UNIVERSIDAD TECNOLÓGICA ISRAEL

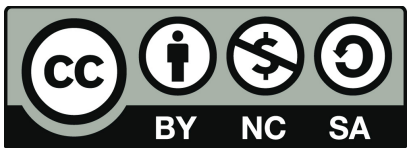
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

TEMA: Estudio, diseño e implementación de módulos de entrenamiento sobre plataforma Arduino para el Laboratorio de Microcontroladores de la Facultad de Ingeniería Electrónica de la Universidad Tecnológica Israel

AUTOR: FABRICIO MARCELO FLORES FLORES

TUTOR: ING. JUAN CARLOS ROBLES

2012



Declaración

Yo, Fabricio Marcelo Flores Flores, declaro bajo juramento que el trabajo aquí descrito, es de mí autoría; que no ha sido previamente presentado para ningún grado o calificación profesional y que consultado e investigado en base a las referencias bibliográficas que se incluyen en este documento.

_____

Fabricio Flores

Certificación

Una vez que se ha culminado la elaboración del trabajo de titulación de pregrado cuyo tema es: “Estudio, diseño e implementación de módulos de entrenamiento sobre plataforma Arduino para el Laboratorio de Microcontroladores de la Facultad de Ingeniería Electrónica de la Universidad Tecnológica Israel”, certifico que el mismo se encuentra habilitado para su defensa pública.

_____

Ing. Charles Escobar MBA

DECANO DE LA FACULTAD DE
ELECTRÓNICA Y TELECOMUNICACIONES
UNIVERSIDAD ISRAEL

Certificación

A través de la presente, Certifico que el señor Fabricio Marcelo Flores Flores ha realizado y concluido su trabajo de titulación de pregrado cuyo tema es: “Estudio, diseño e implementación de módulos de entrenamiento sobre plataforma Arduino para el Laboratorio de Microcontroladores de la Facultad de Ingeniería Electrónica de la Universidad Tecnológica Israel“, para obtener el título de Ingeniero en Electrónica y Telecomunicaciones, bajo mí tutoría.

_____

Ing. Juan Carlos Robles R

DIRECTOR DE TRABAJO DE TITULACIÓN DE PREGRADO

Agradecimiento

Quiero agradecer a doña Lolita y a don Ramiro que por su apoyo y presión, me permitieron concluir con esta meta académica; a mis hermanos que siempre han sido una motivación; y a los docentes de la Universidad Israel que han puesto su mayor esfuerzo al servicio de los estudiantes.

Dedicatoria

Dedico este trabajo de titulación de pregrado a las tres mujeres de mi vida: Dasid, Betsabe y Sarita. Las quiero mucho.

Prólogo

Los microcontroladores y la programación forman una parte importante en el desarrollo de proyectos electrónicos, por lo cual la existencia de una nueva tecnología que aporte con la simplificación y optimización en el desarrollo de proyectos es una herramienta a tomarse en cuenta. La plataforma Arduino, es esta herramienta, debido a su sencillez de programación y facilidad de conexión física.

En este trabajo de titulación de pregrado se da una introducción a la plataforma Arduino, lenguaje de programación, entorno de desarrollo y las características técnicas de la placa Arduino Uno. Además se incluye el diseño e implementación de un módulo de entrenamiento sobre plataforma Arduino, el diseño está basado en la utilización de elementos digitales y analógicos para que funcionen como sistemas de entrada y salida para la placa Arduino. A través de los componentes electrónicos instalados en el módulo de entrenamiento se diseñan prácticas para aprender el lenguaje de programación. Lenguaje basado en C y C++ simplificado.

Cada práctica se encuentra con su respectivo programa para ser grabado y compilado en la placa Arduino y con su diagrama esquemático de conexión para facilitar la implementación en el módulo de entrenamiento.

Para finalizar se incluye conclusiones y recomendaciones para la mejor utilización del módulo y de las prácticas diseñadas.

Abstract

The microcontrollers and the programming are an important part in the development of electronic projects, for which the existence of a new technology that provides the simplification and optimization of project development is a tool to be considered. The Arduino platform is the tool, due to its simplicity and ease of programming and physical connection.

In this work undergraduate degree gives an introduction to the Arduino platform, language programming, development environment and the technical characteristics of the Arduino Uno board. It also includes the design and implementation of a training module for Arduino platform, the design is based on the use of digital and analog components to function as input and output systems for the Arduino board. Through the electronics components installed on the training module is designed to learn practical programming language. Language based on C and C + + simplified.

Each practice is with its own program to be recorded and compiled into the Arduino board schematic diagram and its connection to facilitate implementation in the training module.

Finally conclusions and recommendations are included for the best use of the module and practices designed.

ÍNDICE

CAPÍTULO I. PROBLEMATIZACIÓN

CAPÍTULO II. MARCO TEÓRICO

CAPÍTULO III. DISEÑO E IMPLEMENTACIÓN

ÍNDICE DE FIGURAS

CAPÍTULO II. MARCO TEÓRICO

CAPÍTULO III. DISEÑO E IMPLEMENTACIÓN

ÍNDICE DE TABLAS

CAPÍTULO II. MARCO TEÓRICO

CAPÍTULO III. DISEÑO E IMPLEMENTACIÓN

CAPÍTULO IV. ANÁLISIS FINANCIERO

Capítulo I

PROBLEMATIZACIÓN

1.1. Introducción

El mcu[1] PIC[2] de Microchip es el primer elemento de programación electrónica que el alumno de la Facultad de Ingeniería Electrónica de la UISRAEL aprende a programar, y con este a diseñar e implementar proyectos electrónicos. Pero la materia de Microcontroladores se la enseña en los últimos semestres de la carrera, dando poco tiempo para que el estudiante profundice y de el mejor provecho a la programación en PIC.

Para armar un circuito electrónico con el PIC como elemento central, es necesario elementos mínimos como la alimentación entre 3 a 5.5 voltios y los elementos externos a manipularse, cuya conexión no resulta complicada pero puede dar cierta dificultad a las personas poco familiarizadas con las conexiones en protoboard o placas PCB[3].

El lenguaje de programación utilizado para el PIC en la UTECI es el Visual Basic. La mayoría de compiladores para este lenguaje pueden utilizarse únicamente bajo ambiente Windows, y la universidad se ha mantenido con el uso de sistemas operativos bajo software libre (Ubuntu).

1.2. Antecedentes

En la malla curricular de la Facultad de Ingeniería Electrónica de la Universidad Tecnológica Israel consta la materia de Microcontroladores. La Facultad enseña

1 mcu: Micro-Controller Unit, microcontrolador.
2 PIC: Peripheral Interface Controller, Controlador de Interfases Periféricas.
3 PCB: Printed Circuit Board, Placa de Circuito Impreso.

a sus alumnos los funcionamientos, configuraciones y aplicaciones de microcontroladores usando el PIC, para comprender su programación es necesario un conocimiento un poco más amplio de sistemas informáticos y de electrónica. Pero si se quiere aplicar la programación desde los niveles más básicos y sin la necesidad de un conocimiento tan profundo es preferible la utilización de plataformas, y en la plataforma Arduino se tiene la ventaja de bajos costos, lenguaje de comunicación no complejo, compatibilidad entre software y open – hardware.

Para facilitar las implementaciones de diseños de circuitos electrónicos una opción recomendada es la utilización de módulos de entrenamiento, ya que se tienen los elementos necesarios para circuitos básicos, reduce los errores de conexión y ayuda al estudiante a centrarse en la programación.

Un módulo de entrenamiento sobre plataforma Arduino, está orientado para la utilización de los alumnos de los primeros niveles de la carrera por su sencillez de programación y conexión, así se adentran a la electrónica tempranamente. Arduino a diferencia de la mayoría de plataformas es escalable y permite la conexión con otros circuitos existentes a través de Shields[4], con esto se puede ampliar su uso, dándole una mayor variedad de aplicaciones.

4 Ver punto 2.2.2.

1.3. Problema investigado

1.3.1. Problema principal

La Facultad de Electrónica de la Universidad Tecnológica Israel no posee en su Laboratorio de Microcontroladores módulos de entrenamiento sobre plataforma Arduino.

1.3.2. Problemas secundarios

- Los alumnos de niveles inferiores no tienen en los laboratorios de Electrónica sencillos módulos de entrenamiento para programación.
- No existen las prácticas y sus guías que utilicen los elementos disponibles en los módulos de entrenamiento sobre plataforma Arduino.
- El Laboratorio de Microcontroladores no utiliza software libre como sistema operativo.

1.4. Formulación del problema

¿Se puede implementar módulos de entrenamiento para el aprendizaje de programación en niveles básicos dirigida a alumnos de primeros niveles?

1.5. Justificación

1.5.1. Justificación teórica

La investigación se desarrolló en el marco de la investigación – acción y de la investigación participativa.

Se utilizó métodos de observación para analizar resultados y la interacción entre docente, alumno y prácticas de laboratorio.

Al implementar el laboratorio de plataforma Arduino se complementa lo aprendido teóricamente en las aplicaciones de microcontroladores.

1.5.2. Justificación metodológica

Se utilizaron los siguientes métodos de investigación de acuerdo al desarrollo del proyecto:

- Método de análisis que permitió determinar la realidad actual en la que se encuentra la enseñanza teórica – práctica de la sección de microcontroladores.
- Métodos deductivo e inductivo se aplicaron con el fin de determinar las prácticas que más se acoplen a la malla curricular y acorde a la realidad de proyectos a desarrollarse en la universidad.
- Con el método experimental se determinó la validez de las prácticas de laboratorio.

1.5.3. Justificación práctica

El desarrollo de las prácticas con Arduino permite tener una plataforma de sencillo entendimiento y dar el conocimiento al estudiante para ingresar al mundo de los microcontroladores.

