización del funcionamiento del prototipo, en descarga o visionado online:

Google Drive:

Archivo video en formato MP4
https://drive.google.com/file/d/0B8vzTRcq1VL0MXFWcF84U2M3WG8/view?usp=sharing

Archivo comprimido que contiene video en formato MP4
https://drive.google.com/file/d/0B8vzTRcq1VL0UkZENHB0QkJPeUE/view?usp=sharing

Servidor propio:

Archivo video en formato MP4
http://www.errenet.com/ramadorr_prototipo-en-funcionamiento.mp4

Archivo comprimido que contiene video en formato MP4
http://www.errenet.com/ramadorr_prototipo-en-funcionamiento.zip

8. Valoración económica del trabajo

En este punto se han determinado los costes del material adquirido para el desarrollo del prototipo, mostrados en la tabla a continuación.

Cantidad	Descripción	Precio total
1	Placa Arduino UNO	13,95 €
1	Protoboard	2,07 €
1	PIR	2,17 €
1	LDR	2,00 €
15	Cables jumper	2,00 €
1	Relé	1,87 €
1	Interruptor	0,50 €
1	Portalámpara	0,50 €
1	Bombilla bajo consumo	3,75 €
1	Cable AC	0,30 €
1	Cable USB	1,79 €
1	App Tasker	2,99 €
	Base imponible	33,89 €
	IVA 21 %	7,12 €
	Total costes	**41,01 €**

En cuanto al software ha sido solo el coste de la app, y dado las herramientas utilizadas su coste de mantenimiento es mínimo, ya que se puede considerar si se quisiera un tiempo para algún tipo de recambio de alguna pieza por un tiempo estimado de 5 años. Tampoco se ha tenido en cuenta el control central que repercute en un ordenador, ya que se considera que en cualquier hogar ya existe uno.

Como se puede observar los costos de adquisición son muy bajos y el coste de mantenimiento prácticamente nulo, permitiendo que esta opción sea totalmente factible económicamente, cabe recalcar que no se tiene en cuenta el coste de mimetización de la electrónica con el lugar a implementar, que podría suponer un gasto extra de acondicionamiento.

Se ha discutido en el marco teórico que si se ha de desarrollar un producto comercial con estas tecnologías los costes de hardware pueden ser muchísimo más bajos (menores a 20 €), utilizando versiones de los circuitos integrados mucho más económicos y con placas de circuito impreso que tengan solo los componentes mínimos necesarios para las aplicaciones a las que vayan destinados y adquiriendo los productos en volúmenes de producción y no en calidad de prototipado.

9. Análisis de viabilidad

Se ha ido indicando durante todo el desarrollo del proyecto, que se puede considerar que hay grandes posibilidades de integración de las tecnologías utilizadas en el proyecto, con posibilidades de traducirse en ahorro energético mayores que a los costes de implementación, y en un retorno sobre la inversión.

Desde el punto de vista tecnológico, cada vez más hogares cuentan con tecnologías como PLC, fibra óptica o redes inalámbricas, todavía queda mucho por desarrollar siendo un punto clave en la flexibilidad de la integración de sistemas heterogéneos del “Internet de las cosas”, sus posibilidades futuras de desarrollo dependen de su adopción por parte de los grandes fabricantes.

En contexto general, se observan grandes posibilidades de desarrollo a muy bajo coste.

8. Conclusiones

Todo proyecto siempre brinda una buena gama de conocimientos tanto de base histórica, actual y práctica que incrementan las capacidades y habilidades de la persona.

Puntualizando algunos aspectos generales, se considera:

- Las ventajas de la metodología iterativa respecto a la metodología tradicional en cascada.
- Las posibilidades de mejoras en el hogar que brinda la integración de las nuevas tecnologías.
- Las ventajas de los desarrollos en código abierto que mejoran a través de la colaboración internacional por medio de internet.
- La importancia de la usabilidad en el diseño de un producto.

Se han logrado los objetivos planteados manteniendo al mismo tiempo grandes restricciones sobre el costo del proyecto, así como los primeros ordenadores personales se valieron de la tecnología de la televisión para reducir costos, la integración con los diferentes dispositivos y sistemas de comunicación presentes en los hogares permite reducir enormemente los costos y disponer de grandes capacidades de procesamiento y almacenamiento en donde sea requerida. Gracias a la continua reducción del tamaño de los circuitos integrados se encuentra un número cada vez mayor de dispositivos interconectados e inteligentes permitiendo que los futuros proyectos relacionados con las tecnologías del presente trabajo puedan alcanzar nuevas posibilidades.