Al estar la materia de Microcontroladores en los últimos semestres, el alumno tiene el suficiente conocimiento para desarrollar proyectos con mayor complejidad y desempeñar de mejor manera su carrera profesional teniendo ya las bases desde los niveles inferiores de la carrera con la plataforma Arduino.

1.6. Objetivos

1.6.1. Objetivo principal

Estudiar, diseñar e implementar módulos de entrenamiento sobre plataforma Arduino para el Laboratorio de Microcontroladores de la Facultad de Ingeniería Electrónica de la Universidad Tecnológica Israel.

1.6.2. Objetivos específicos

- Implementar módulos de entrenamiento sobre plataforma Arduino, dirigido a los alumnos de niveles inferiores.
- Desarrollar y escribir las guías prácticas que contengan las conexiones físicas y configuraciones lógicas con los diferentes elementos instalados en el módulo de entrenamiento sobre plataforma Arduino.
- Utilizar software libre en el sistema operativo y en el compilador para los módulos de entrenamiento sobre plataforma Arduino.

1.7. Metodología científica

El sistema metodológico usado para la recopilación de información fue el método de investigación bibliográfico; al ser Arduino una tecnología relativamente nueva no existe una amplia documentación acerca del tema, incluso los libros disponibles son bastante limitados. La investigación necesaria se basó en páginas web e información de diferentes portales de internet, en este caso se debió aplicar el método lógico deductivo y el experimental para a

través de pruebas y deducción obtener la información más certera al no tener una fuente fiable o certificada.

Capítulo II

MARCO TEÓRICO

2.1. Arduino[5]

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos.

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador de la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring[6]) y el entorno de desarrollo Arduino (basado en Processing[7]). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software.

Las placas pueden ser hechas a mano o comprarlas montadas de fábrica; el software puede ser descargado de forma gratuita. Los ficheros de diseño de referencia CAD[8] están disponibles bajo una licencia abierta, así pues se es libre de adaptarlos a las necesidades.

“Arduino recibió una Mención Honorífica en la sección *Digital Communities* de

5 http://www.arduino.cc/es/
6 http://www.wiring.org.co
7 http://www.processing.org
8 CAD: Computer-aided design, Diseño asistido por computadora.

la edición del 2006 del *Ars Electrónica Prix*".

2.2. Hardware

Hay multitud de diferentes versiones de placas Arduino. La actual placa básica, la Uno (figura 2.1), usa Atmel ATmega328[9], al igual que la Duemilanove. La anterior Diecimila, y las primeras unidades de Duemilanove usaban el Atmel ATmega168, mientras que las placas más antiguas usan el ATmega8. El Arduino Mega está basado en el ATmega1280.

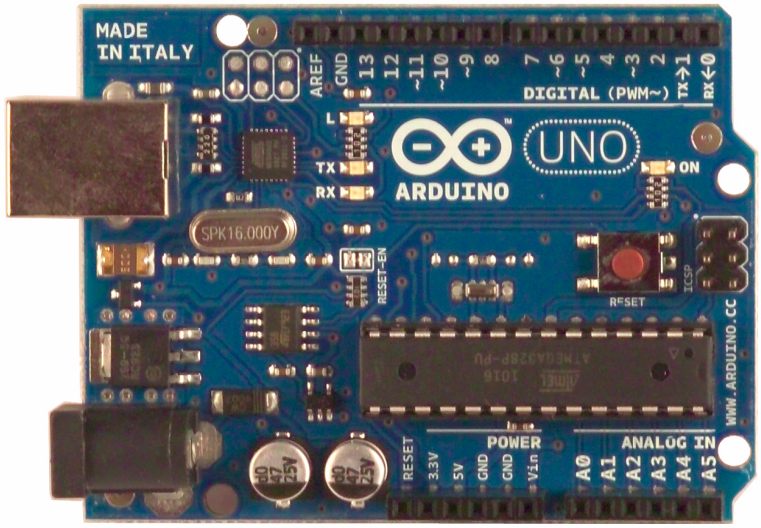


Figura 2.1. Placa Arduino Uno[10]

2.2.1. Placas de E/S

- Uno.- Esta es la última revisión de la placa Arduino USB básica. El Uno se diferencia de todas las placas anteriores, ya que no utiliza el chip FTDI USB - serie. Por el contrario, utiliza ATmega8U2 programado como

9 ATmega328 es un microcontrolador de la compañía Atmel que cuenta con 32KB de memoria flash, 2KB de memoria RAM y 1KB de memoria EEPROM.

10 http://arduino.cc/en/Main/ArduinoBoardUno

un convertidor de USB a serie. Figura 2.1.

- Duemilanove.- "Duemilanove" significa 2009 en italiano que fue el año cuando salió al mercado. El Duemilanove es el más popular en dentro de las series de placas con USB. Figura 2.2.

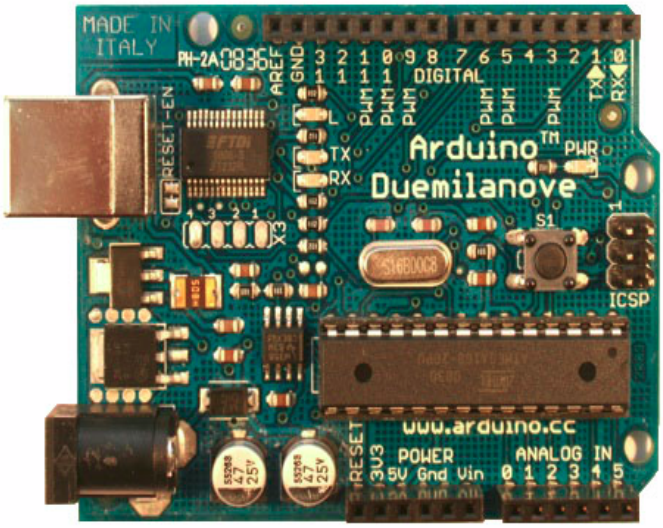


Figura 2.2. Placa Arduino Duemilanove[11]

- Diecimila.- Fue la primera placa Arduino en integrar el chip FTDI para convertir de USB a serie y alimentarse con la energía proporcionada por la salida USB de la PC.

- Serial.- Placa básica que utiliza interfaz RS232 como comunicación con el ordenador para programar o intercambiar datos.

- Serial a una cara.- Está diseñada para ser trazada y montada a mano, es un poco más grande que la Uno, pero compatible con los shields. Figura 2.3.

11 http://arduino.cc/es/Main/ArduinoBoardDuemilanove

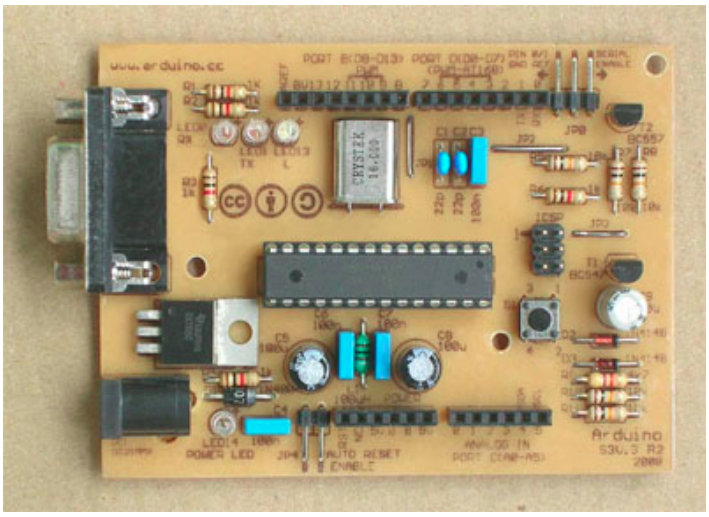


Figura 2.3. Placa Arduino Serial a una cara.[12]

- Nano.- Una placa diseñada para usar directamente en placas de desarrollo, el Nano se conecta al ordenador con un cable Mini – B USB.
- Mega.- Más grande y potente placa Arduino, compatible con los shields de Uno, Duemilanove y Diecimila.
- Bluetooth.- El Arduino BT contiene un módulo bluetooth que permite comunicarse y programarse sin cables. Es compatible con los shields de Arduino.
- LilyPad.- Diseñado para aplicaciones sobre prendas, esta placa puede ser cosida a la ropa y es de color púrpura.
- Fio.- Diseñada para aplicaciones inalámbricas. Incluye un zócalo para Xbee[13], un conector para baterías LiPo[14] y electrónica para cargar

12 http://arduino.cc/es/Main/ArduinoBoardSerialSingleSided3
13 Los módulos Xbee son módulos de radio frecuencia que trabajan en la banda de 2.4 GHz con protocolo de comunicación 802.15.4.
14 Polímero de Litio.

baterías.

- Mini.- La placa Arduino más pequeña. Funciona perfectamente en una placa de desarrollo o en aplicaciones donde el espacio es primordial. Se conecta al ordenador usando el adaptador Mini USB.
- Pro.- Está diseñada para aquellos que quieren dejar la placa incrustada en un sistema; es más barata que la Uno y se puede alimentar fácilmente con baterías, pero requiere de componentes extra y montaje.

2.2.2. Shields Arduino

Un shield es una placa impresa que se pueden conectar en la parte superior de la placa Arduino para ampliar sus capacidades, pudiendo ser apilada una encima de la otra.

Las shields suelen ser diseños bastante simples y en general de código abierto y publicados libremente. Pero también se pueden encontrar shields un poco más sofisticados como una unidad de medida inercial con una estructura en seis giroscopios DOF[15] para ser usados como parte de un piloto automático en un avión de aeromodelismo.