La metodología iterativa ha sido la adecuada para el desarrollo del trabajo ya que nos permite rápidamente comprobar la funcionabilidad de cada parte que se va desarrollando, así como, medir sus posibilidades y limitaciones y adaptar lo que sea necesario para poder cumplir con los objetivos del proyecto desde etapas tempranas.

Queda pendiente desarrollar dispositivos específicos que permitan interfaces naturales mejores que las encontradas normalmente en los actuales dispositivos que incorporan el “Internet de las cosas”.

9. Glosario

IDE: Entorno de Desarrollo Integrado

IoT: internet de las cosas, Internet of Things en inglés

ROM: memoria de sólo lectura, en inglés de read-only memory

ENIAC: Computador e Integrador Numérico Electrónico, Electronic Numerical Integrator And Computer

NASA: National Aeronautics and Space Administration

CPU: Unidad central de procesamiento

PLC: control lógico programable

PLC: Comunicaciones a través de las líneas eléctricas, Power Line Communications

CNC: control numérico por ordenador

CAD: diseño asistido por ordenador

CAM: fabricación asistida por ordenador

ICE: In-Circuit Debugger – depurador en circuito

FPGA: Field Programmable Gate Array – arreglo de puertas de campos programables

SCADA: Supervisory Control And Data Adquisition

DIY: hazlo tú mismo, en inglés Do It Yourself

PDA: asistente digital personal, del inglés personal digital assistant

ICSP: In-Circuit Serial Programming

EEPROM: ROM programable y borrable eléctricamente

SRAM: memoria estática de acceso aleatorio

TTL: lógica de transistor a transistor

LED: diodo emisor de luz, light-emitting diode

PIR: sensor de detección de infrarrojos pasivo, Passive Infrarred

LDR: fotorresistencia, Light-Dependent Resistor

USB: Universal Serial Bus

UART: Transmisor-Receptor Asíncrono Universal, Universal Asynchronous Receiver-Transmitter