Entre las shields más conocidas se tiene las siguientes:

- Arduino Ethernet Shield
- Arduino microSD Shield
- Arduino Relay Shield

15 Degree of freedom (grado de libertad)

- Arduino Celular Shield SM5100B
- Arduino Xbee Shield
- Arduino GPS Shield
- Arduino Motor Shield

2.2.2.1. Arduino Ethernet Shield

La Arduino Ethernet Shield (figura 2.4) permite a una placa Arduino conectarse a internet o a una red LAN. Está basada en el chip ethernet Wiznet W5100. El Wiznet W5100 provee de una pila de red IP capaz de TCP y UDP. Soporta hasta cuatro conexiones de sockets simultáneas. Usa la librería ethernet para escribir programas que se conecten por red usando la shield.

La Ethernet Shield dispone de unos conectores que permiten conectar a su vez otras placas encima y apilarlas sobre la placa Arduino.

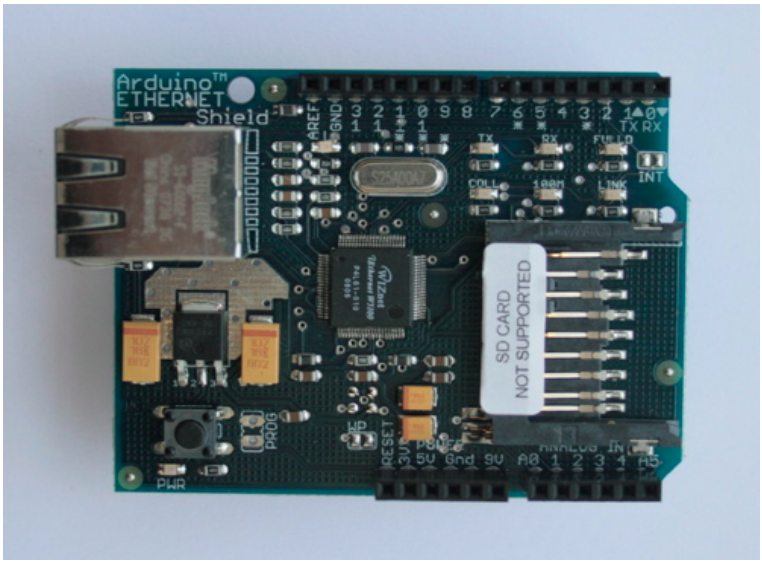


Figura 2.4. Arduino Ethernet Shield[16]

16 http://arduino.cc/es/Main/ArduinoEthernetShield

Arduino usa los pines digitales 10, 11, 12 y 13 (SPI) para comunicarse con el W5100 en la ethernet shield. Estos pines no pueden ser usados para E/S genéricas.

La shield provee un conector ethernet estándar RJ45.

El botón de reset en la shield resetea ambos, el W5100 y la placa Arduino.

La shield contiene un número de LEDS para información:

- PWR: Indica que la placa y la shield están alimentadas.
- LINK: Indica la presencia de un enlace de red y parpadea cuando la shield envía o recibe datos.
- FULL D: Indica que la conexión de red es full duplex.
- 100M: Indica la presencia de una conexión de red de 100 Mb/s.
- RX: Parpadea cuando la shield recibe datos
- TX: Parpadea cuando la shield envía datos.
- COLL: Parpadea cuando se detectan colisiones en la red.

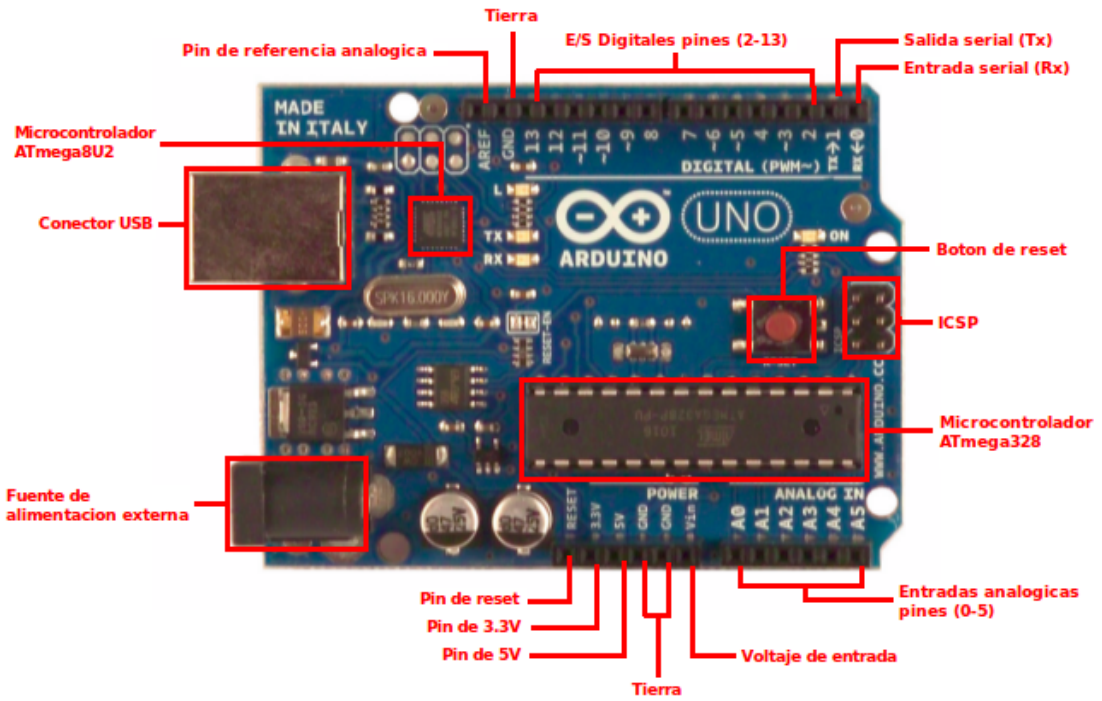
El jumper soldado marcado como INT puede ser conectado para permitir a la placa Arduino recibir notificaciones de eventos por interrupción desde el W5100, pero esto no está soportado por la librería ethernet. El jumper conecta el pin INT del W5100 al pin digital 2 de Arduino.

El slot SD en la shield no está soportado por el software Arduino.

2.3. Arduino Uno

El Arduino Uno (figura 2.1) es una placa con microcontrolador basado en el ATmega328. Tiene 14 pines con entradas/salidas digitales (6 de las cuales pueden ser usadas como salidas PWM), 6 entradas analógicas, un cristal oscilador a 16Mhz, conexión USB, entrada de alimentación DC[17], una cabecera ICSP[18], y un botón de reset. Contiene todo lo necesario para utilizar el microcontrolador; simplemente se conecta a un ordenador a través del cable USB para alimentarlo también se puede utilizar u adaptador o una batería para empezar a trabajar.

En la figura 2.5 se observa la placa Arduino Uno con sus componentes físicos y todos los pines disponibles.



17 DC: Direct Current, Corriente Directa.

18 ICSP o "In Chip Serial Programmer" es el método de acceso a toda la memoria de programa de un procesador Atmel. Sirve para poder programar el bootloader de Arduino. El bootloader es el programa básico que escucha al puerto serie y así poder descargar programas desde la IDE (Gestor de arranque del sistema).

Figura 2.5. Componentes de la placa Arduino Uno

2.3.1. Características Generales

Microcontrolador	ATmega328
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines E/S digitales	14 (6 proporcionan salida PWM)
Pines de entrada analógica	6
Intensidad por pin	40 mA
Intensidad en pin 3.3V	50 mA
Memoria Flash	32 KB de las cuales 2 KB las usa el gestor de arranque(bootloader)
SRAM	2 KB
EEPROM	1 KB
Velocidad de reloj	16 MHz

Tabla 2.1. Características generales de la placa Arduino[19]

19 http://arduino.cc/es/Main/ArduinoBoardDuemilanove

2.3.2. Alimentación

Puede ser alimentado vía conexión USB o con una fuente de alimentación externa DC. El origen de la alimentación se selecciona automáticamente.

Las fuentes de alimentación externas (no USB) pueden ser tanto un adaptador de pared AC[20]/DC o una batería. El adaptador se puede conectar usando un conector macho de 2.1 mm con centro positivo en el conector hembra de la placa. Los cables de la batería a los pines GND y Vin en los conectores de alimentación POWER.

La placa puede trabajar con una alimentación externa de entre 6 a 20 V. Si el voltaje suministrado es inferior a 7 V el pin de 5 V puede proporcionar menos de 5 V y la placa puede volverse inestable, si se usan más de 12 V los reguladores de voltaje se pueden sobre calentar y dañar la placa. El rango recomendado es de 7 a 12 V.

Los pines de alimentación son los siguientes:

- **Vin:** Se puede proporcionar voltaje a través de este pin, o, si se está alimentando a través de la conexión de 2.1 mm, acceder a ella a través de este pin (7 a 12 V).
- **5V:** Es el pin de salida de voltaje estabilizado de 5 V, que es proporcionado por el Vin a través de un regulador integrado a la placa o directamente de la USB.
- **3V3:** Es una fuente de 3.3 V, generada por el regulador incluido en la placa. La corriente máxima soportada es de 50 mA.
- **GND:** Pines de toma a tierra.

20 AC: Alternating Current, Corriente Alterna.

2.3.3. Memoria

El ATmega328 tiene 32 KB de memoria flash para almacenar códigos, 2 KB son usados para el arranque del sistema (bootloader). Tiene 2 KB de memoria SRAM. Posee 1 KB de EEPROM, a la cual se puede acceder para leer o escribir.