FIFO: Primero en entrar, primero en salir, en inglés first in, first out

COM: puerto serie RS-232

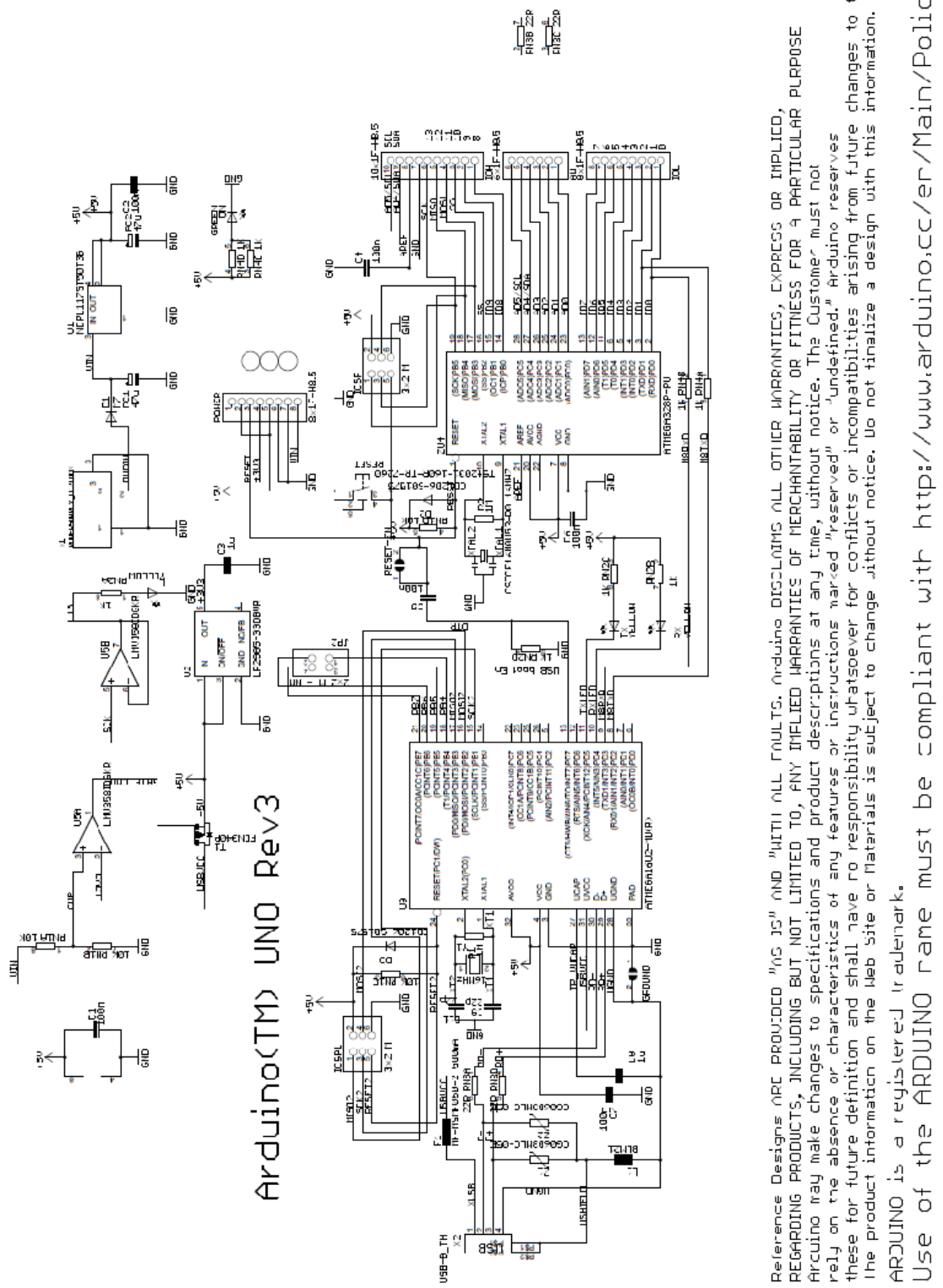
10. Bibliografía

- http://www.economist.com/blogs/economist-explains/2015/04/economist-explains-17 Noviembre 2016.
- http://www.gartner.com/newsroom/id/3270418 Noviembre 2016.
- http://www.smart-homes.nl/Domotica.aspx?lang=en-US Noviembre 2016.
- "Inside the Smart Home", Springer 2003 Dr. Richard Harper", http://www.springer.com/la/book/9781852336882 Noviembre 2016.
- https://github.com/PaulStoffregen/TimerOne/blob/master/TimerOne.h
- http://arduino.cc/ Octubre 2016.
- http://arduino.cc/en/Reference/Serial Diciembre 2016
- http://www.eventghost.org/ Diciembre 2016
- Manual EventGhost: http://www.eventghost.org/mediawiki/index.php?title=Short_Manual
- https://tasker.dinglisch.net/ Diciembre 2016
- Manual Tasker: https://tasker.dinglisch.net/guides.html
- Phil. Trans. R. Soc. Lond, On a method of expressing by signs the action of machinery. doi: 10.1098/rstl.1826.0022. 1 January 1826 vol. 116 250-265
- https://es.wikipedia.org/wiki/Atanasoff_Berry_Computer. Noviembre 2016.
- https://en.wikipedia.org/wiki/ENIAC. Noviembre 2016
- http://www.circuitstoday.com/the-story-history-of-transistor-invention. Noviembre 2016
- https://en.wikipedia.org/wiki/Transistor. Noviembre 2016
- http://museum.mit.edu/nom150/entries/518. Noviembre 2016
- https://es.wikipedia.org/wiki/Intel_4004. Noviembre 2016
- Gary B. Little. Inside the Apple 1/e. Brady Communications Company, Inc. A Prentice-Hall Publishing Company Bowie, MD 20715. Copyright © 1985 by Brady Communications Company, Inc. ISBN 0-89303-551-3
- https://en.wikipedia.org/wiki/Amiga_1000. Noviembre 2016
- https://es.wikipedia.org/wiki/Internet. Noviembre 2016

- https://www.iuma.ulpgc.es/users/jmiranda/docencia/libro_ada/libro_ada_html/node133.htm. Noviembre 2016
- https://en.wikipedia.org/wiki/OpenStack. Noviembre 2016
- Gordon E. Moore. Cramming more components onto integrated circuits. Proceedings of the ieee, VOL. 86, NO. 1, january 1998
- Gartner. Gartner Says Global Smartphone Sales to Only Grow 7 Per Cent in 2016. Gartner Says Global Smartphone Sales to Only Grow 7 Per Cent in 2016
- https://es.wikipedia.org/wiki/MIPS_(procesador). Noviembre 2016
- https://www.raspberrypi.org/. Noviembre 2016
- Dr. Richard Harper. "Inside the Smart Home", Springer 2003 "
- www.iberdrola.com/sala-comunicacion/noticias/detalle/iberdrola-supera-los-ocho-millones-de-contadores-inteligentes-instalados-en-espana. Noviembre 2016.

11. Anexos

11.1 Hoja de datos Arduino UNO



Technical specs

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

11.2 Hoja de datos ATMega328

Ver documento adjunto: Anexo A - Datasheet ATmega328.pdf

11.3 Hoja de datos PIR

Specification

1 Electrical parameters

Product Type	HC--SR501 Body Sensor Module
Operating voltage range	DC 4.5-20V
Quiescent Current	<50uA
Level output	High 3.3 V /Low 0V
Trigger	L can not be repeated trigger/H can be repeated trigger(Default repeated trigger)
Delay time	5-200S(adjustable) the range is (0.xx second to tens of second)
Block time	2.5S(default)Can be made a range(0.xx to tens of seconds
Board Dimensions	32mm*24mm
Angle Sensor	<100 ° cone angle
Operation Temp.	-15-+70 degrees
Lens size sensor	Diameter:23mm(Default)

2 Features:

1, the automatic sensor: to enter the sensor output range is high, people leave the sensor range of the automatic delay off high, output low.
2, the photosensitive control (optional, factory is not set) may set the photosensitive control during the day or light intensity without induction.

3, the temperature compensation (optional, factory is not set): In the summer when the ambient temperature rises to 30 ~ 32 ℃, slightly shorter detection range, temperature compensation can be used as a performance compensation.

4, two trigger mode: (can be selected by jumpers)

a. can not repeat the trigger: the sensor output high, the delay time is over, the output will automatically become low from high;

b. repeatable trigger: the sensor output high after the delay period, if the human body in its sensing range

Activities, its output will remain high until after the delay will be left high to low (sensor module review

Measured activities of each body will be automatically extended after a delay time, and the final event of the delay time

Starting point of time).

5, with induction blocking time (the default setting: 2.5S block time): sensor module, after each sensor output (high change

Into a low level), you can set up a blockade followed by time period, in this time period the sensor does not accept any sensor signal.

This feature can have a "sensor output time" and "blocking time" the interval between the work produced can be applied to detect the interval Products; also inhibit this function during load switching for a variety of interference. (This time can be set at zero seconds

- Tens of seconds).

6, the working voltage range: the default voltage DC4.5V-20V.

7, micro-power consumption: static current "50 microamps, especially for battery-powered automatic control products.

8, the output high level signals: types of circuits can be easily and docking.

3 Instructions:

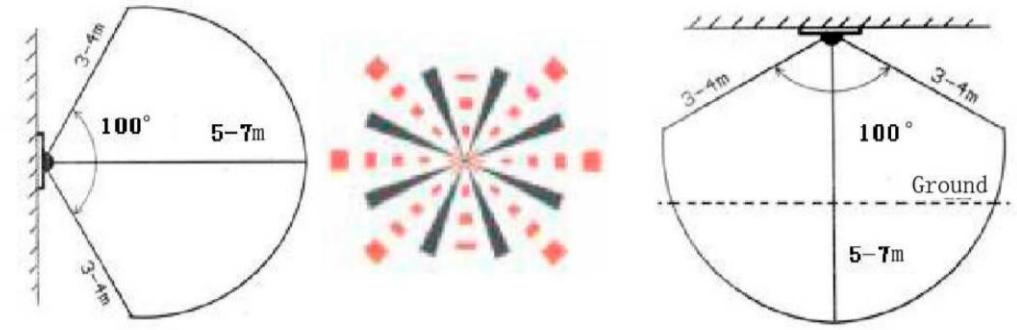
1. Sensing module for about a minute after power initialization time, during the interval to the output module

0-3 times a minute in standby mode.

2. Should avoid direct lighting such as interference sources close the surface of the lens module so as to avoid the introduction of interference signal generator malfunction; use of the environment to avoid the flow of the wind, the wind sensor will also cause interference.