2.3.4. Entradas y Salidas

Cada uno de los 14 pines digitales pueden utilizarse como entradas o salidas usando las funciones pinMode(), digitalWrite() y digitalRead(). Las E/S operan a 5 V. Cada pin puede proporcionar o recibir una intensidad máxima de 40 mA y tienen una resistencia interna, pull up, (desconectada por defecto) de 20 KΩ. Además, algunos pines tienen funciones especializadas:

- **Serie: pin 0 (RX) y pin 1 (TX).** Usado para recibir (RX) y transmitir (TX) datos a través del puerto serie TTL. Estos pines están conectados en paralelo a los pines correspondientes del Atmega8U2 y a los pines RXD y TXD del Atmega.
- **Interrupciones Externas: pin 2 y pin 3.** Estos pines se pueden configurar para lanzar una interrupción en un valor LOW (0V), en flancos de subida o bajada (cambio de LOW a HIGH o viceversa), o en cambios de valor.
- **PWM: pines 3, 5, 6, 9 ,10 y 11.** Proporciona una salida PWM (Pulse Wave Modulation, modulación por onda de pulso) de 8 bits de resolución con valores de 0 a 255. Se los identifica por el símbolo ~ en la placa Arduino.

- **SPI[21]: pines 10 (SS), 11 (MOSI), 12 (MISO) y 13 (SCK).** Estos pines proporcionan comunicación SPI, que ha pesar de que el hardware la proporcione actualmente no está incluido en el lenguaje Arduino.
- **LED: pin 13.** Hay un led integrado en la placa conectado al pin digital 13, cuando este pin tiene un valor HIGH (5V) el led se enciende y cuando éste tiene un valor LOW (0V) este se apaga.

La Uno tiene 6 entradas analógicas y cada una de ellas proporciona una resolución de 10 bits (1024 valores). Por defecto se mide de tierra a 5 V, aunque es posible cambiar la cota superior de este rango usando el pin AREF y la función analogReference(). Además algunos pines tienen funciones especializadas:

- **I²C: pin 4 (SDA) y pin 5 (SCL).** Soporte del protocolo de comunicaciones I²C (TWI) usando la librería Wire.

Hay otros pines en la placa:

- **AREF.** Voltaje de referencia para las entradas analógicas. Configura el voltaje de referencia usado por la entrada analógica. La función analogRead() devolverá un valor de 1023 para aquella tensión de entrada que sea igual a la tensión de referencia. El valor del voltaje debe estar en el rango de 0 a 5 V.
- **Reset.** Suministra un valor LOW (0V) para reiniciar el microcontrolador. Típicamente usado para añadir un botón de reset a los Shields que no permiten acceso a la placa.

21 SPI (Serial Peripheral Interface, comunicación serial sincrónica)

2.3.5. Comunicación

EL Arduino Uno facilita en varios aspectos la comunicación con el ordenador, otro Arduino o otros microcontroladores. El ATmega328 proporciona comunicación vía serie UART TTL (5V), disponible a través de los pines digitales 0(RX) y 1(TX). Un microcontrolador ATmega8U2 integrado en la placa que canaliza la comunicación serie a través del USB y proporcionan un puerto serie virtual en el ordenador. El software incluye un monitor de puerto serie que permite enviar y recibir información textual de la placa Arduino. Los LEDs RX y TX de la placa parpadearán cuando se detecte comunicación transmitida a través del Atmega8U2 y la conexión USB (no parpadearán si se usa la comunicación serie a través de los pines 0 y 1).

La librería SoftwareSerial permite comunicación serie por cualquier par de pines digitales del Uno.

El ATmega328 también soportan la comunicación I2C (TWI) y SPI . El software de Arduino incluye una librería Wire para simplificar el uso el bus I2C.

2.3.6. Protección contra sobre tensiones en el USB

El Arduino Uno tiene un multi fusible reiniciable que protege la conexión USB del ordenar de cortocircuitos y sobre tensiones. A parte que la mayoría de ordenadores proporcionan su propia protección interna, el fusible proporciona una capa extra de protección. Si más de 500 mA son detectados en el puerto USB, el fusible automáticamente corta la conexión hasta que el cortocircuito o la sobre tensión desaparece.

2.3.7. Características físicas

La longitud y amplitud máxima de la placa Uno es de 2,7 y 2,1 pulgadas respectivamente, con el conector USB y la alimentación externa sobresaliendo de estas dimensiones. Tres agujeros para fijación con tornillos permite colocar la placa en superficies y cajas.

2.4. Software para Arduino

El entorno de código abierto Arduino hace fácil escribir el código y cargarlo a la placa de E/S. Funciona en Windows, Mac OS X, Linux y Android. El entorno está escrito en Java y basado en Processing, avr-gcc y otros programas también de código abierto.

2.4.1. Entorno Arduino

El entorno Arduino también es conocido como IDE[22] para Arduino, que se puede ver en la figura 2.6.

El entorno de desarrollo está constituido por un editor de textos para escribir el código, una área de mensajes, una consola de textos, una barra de herramientas con botones para las funciones comunes y una serie de menús. Permite la conexión con el hardware de Arduino para cargar los programas y comunicarse con ellos.

Arduino utiliza para escribir el software lo que se denomina sketch (programa). Estos programas son escritos en el editor de texto. Existe la posibilidad de cortar, pegar y buscar/reemplazar texto. En el área de mensajes se muestra

22 IDE. Integrated Devolpment Environment. Entorno de desarrollo integrado, es un programa informático compuesto por un conjunto de herramientas de programación.

información mientras se carga el programa y también muestra errores. La consola muestra el texto de salida para el entorno de Arduino incluyendo los mensajes de error completos y otras informaciones. La barra de herramientas permite verificar el proceso de carga, creación, apertura y guardado de programas, y la monitorización serie.

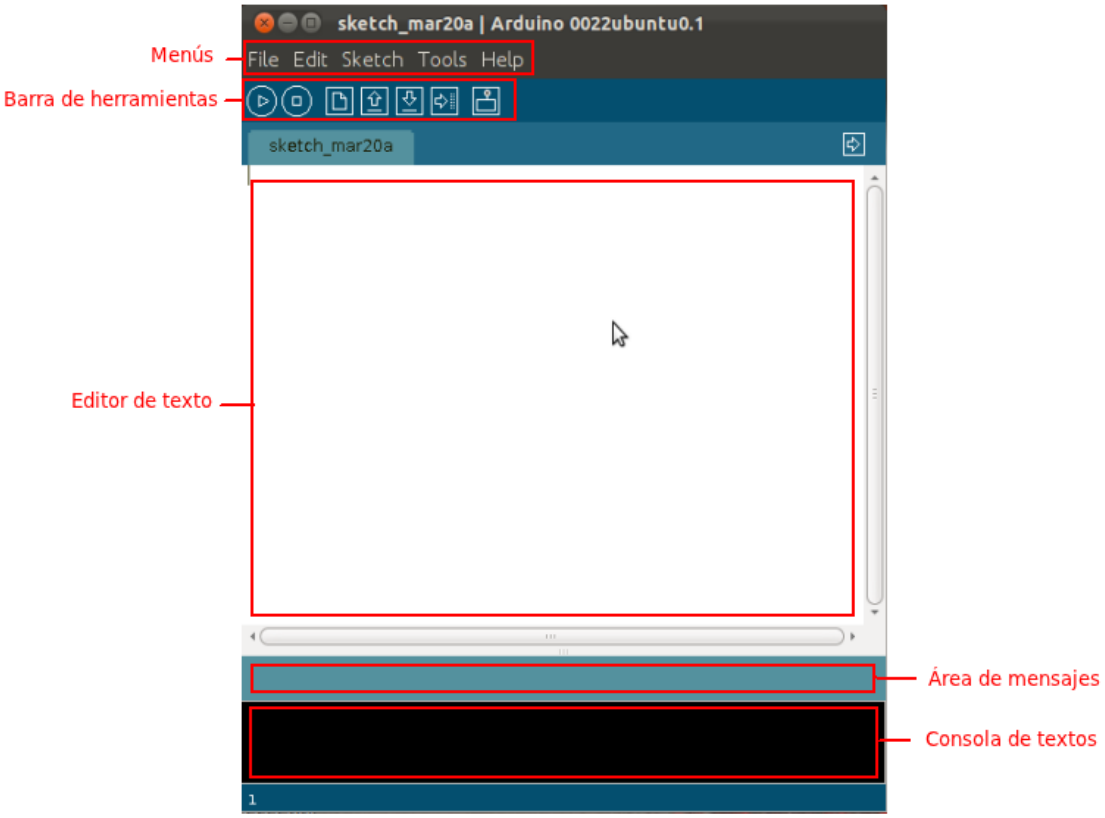


Figura 2.6. IDE para Arduino

2.4.1.1. Barra de herramientas

Verify/Compile

Chequea el código en busca de errores.

Stop

Finaliza la monitorización serie y oculta otros botones.

New

Crea un nuevo sketch.

Open

Presenta un menú de todos los programas sketch "sketchbooks" (librería sketch).

Save

Guarda el programa sketch.

Upload to I/O Board

Compila el código y lo vuelca en la placa E/S Arduino.

Serial Monitor

Inicia la monitorización serial.

2.4.1.2 Menús

El entorno de Arduino contiene una serie de menús que son sensibles al contexto, esto quiere decir que sólo se habilitan de acuerdo a la acción que se esté realizando. Los menús son los siguientes:

- File
    - New

- Open
- Sketchbook: Abre los sketch creados.
- Examples: Abre los sketch que vienen de ejemplo al descargar el IDE para Arduino.
- Close
- Save
- Save As
- Upload to I/O Board: Compila el código y lo vuelca en la placa E/S Arduino, esta opción también se la encuentra en la barra de herramientas.
- Page Setup
- Print
- Preferences: Permite modificar opciones del editor, sobre todo la ubicación para almacenar los sketch creados.
- Quit

- Edit
  - Undo
  - Redo
  - Cut
  - Copy
  - Copy for Forum: Copia al portapapeles el texto del sketch seleccionado, el texto posee el formato usado en los foros.