3. Sensor module using a dual probe, the probe's window is rectangular, dual (A per B million) in the direction of the ends of long, when the body passed from left to right or right to left when the reach the dual IR time, distance difference, the greater the difference, more sensitive sensors, when the body from the front to the probe or from top to bottom or from bottom to top direction passing, dual IR not detected changes in the distance, no difference value, the sensor insensitive or does not work; so the sensors should be installed dual direction of the probe with human activities as much as possible parallel to the direction of maximum to ensure that the body has been passed by the dual sensor probe. To increase the sensing range of angles, the module using a circular lens, the probe also makes sense on all four sides, but still higher than the upper and lower left and right direction of sensing range, sensitivity and strong, still as far as possible by the above installation requirements. VCC, trig (control side), echo (receiving end), GND

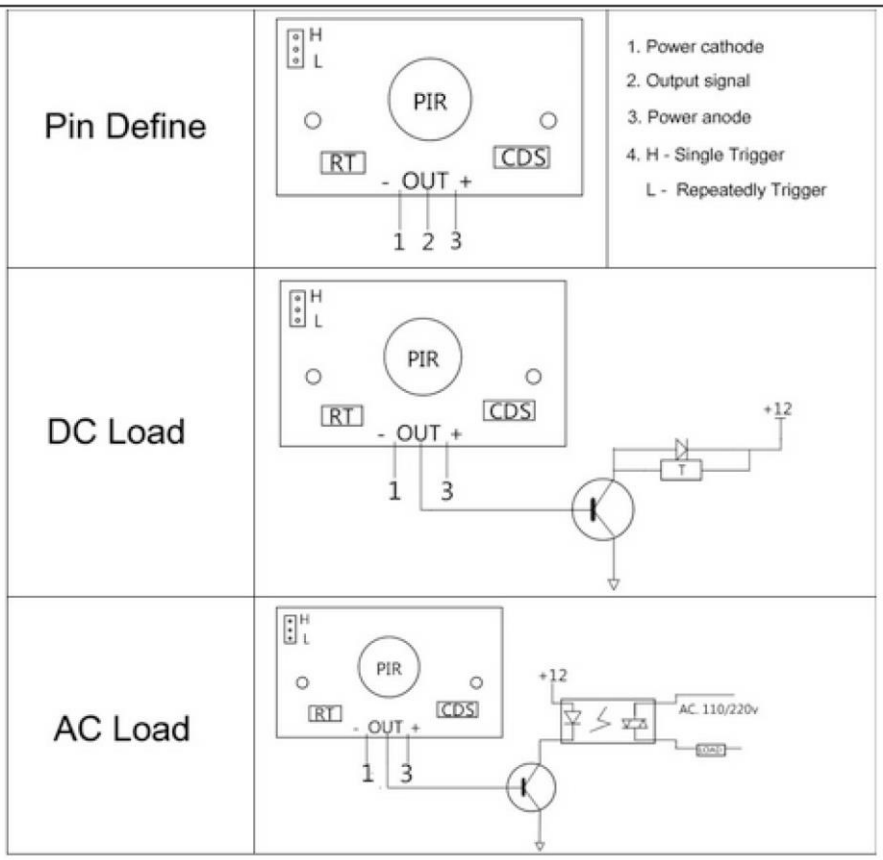
Induction Range:



Dimensions and Adjustment:

Note: The potentiometer clockwise to adjust the distance, sensing range increases (about 7 meters), on the contrary, sensing range decreases (about 3 meters).

Delay adjustment potentiometer clockwise rotation, sensor delay longer (about 300S), the other hand, induction by the short delay (about 5S).



Applications:

1, Security Products

2, the human body sensors toys

3, the human body sensor lighting

4, industrial automation and control, etc.

It can automatically and quickly open various types of incandescent, fluorescent lamps, buzzer, automatic doors, electric fans, automatic washing machine and dryer

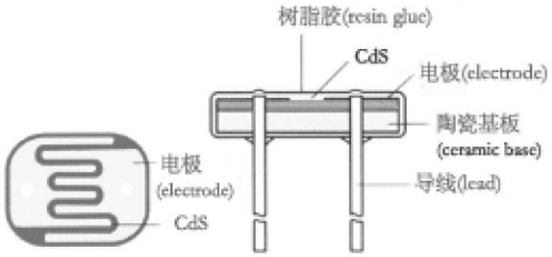
Machines and other devices, is a high-tech products. Especially suitable for enterprises, hotels, shopping malls, warehouses and family aisles, corridors and other sensitive.