    - Copy as HTML: Copia al portapapeles el texto del sketch seleccionado, el texto tiene el formato HTML para páginas web.
    - Paste
    - Select All
    - Comment/Uncomment: Inicia y finaliza los comentarios en el sketch.
    - Increase Indent: Aumentar el guión – sangría.
    - Decrease Indent: Disminuir el guión - sangría
    - Find
    - Find Next
- Sketch
    - Verify/Compile: Verifica los errores del programa (sketch).
    - Stop: Detiene la verificación de los errores.
    - Show Sketch Folder: Abre la carpeta de programas (sketch) en el escritorio.
    - Import Library: Añade una librería al programa, se incluye la sentencia #include en el código.
    - Add File: Añade un fichero fuente al programa que se esté realizando.
- Tools
    - Auto Format: Da formato al código proporcionando estética, por ejemplo realiza tabulaciones entre la apertura y cierre de llaves, y las sentencias que tengan que ser tabuladas lo estarán.

- Archive Sketch
- Fix Encoding & Reload: En caso de que el sketch se lo realice con una versión antigua del Arduino/Processing y se quiera compilar en un IDE más actualizado esto da un conflicto y se presenta mensajes de error, esta opción permite las actualizaciones necesarias para que se pueda compilar el sketch en la placa E/S Arduino.
- Serial Monitor: Inicia la monitorización serial, esta opción también se la encuentra en la barra de herramientas.
- Board: Permite seleccionar la placa que se esté utilizando.
- Serial Port: Contiene todos los dispositivos seriales (virtuales o reales).
- Burn Bootlander: Permite grabar un gestor de arranque (bootloader) dentro del microcontrolador de la placa Arduino. Aunque no es un requisito para el normal funcionamiento de la placa Arduino, es útil si se compra un nuevo ATmega (el cual viene normalmente sin gestor de arranque). Asegurándose que se ha seleccionado la placa correcta en el menú Boards antes de grabar el bootloader.

- Help: Permite revisar la información de Arduino, página web, preguntas frecuentes y otras opciones de ayuda; para lo cual se necesita conexión a internet.

2.5. Programación de Arduino

La programación de la placa Arduino se la realiza a través del IDE para crear el sketch, el lenguaje usado es el Processing manteniendo la estructura de este.

2.5.1. Estructura

La estructura básica del lenguaje de programación de Arduino se compone de al menos dos partes. Estas dos partes necesarias, o funciones, encierran bloques que contienen instrucciones y son las siguientes:

- void **setup**()
- void **loop**()

setup() *inicialización*

La función setup() se establece cuando se inicia un programa (sketch). Se emplea para iniciar variables, establecer el estado de los pines e inicializar las comunicaciones. Esta función se ejecutará una única vez después de que se conecte la placa Arduino a la fuente de alimentación, o cuando se pulse el botón de reinicio de la placa.

```
void setup()
{
  pinMode(pin, OUTPUT);        // ajusta a “pin” como salida
}
```

loop() *bucle*

Luego de crear la función **setup**(), la cual inicializa y prepara los valores iniciales, la función **loop**() hace justamente lo que su nombre sugiere, por lo

tanto se ejecuta continuamente, permitiéndole al programa variar y responder, leyendo entradas, activando salidas, etc. Esta función es el núcleo de todos los programas de Arduino y hace la mayor parte del trabajo.

```
void loop()
{
  digitalWrite(pin, HIGH);   // activa “pin”
  delay(1000);               // espera 1000 milisegundos
  digitalWrite(pin, LOW);    // desactiva “pin”
  delay(1000);               // espera 1000 milisegundos
}
```

2.5.2. Sintaxis

La sintaxis del lenguaje es muy parecida a la de C y C++, manteniendo las mismas estructuras, que son las siguientes:

- Funciones
- Llaves
- Punto y coma
- Bloques de comentarios
- Comentarios de línea

Funciones

Una función es un bloque de código que tiene un nombre y un grupo de declaraciones que se ejecutan cuando se llama a la función. Se puede hacer uso de funciones integradas como **void** setup() y **void** loop() o escribir nuevas. Las funciones se escriben para ejecutar tareas repetitivas y reducir el desorden en un programa. En primer lugar se declara el tipo de la función, que será el

valor retornado por la función (int, void...). A continuación del tipo, se declara el nombre de la función y, entre paréntesis, los parámetros que se pasan a la función.

```
tipo nombre_funcion(parametros)
{
  realizar esta acción;
}
```

La siguiente función int retardo(), asigna un valor de retardo en un programa por lectura del valor de un potenciómetro.

```
int retardo()
{
  int v;                          // crea una variable temporal “v”
  v = analogRead(pot);            // lee el valor analógico de “pot” y lo asigna a “v”
  v /= 4;                         // convierte de 0-1023 a 0-255
  return v;                       // devuelve el valor final de “v”
}
```

Llaves {}

Las llaves definen el comienzo y el final de bloques de función y bloques de declaraciones como void **loop**() y sentencias for e if. Las llaves deben estar balanceadas (a una llave de apertura { debe seguirle una llave de cierre }). Las llaves no balanceadas provocan errores de compilación.

```
void loop()
{
  realizar esta acción;
}
```

El entorno Arduino incluye una práctica característica para chequear el balance de llaves. Sólo selecciona una llave y su compañera lógica aparecerá resaltada.

Punto y coma ;

Un punto y coma debe usarse al final de cada declaración y separa los elementos del programa. También se usa para separar los elementos en un bucle for.

```
int x = 13;                    //declara la variable “x” como el entero 13
```

Nota: Olvidar un punto y coma al final de una declaración producirá un error de compilación.

Bloques de comentarios /*...*/

Los bloques de comentarios, o comentarios multilínea, son áreas de texto ignoradas por el programa y se usan para grandes descripciones de código o comentarios que ayudan a otras personas a entender partes del programa. Empiezan con /* y terminan con */ y pueden abarcar múltiples lineas. Los comentarios son ignorados por el programa y no ocupan espacio en memoria.

```
/* Parpadeo de un led con intervalos de un segundo */
```

Comentarios de línea //

Comentarios de una línea empiezan con // y terminan con la siguiente línea de código. Como los bloques de comentarios son ignorados por el programa, no toman espacio en memoria. Comentarios de una línea se usan a menudo después de declaraciones válidas para proporcionar más información sobre qué lleva la declaración o proporcionar un recordatorio en el futuro.

```
int x = 13;                    //declara la variable “x” como el entero 13
```

2.5.3. Variables

Una variable es una forma de llamar y almacenar un valor numérico para usarse después por el programa. Como su nombre indica, las variables son números que pueden cambiarse continuamente al contrario que las constantes, cuyo valor nunca cambia. Una variable necesita ser declarada y, opcionalmente, asignada al valor que necesita para ser almacenada.

```
int valor = 0;                  // declara una variable y asigna el valor a 0
valor = analogRead(2);          // ajusta la variable al valor del pin analógico 2
```

Una vez que una variable ha sido asignada, o reasignada, puedes testear su valor para ver si cumple ciertas condiciones, o puede usarse directamente.

```
if(valor < 100)          // comprueba si la variable es menor que 100
{
valor = 100;             // si es cierto asigna el valor 100
}
delay(valor);            // usa la variable como retardo
```

Declaración de variable

Todas las variables tienen que ser declaradas antes de que puedan ser usadas. Declarar una variable significa definir su tipo de valor, como int, long, float, etc., definir un nombre específico, y, opcionalmente, asignar un valor inicial. Esto sólo necesita hacerse una vez en un programa pero el valor puede cambiarse en cualquier momento usando aritmética y varias asignaciones.

```
int valor = 0;                  // declara una variable y asigna el valor a 0
```

Una variable puede ser declarada en un número de posiciones en todo el programa y donde esta definición tiene lugar determina que partes del programa pueden usar la variable.

Ámbito de la variable

Una variable puede ser declarada al comienzo del programa antes del void **setup**(), localmente dentro de funciones, y algunas veces en un bloque de declaración, por ejemplo bucles for. Donde la variable es declarada determina el ámbito de la variable, o la habilidad de ciertas partes de un programa de hacer uso de la variable.

Una variable global es una que puede ser vista y usada por cualquier función y declaración en un programa. Esta variable se declara al comienzo del programa, antes de la función setup().

Una variable local es una que se define dentro de una función o como parte de un bucle for. Sólo

es visible y sólo puede ser usada dentro de la función en la cual fue declarada. Además, es posible tener dos o más variables del mismo nombre en diferentes partes del programa que contienen diferentes valores.

```
int valor;                          // 'valor' es visible por cualquier función

void setup() { }

void loop()
{
   for(int i=0; i<20;)              // 'i' es sólo visible dentro del bucle for
  {
     i++;
  }
  float f;                         // 'f' es sólo visible dentro de loop
}
```

2.5.3.1. Tipos de datos

byte

Byte almacena un valor numérico de 8 bits sin puntos decimales. Tienen un rango de 0 a 255.

```
byte x = 180;          // declara 'x' como un tipo byte y asigna el valor de 180
```

char

Es un tipo de dato que ocupa un byte de memoria y almacena un valor de caracter. Los caracteres literales se escriben con comillas simples: 'A' (para varios caracteres -strings[23]- se utiliza dobles comillas "ABC").

De todas maneras los caracteres son almacenados como números. Se puede ver su codificado en la tabla ASCII. Con esto se puede entender que es posible realizar cálculos aritméticos con los caracteres, en este caso se utiliza el valor ASCII del caracter (por ejemplo 'A' + 1 tiene el valor de 66, ya que el valor ASCII de la letra mayúscula A es 65)

El tipo de datos char tiene signo, esto significa que codifica números desde -128 hasta 127. Para un dato de un byte (8 bits), utiliza el tipo de dato "byte".

```
char x = 'A';

char y = 65;              // el valor de 'x' y 'y' equivalen a lo mismo
```

23 Strings: Cadena de caracteres.

int

Enteros son los tipos de datos primarios para almacenamiento de números sin puntos decimales y almacenan un valor de 16 bits con un rango de -32,768 a 32,767.

```
int x = 1000;                // declara 'x' como tipo int y asigna el valor de 1000
```

long

Tipo de datos de tamaño extendido para enteros largos, sin puntos decimales, almacenados en un valor de 32 bits con un rango de -2,146,483,648 a 2,147,483,647.

```
long x = 90000;              // declara 'x' como tipo long y asigna el valor de 90000
```

float

Un tipo de datos para números en punto flotante, o números que tienen un punto decimal. Los números en punto flotante tienen mayor resolución que los enteros y se almacenan como valor de 32 bits con un rango de -3.4028235E+38 a 3.4028235E+38.

```
float x = 3.14;              // declara 'x' como tipo float y asigna el valor de 3.14
```

arrays

Un array es una colección de valores que son accedidos con un índice numérico. Cualquier valor en el array debe llamarse escribiendo el nombre del

array y el índice numérico del valor. Los arrays están indexados a cero, con el primer valor en el array comenzando con el índice número 0. Un array necesita ser declarado y opcionalmente asignarle valores antes de que puedan ser usados.

```
int matriz[] = {value0, value1, value2...};
```

Asimismo es posible declarar un array declarando el tipo del array y el tamaño y luego asignarle valores a una posición del índice.

```
int matriz[5];          //declara un array de enteros con 6 posiciones
matriz3] = 10;          //asigna a la cuarta posición del índice el valor 10
```

Para recibir un valor desde un array, asignamos una variable al array y la posición del índice:

```
x = myArray[3];                 //x ahora es igual a 10
```

Constantes

El lenguaje Arduino tiene unos cuantos valores predefinidos que se llaman constantes. Se usan para hacer los programas más legibles. Las constantes se clasifican en grupos.

- TRUE / FALSE
- HIGH / LOW
- INPUT / OUTPUT

TRUE / FALSE

Estas son constantes booleanas que definen niveles lógicos. FALSE se define como 0 (cero) mientras TRUE es 1 o un valor distinto de 0.

```
if(b == TRUE)                  // si 'b' es verdadero
{
  digitalWrite(9, HIGH)        // envía un valor HIGH al pin 9;
}
```

HIGH / LOW

Estas constantes definen los niveles de pin como HIGH o LOW y se usan cuando se leen o se escriben los pines digitales. HIGH esta definido como el nivel 1 lógico, ON ó 5 V, mientras que LOW es el nivel lógico 0, OFF ó 0 V.

```
digitalWrite(13, HIGH);          // envía un valor HIGH al pin 13
```

INPUT / OUTPUT

Constantes usadas con la función pinMode() para definir el modo de un pin digital como INPUT u OUTPUT.

```
pinMode(13, OUTPUT);          // define el pin 13 como salida
```

Conversión de datos

Permite cambiar un valor dado a otro de un tipo de dato específico.

char(x), byte(x), int(x), long(x), float(x)

La variable “ x ” es el valor a convertir.

```
int i;              // declara 'i' como variable int
float f;            // declara 'f' como variable float
f = 3.14;           // asigna el valor de 3.14 a 'f'
i = int(f);         // convierte a 'f' en entero y se asigna a 'i', i es igual a 3
```

2.5.4. Operadores Aritméticos

Asignación " = "

El signo de igualdad "=" en el lenguaje de programación C se llama el operador de asignación. Tiene un significado diferente que en la clase de álgebra en el que se indica una ecuación o igualdad. El operador de asignación le dice al microcontrolador que evalúe cualquier valor o expresión en el lado derecho del signo igual, y lo almacene en la variable a la izquierda del signo igual.

```
int x = 1000;           // declara 'x' como tipo int y asigna el valor de 1000
```

Suma " + ", Resta " - ", Multiplicación " * " y División " / "

Estos operadores devuelven la suma, diferencia, producto o cociente (respectivamente) de los dos operandos. La operación se lleva a cabo utilizando el tipo de datos de los operandos, por lo que, por ejemplo, 9 / 4 resulta 2 desde 9 y 4 que son enteros int. Esto también significa que la operación puede desbordarse si el resultado es mayor que el que se puede almacenar en el tipo de datos (por ejemplo, la suma de 1 a un int con el valor de 32.767 resulta -32.768). Si los operandos son de tipos diferentes, se utiliza el tipo del "más grande" para el cálculo.

```
int x = 4;          // declara la variable 'x' y asigna el valor de 4
int y = 10;         // declara la variable 'y' y asigna el valor de 10
int i = 2;          // declara la variable 'i' y asigna el valor de 2
int j = 6;          // declara la variable 'j' y asigna el valor de 6
x = x + 2;          // suma 2 a 'x'
y = y -7;           // resta 7 a 'y'
i = i * 5;          // se multiplica 5 por 'i'
j = j / 2;          // se divide 2 a 'j'
```

Asignaciones Compuestas

Las asignaciones compuestas combinan una operación aritmética con una asignación de variable. Estas son muy frecuentemente encontradas en bucles for. Las asignaciones compuestas más comunes incluyen:

```
x++;        // lo mismo que x = x + 1
x--;        // lo mismo que x = x – 1
x += y      // lo mismo que x = x + y
x -= y      // lo mismo que x = x – y
x *= y      // lo mismo que x = x * y
x /= y      // lo mismo que x = x / y
```

Operadores Comparativos

Las comparaciones de una variable o constante con otra se usan a menudo en declaraciones if para comprobar si un condición específica es cierta.

```
x == y;     // x es igual a y
x != y;     // x no es igual a y
x < y;      // x es menor a y
x > y;      // x es mayor a y
x <= y;     // x es menor o igual a y
x >= y;     // x es mayor o igual a y
```

Operadores Booleanos

Los operadores booleanos o lógicos son normalmente una forma de comparar dos expresiones y devuelven TRUE o FALSE dependiendo del operador. Hay

tres operadores lógicos, AND, OR y NOT, que se usan a menudo en declaraciones if.

AND lógico (&&):

```
if (x>0 && x<5)          //verdadero sólo si las dos expresiones son ciertas
```

OR lógico (||):

```
if (x>0 || y>0)          //verdadero si al menos una expresión es cierta
```

NOT lógico (!):

```
if (!(x>0))              //verdadero sólo si la expresión es falsa
```

2.5.5. Control de Flujo

if

Las sentencias if comprueban si cierta condición ha sido alcanzada y ejecutan todas las sentencias dentro de las llaves si la declaración es cierta. Si es falsa el programa ignora la sentencia.

```
if (x < 2)          // si 'x' es menor a 2, realizar la siguiente acción
{
  x++;              // suma el valor de 1 a 'x'
}
```

if - else

if - else permite tomar decisiones entre una opción u otra.

```
if (pin == HIGH)                // si 'pin' es HIGH, realizar la siguiente acción
{
  digitalWrite(9, HIGH)         // envía un valor HIGH al pin 9;
}
else                            // caso contrario, realizar la siguiente acción
{
  digitalWrite(9, LOW)          // envía un valor LOW al pin 9;
}
```

else puede preceder a otra comprobación if, por lo que múltiples y mutuas comprobaciones exclusivas pueden ejecutarse al mismo tiempo.

```
if (x < 2)                      // si 'x' es menor a 2, realizar la siguiente acción
{
  digitalWrite(9, HIGH)         // envía un valor HIGH al pin 9;
}
else if (x >= 5)                // en el caso que 'x' sea mayor o igual a 5, haga
{
  digitalWrite(10, HIGH)        // envía un valor HIGH al pin 10;
}
else                            // caso contrario, realizar la siguiente acción
{
  digitalWrite(9, LOW)          // envía un valor LOW al pin 9;
  digitalWrite(10, LOW)         // envía un valor LOW al pin 10;
}
```

for

La sentencia for se usa para repetir un bloque de declaraciones encerradas en llaves un número específico de veces. Un contador de incremento se usa a menudo para incrementar y terminar el bucle. Hay tres partes separadas por punto y coma (;), en la cabecera del bucle.

```
for (inicialización; condición; expresión)
{
  // realizar esta acción
}
```