11.4 Hoja de datos Fotoresistor LDR

GL55 Series Photoresistor

Photoresistor is a resistor which made of semi-conductor material,and the conductance changes with luminance variation .The photoresistor can be manufactured with different figures and illuminated area based on this characteristic.Photoresistor is widely used in many industries, such as toys,lamps,camera,etc.

Schematic Drawing



Performances and Features

- Coated with epoxy
- Small volume
- Fast response
- Good reliability
- High sensitivity
- Good spectrum characteristic

Typical Applications

- Camera automatic photometry
- Indoor ray control
- Industrial control
- Light control lamp
- Photoelectric control
- Annunciator
- Light control switch
- Electronic toy

Types and Specifications

Specification	Type	Max. Voltage	Max. power	Environmental temp.	Spectrum peak value
Φ5 series	GL5516	150	90	-30~+70	540
	GL5528	150	100	-30~+70	540
	GL5537-1	150	100	-30~+70	540
	GL5537-2	150	100	-30~+70	540
	GL5539	150	100	-30~+70	540
	GL5549	150	100	-30~+70	540

Specification	Light resistance (10Lux) (KΩ)	Dark resistance (MΩ)	γ_{10}^{100}	Response time (ms)		Illuminance resistance Fig. No.
				Increase	Decrease	
Φ5 series	5-10	0.5	0.5	30	30	2
	10-20	1	0.6	20	30	3
	20-30	2	0.6	20	30	4
	30-50	3	0.7	20	30	4
	50-100	5	0.8	20	30	5
	100-200	10	0.9	20	30	6

Test Conditions

Max. external voltage: Maximum voltage to be continuously given to component in the dark.

Dark resistance: Refer to the resistance ten seconds after the 10Lux light is shut up.

Max. power consumption: Maximum power at the environmental temperature 25℃.

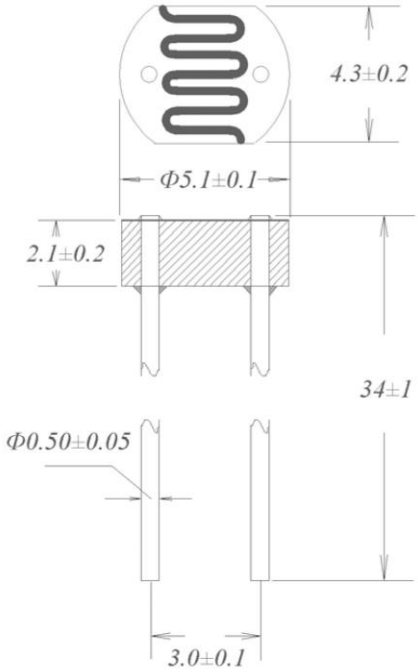
Light resistance: Irradiated by 400-600Lux light for two hours, then test with 10Lux under standard light source A(as colour temperature 2856K).

γ value: Logarithm of the ratio of the standard resistance value under 10Lux and that under 100Lux.

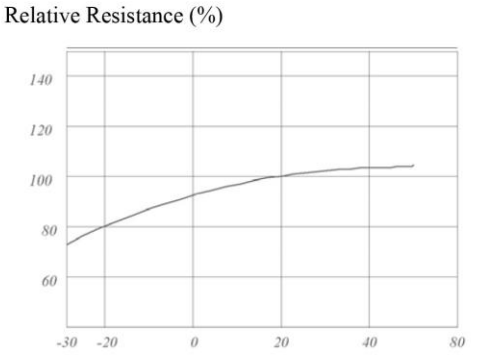
$$\gamma = \frac{Lg(R10/R100)}{Lg(100/10)} = Lg(R10/R100)$$

R10,R100 are the resistances under 10Lux and 100Lux respectively.