La inicialización de una variable local, o contador de incremento, sucede primero y una sola una vez. Cada vez que pasa el bucle, la condición siguiente es comprobada. Si la condición devuelve TRUE, las declaraciones y expresiones que siguen se ejecutan y la condición se comprueba de nuevo. Cuando la condición se vuelve FALSE, el bucle termina.

```
for (int i = 0; i < 20; i++)        // declara i, comprueba si es menor
{                                   // que 20, incrementa i en 1
  digitalWrite(13, HIGH);           // activa el pin 13
  delay(250);                       // pausa por 250 milisegundos
  digitalWrite(13, LOW);            // desactiva el pin 13
  delay(250);                       // pausa por 250 milisegundos
}
```

while

El bucle while se repetirá continuamente, e infinitamente, hasta que la expresión dentro del paréntesis se vuelva falsa. Algo debe cambiar la variable comprobada o el bucle while nunca saldrá. Lo que modifique la variable puede estar en el código, como una variable que se incrementa, o ser una condición externa, como el valor que da un sensor.

```
while (x < 1000)                    // comprueba si es menor que 1000
{
  digitalWrite(9, HIGH);            // activa el pin 9
  delay(x);                         // espera el tiempo que de la variable x
  digitalWrite(9, LOW);// desactiva el pin 9
  x++;                              // incrementa la variable en 1
}
```

do. . . while

El bucle do. . . while es un bucle que trabaja de la misma forma que el bucle while, con la excepción de que la condición es testeada al final del bucle, por lo que el bucle do. . . while siempre se ejecutará al menos una vez.

```
do
{
x = analogRead(A0);          // asigna el valor de analogRead(A0) a 'x'
delay(50);                   // pausa de 50 milisegundos
}
while(x < 100);              // repite si 'x' es menor que 100
```

break

break es usado para salir de los bucles do, for, o while, pasando por alto la condición normal del bucle.

```
for (int i = 0; i < 20; i++)      // declara 'i', comprueba si es menor
{                                 // que 20, incrementa 'i' en 1
  digitalWrite(13, HIGH);         // activa el pin 13
  delay(250);                     // pausa por 250 milisegundos
  digitalWrite(13, LOW);          // desactiva el pin 13
  delay(250);                     // pausa por 250 milisegundos
  if (i == 10)                    // si 'i' = 10
  {
    i = 0;                        // asigna a 'i' = 0
    break;                        // sale del bucle for
  }
}
```

2.5.6. Funciones

2.5.6.1. E/S Digitales

pinMode(pin, modo)

Configura el pin especificado para que funcione como entrada o salida, este comando se lo especifica dentro del void **setup**().

pin = el número de pin que se quiere configurar.

modo = entrada INPUT, salida OUTPUT.

```
void setup()
{
  pinMode (9, OUTPUT);              // configura el pin 9 como salida
}
```

Una entrada analógica puede ser usada como un pin digital, refiriéndose a ellos desde el número 14 (entrada analógica 0) a 19 (entrada analógica 5).

digitalRead(pin)

Lee el valor desde un pin digital especificado con el resultado HIGH o LOW.

```
x = digitalRead(pin);          // ajusta 'x' igual al pin de entrada
```

digitalWrite(pin, valor)

Devuelve el nivel lógico HIGH o LOW (activa o desactiva) a un pin digital especificado.

```
digitalWrite(pin, HIGH);       // envía un valor HIGH al pin de salida
```

```
int led = 13;                  // LED conectado al pin digital número 13
int boton = 7;                 // boton conectado al pin digital número 7
int x = 0;                     // variable donde se almacena el valor leído

void setup() {
  pinMode(led, OUTPUT);        // establece el pin digital 13 como salida
  pinMode(boton, INPUT);       // establece el pin digital 7 como entrada
}
void loop() {
  x = digitalRead(boton);      // leer el pin de entrada
  digitalWrite(led, x);        // establece el LED al valor del botón
}
```

2.5.6.2. E/S Analógicas

analogRead(pin)

Lee el valor desde un pin analógico especificado con una resolución de 10 bits. Esta función sólo trabaja en los pines analógicos (0 - 5). Los valores enteros devueltos están en el rango de 0 a 1023.

```
x = analogRead(pin);   // ajusta 'x' igual al valor leído del 'pin'
```

analogWrite(pin, valor)

Escribe un valor pseudo analógico usando PWM[24] a un pin de salida marcado como PWM. En los Arduinos más nuevos con el chip ATmega328, esta función trabaja en los pines 3, 5, 6, 9, 10 y 11. Los Arduinos más antiguos con un ATmega8 sólo soporta los pines 9, 10 y 11. El valor puede ser especificado como una variable o constante con un valor de 0 a 255.

```
analogWrite(pin, valor);          // escribe el 'valor' en el 'pin' analógico
```

Valor	Nivel de Salida
0	0 V (t)
64	0 V (3/4 de t) y 5 V (1/4 de t)
128	0 V (1/2 de t) y 5 V (1/2 de t)
192	0 V (1/4 de t) y 5 v (3/4 de t)
255	5 V (t)

Tabla 2.2. Relación valor-salida con analogWrite()

El valor de salida varía de 0 a 5 V según el valor de entrada (de 0 a 255) en función del tiempo de pulso. Si t es el tiempo de pulso, la tabla 2.2. muestra la equivalencia entre el valor y la salida en función del tiempo.

Como esta es una función hardware, el pin generará una onda estática después de una llamada a analogWrite en segundo plano hasta la siguiente

24 PWM. Pulse width modulation. Modulación por ancho de pulso

llamada a analogWrite (o una llamada a digitalRead o digitalWrite en el mismo pin).

2.5.6.3. E/S Avanzadas

tone() / noTone()

Genera una onda cuadrada de la frecuencia especificada (y un 50% de ciclo de trabajo) en un pin. La duración puede ser especificada, en caso contrario la onda continua hasta que haya una llamada a noTone(). El pin puede conectarse a un zumbador piezoeléctrico u otro altavoz que haga sonar los tonos.

Sólo puede generarse un tono cada vez. Si un tono está sonando en un pin diferente, la llamada a tone() no tendrá efecto. Si el tono está sonando en el mismo pin, la llamada establecerá la nueva frecuencia.

Si se quiere hacer sonar diferentes tonos en múltiples pines se necesita llamar a noTone() en un pin antes de llamar a tone() en el siguiente pin.

tone(pin, frecuencia)

tone(pin, frecuencia, duración)

Donde:

pin: el pin en el que se va a generar el tono.

frecuencia: la frecuencia del tono en hercios.

duración: la duración del tono en milisegundos (opcional)

Las frecuencias audibles por el oído humano van aproximadamente de los 20Hz a los 20KHz por lo que el parámetro "frecuencia" debería estar comprendido entre estos dos valores.

2.5.6.4. Tiempo

delay(ms)

Pausa el programa por la cantidad de tiempo especificada en milisegundos, donde 1000 es igual a 1 segundo.

```
delay(1000);              // espera por un segundo
```

millis()

Devuelve el número de milisegundos desde que la placa Arduino empezó a ejecutar el programa actual como un valor long sin signo.

```
valor = millis();         // ajusta 'valor' igual a millis()
```

Nota: Este número se desbordará (resetear de nuevo a cero), después de aproximadamente 9 horas.

2.5.6.5. Matemáticas

min(x,y)

Calcula el mínimo de dos números de cualquier tipo de datos y devuelve el número más pequeño.

```
x = min(x, 100);          // asigna a 'x' al más pequeño de 'x' o 100
```

max(x,y)

Calcula el máximo de dos números de cualquier tipo de datos y devuelve el número más grande.

```
val = max(val, 100);      // asigna a 'val' al más grande de 'val' o 100
```

2.5.6.6. Aleatorio

random(max) / random(min, max)

La función random permite devolver números pseudo aleatorios en un rango especificado por los valores min y max.

```
v = random(100, 200);          // asigna a 'v' un número aleatorio entre 100 y 200
```

2.5.6.7. Serial

Serial.begin(rate)

Abre el puerto serie y asigna la tasa de baudios para la transmisión de datos serie. La típica tasa de baudios para comunicarse con el ordenador es 9600 aunque otras velocidades están soportadas.

```
void setup()
{
Serial.begin(9600);        // abre el puerto serie, ajusta la tasa de datos a 9600 bps
}
```

Cuando se usa la comunicación serie, los pines digitales 0 (Rx) y 1 (Tx) no pueden ser usados al mismo tiempo.

Serial.println(data)

Imprime datos al puerto serie, seguido de un retorno de carro y avance de línea automáticos.

Este comando toma la misma forma que Serial.print(), pero es más fácil para leer datos en el Serial Monitor.

```
Serial.println(analogRead(A0));        // envía el valor analógico leído del pin 0
```

Serial.available()

Devuelve el número de bytes (caracteres) disponibles para ser leídos por el puerto serie. Se refiere a datos ya recibidos y disponibles en el buffer de recepción del puerto (que tiene una capacidad de 128 bytes).

Serial.read()

Lee los datos ingresados por el puerto serie.

```
int datorecibido = 0;                    // para el byte leído

void setup()
{
  Serial.begin(9600);                    // abre el puerto serie a 9600 bps
}
void loop()
{
  if (Serial.available() > 0)            // envía datos solamente cuando recibe
datos
  {
    datorecibido = Serial.read();        // lee el byte entrante:
    Serial.print("He recibido: ");       // dice lo que ha recibido:
    Serial.println(datorecibido, DEC);
  }
}
```

2.5.7. Librerías Arduino

Las librerías proporcionan funcionalidad extra para la utilización en "sketchs", por ejemplo para trabajar con hardware o manipular datos. Para utilizar una librería en un "sketch", se selecciona el menú Sketch > Import Library. Esto insertará una o más sentencias **#include** al principio del programa y compilará la librería con su "sketch". Debido a que las librerías se vuelcan a la placa junto con su "sketch", incrementan la ocupación del espacio disponible. Si un

"sketch" no precisa de una librería, simplemente borra su sentencia **#include** en la parte inicial de su código.

Las siguientes son las librerías estándar:

- EEPROM: Para leer y escribir en memorias permanentes.
- Ethernet: Para conectar por red a través de la Ethernet Shield.
- Firmata: Para comunicarse con aplicaciones a la computadora usando un protocolo estándar serial.
- LiquidCristal: Para controlar display de cristal líquido LCD.
- Servo: Para controlar servomotores.
- SoftwareSerial: Para la comunicación serial de cualquier pin digital.
- Stepper: Para controlar motores paso a paso.
- Wire: Interfaz de dos cables, ó *Two Wire Interface (TWI/I2C)*, para enviar y recibir datos a través de una red de dispositivos y sensores.