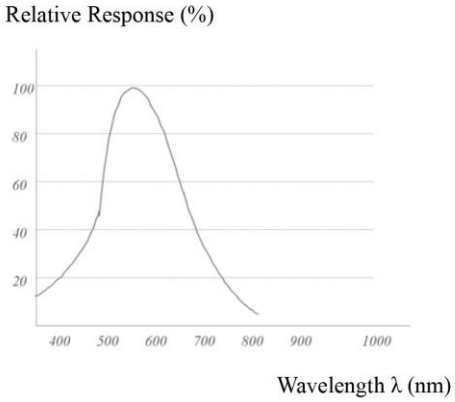
Main Characteristics Curve and Dimensions



Specification unit: mm



Temperature-Property



Spectrum Response Characteristic

Illuminance-Resistance Characteristics Curve

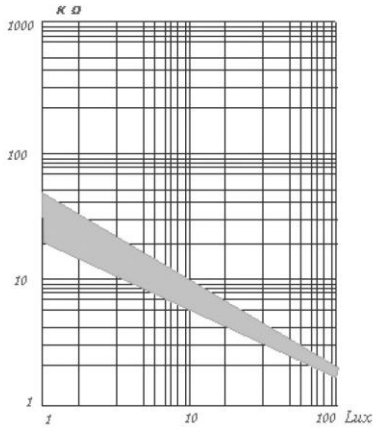


Fig. 1

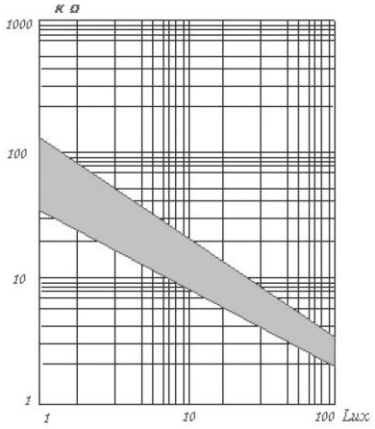


Fig.2

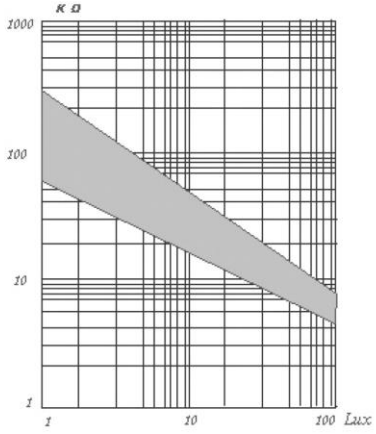


Fig. 3

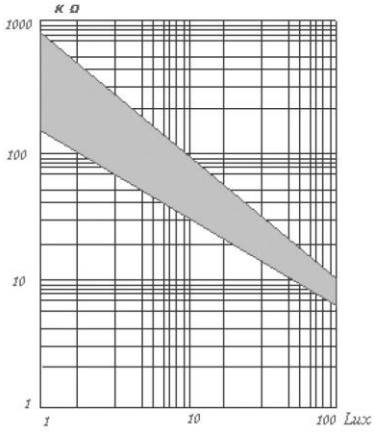


Fig. 4

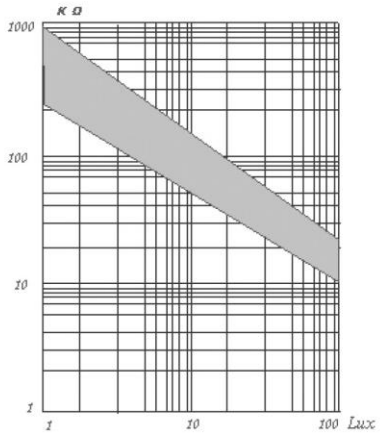


Fig.5

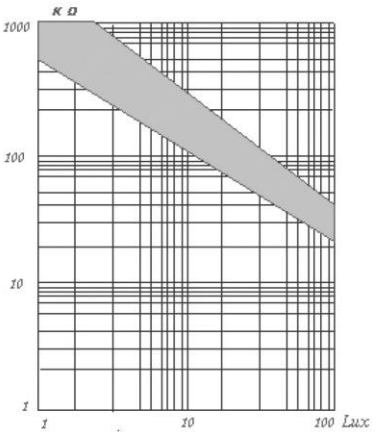


Fig. 6

Packing and Precaution

This product is packed with the environmental protection material, 100pcs per small package,1000pcs per big package.

Avoid high temperature and humidity for storing.

Soldering should be completed in the shortest possible time.

It is recommended that the soldering should keep 4mm away from ceramic substrate.

SHENZHEN SENBA OPTICAL & ELECTRONIC CO., LTD.
Add:No.3 building,huafeng Industry Area,
39 District ,BaoAn ,ShenZhen City, China
Web:www.sbcds.com.cn
E-mail:sbcds@public.szptt.net.cn
Tel:+86-755-27896456 27895411
Fax:+86-755-27897072