Existen librerías no oficiales de Arduino, que se las conoce como librerías contribuidas. Para instalar una librería contribuida, se descarga del proveedor, se descomprime y se instala en la subcarpeta libraries ubicada en el sketchbook, la subcarpeta debe tener 2 archivos con las extensiones “.h” y “.cpp”.

Para utilizar la librería tan solo se agrega la cabecera al inicio del sketch.

```
/* Está es la librería para una matriz de led de 8x8 manejada con el MAX7219 o MAX7221 */

#include "LedControl.h"
```

La cabecera llama a la librería en tan solo una línea de programación, pero Arduino compila una amplia información de los archivos guardados en las subcarpetas de las librerías.

El sketch de la librería #include "LedControl.h", tiene 2 extensiones como se mencionó anteriormente y son las siguientes:

- *LedControl.h*
- *LedControl.cpp*

Entre las librerías contribuidas se tiene:

Comunicación (networking y protocolos):

- Messenger: Para procesar mensajes de texto desde la computadora.
- NewSoftSerial: Versión mejorada de la librería SoftwareSerial.
- OneWire: Controla dispositivos (de Dallas Semiconductor) que usan el protocolo One Wire.
- PS2Keyboard: Lee caracteres de un teclado PS2.
- Simple Message System: Envía mensajes entre Arduino y la computadora.
- SSerial2Mobile: Envía mensajes de texto o emails usando un teléfono móvil (vía comandos AT a través de software serial)
- Webduino: Librería de web server extendible (para usar con Arduino Ethernet Shield)
- X10: Para enviar señales X10 a través de líneas de corriente AC.
- Xbee: Para comunicaciones entre Xbee en modo API.

- SerialControl: Para controlar remotamente otras Arduino a través de una conexión serial.

Sensores:

- Capacitive Sensing: Convertir dos o más pins en sensores capacitivos.
- Debounce: Para lectura de inputs digitales con ruido (por ejemplo, botones).

Displays y LEDs:

- Improved LCD library: Arregla bugs de inicialización de LCD de la librería LCD oficial de Arduino.
- GLCD: Grafica rutinas para LCDs basados en el chipset KS0108 ó equivalentes.
- LedControl: Para controlar matrices de LEDs o displays de siete segmentos con MAX7221 ó MAX7219.
- LedControl: Alternativa a la librería Matrix para controlar múltiples LEDs con chips Maxim.
- LedDisplay: Control para marquesina de LED HCMS-29xx.

Generación de Frecuencias y Audio:

- Tone: Genera frecuencias de audio de onda cuadrada en el background de cualquier pin de un microcontrolador.

Motores y PWM:

- TLC5940: Controlador de PWM de 16 canales y 12 bits.

Medición de Tiempo:

- DateTime: Librería para llevar registro de fecha y hora actual en el software.
- Metro: Útil para cronometrar acciones en intervalos regulares.
- MsTimer2: Utiliza timer 2 interrupt para disparar una acción cada N milisegundos.

Utilidades:

- TextString: También conocido como String - Maneja strings.
- Pstring: Liviana clase para imprimir en búfer.
- Streaming: Método para simplificar declaraciones de impresión.

2.6. Componentes Electrónicos

2.6.1. Pantalla LCD

Una LCD[25] es una pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora. A menudo se utiliza en dispositivos electrónicos de pilas, ya que utiliza cantidades muy pequeñas de energía eléctrica.

La LCD más utilizada en elementos electrónicos es la LCD 16x2 y se trata de un módulo microcontrolado capaz de representar 2 líneas de 16 caracteres cada una. A través de 8 líneas de datos se le envía el carácter ASCII que se desea visualizar así como ciertos códigos de control que permiten realizar

25 LCD: Liquid Cristal Display, Pantalla de Cristal Líquido.

diferentes efectos de visualización. Igualmente mediante estas líneas de datos el módulo devuelve información de su estado interno.

Con otras tres señales adicionales se controla el flujo de información entre el módulo LCD y el equipo informática que lo gestiona.

En la figura 2.7 se observa una pantalla LCD convencional de 16x2.

Figura 2.7. Pantalla LCD 16x2

2.6.2. Motores Eléctricos

Un motor eléctrico es una máquina eléctrica que transforma energía eléctrica en energía mecánica por medio de interacciones electromagnéticas. Algunos de los motores eléctricos son reversibles, pueden transformar energía mecánica en energía eléctrica funcionando como generadores. Los motores eléctricos de tracción usados en locomotoras realizan a menudo ambas tareas, si se los equipa con frenos regenerativos.

Son ampliamente utilizados en instalaciones industriales, comerciales y particulares. Pueden funcionar conectados a una red de suministro eléctrico o a

baterías. Así, en automóviles se están empezando a utilizar en vehículos híbridos para aprovechar las ventajas de ambos.

Los motores de corriente alterna y los de corriente continua se basan en el mismo principio de funcionamiento, el cual establece que si un conductor por el que circula una corriente eléctrica se encuentra dentro de la acción de un campo magnético, éste tiende a desplazarse perpendicularmente a las líneas de acción del campo magnético.

El conductor tiende a funcionar como un electroimán debido a la corriente eléctrica que circula por el mismo adquiriendo de esta manera propiedades magnéticas, que provocan, debido a la interacción con los polos ubicados en el estátor, el movimiento circular que se observa en el rotor del motor.

Partiendo del hecho de que cuando pasa corriente por un conductor produce un campo magnético, además si lo ponemos dentro de la acción de un campo magnético potente, el producto de la interacción de ambos campos magnéticos hace que el conductor tienda a desplazarse produciendo así la energía mecánica. Dicha energía es comunicada al exterior mediante un dispositivo llamado flecha.

Los motores de corriente continua utilizados en electrónica son:

- Motores PAP[26]
- Servomotores
- Motores sin núcleo

26 Motores PAP: Motores paso a paso.

2.6.2.1. Motores PAP

Los motores paso a paso son ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos.

La característica principal de estos motores es el hecho de poder moverlos un paso a la vez por cada pulso que se le aplique. Este paso puede variar desde 90° hasta pequeños movimientos de tan solo 1.8°, es decir, que se necesitarán 4 pasos en el primer caso (90°) y 200 para el segundo caso (1.8°), para completar un giro completo de 360°.

Estos motores poseen la habilidad de poder quedar enclavados en una posición o bien totalmente libres. Si una o más de sus bobinas está energizada, el motor estará enclavado en la posición correspondiente y por el contrario quedará completamente libre si no circula corriente por ninguna de sus bobinas.

Básicamente estos motores están constituidos normalmente por un rotor sobre el que van aplicados distintos imanes permanentes y por un cierto número de bobinas excitadoras bobinadas en su estator.

Las bobinas son parte del estator y el rotor es un imán permanente. Toda la conmutación (o excitación de las bobinas) deber ser externamente manejada por un controlador.

Los Motores PAP suelen ser clasificado en dos tipos, según su diseño y fabricación pueden ser Bipolares o Unipolares, se los puede observar en la figura 2.8.

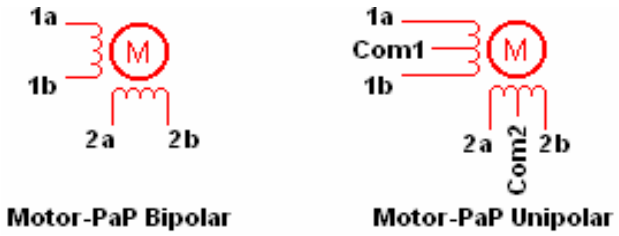


Figura 2.8. Motores PAP Bipolar y Unipolar

Motor PAP Bipolar

Estos tiene generalmente cuatro cables de salida. Necesitan ciertos trucos para ser controlados, debido a que requieren del cambio de dirección del flujo de corriente a través de las bobinas en la secuencia apropiada para realizar un movimiento.

Estos motores necesitan la inversión de la corriente que circula en sus bobinas en una secuencia determinada. Cada inversión de la polaridad provoca el movimiento del eje en un paso, cuyo sentido de giro está determinado por la secuencia seguida.

En la tabla 2.3 se puede ver la secuencia necesaria para controlar motores PAP del tipo bipolares:

PASO	TERMINALES			
	A	B	C	D
1	+V	-V	+V	-V
2	+V	-V	-V	+V
3	-V	+V	-V	+V
4	-V	+V	+V	-V

Tabla 2.3. Secuencia para controlar los motores PAP bipolares

Motores PAP Unipolares

Estos motores suelen tener 5 ó 6 cables de salida dependiendo de su conexión interna. Este tipo se caracteriza por ser más simple de controlar, estos utilizan un cable común a la fuente de alimentación y posteriormente se van colocando las otras lineas a tierra en un orden especifico para generar cada paso, si tienen 6 cables es porque cada par de bobinas tiene un común separado, si tiene 5 cables es porque las cuatro bobinas tiene un solo común; un motor unipolar de 6 cables puede ser usado como un motor bipolar si se deja las lineas del común al aire.

Existen 3 maneras o tipos de movimiento de los motores unipolares, que son los siguientes:

Secuencia Normal: Esta es la secuencia más usada y la que generalmente recomienda el fabricante. Con esta secuencia el motor avanza un paso por vez y debido a que siempre hay al menos dos bobinas activadas, se obtiene un alto torque de paso y de retención.

Secuencia del tipo wave drive: En esta secuencia se activa solo una bobina a la vez. En algunos motores esto brinda un funcionamiento mas suave. La contrapartida es que al estar solo una bobina activada, el torque de paso y retención es menor.

Secuencia del tipo medio paso: En esta secuencia se activan las bobinas de tal forma de brindar un movimiento igual a la mitad del paso real. Para ello se activan primero 2 bobinas y luego solo 1 y así sucesivamente.

2.6.2.2. Servomotores

Un servomotor (también llamado **servo**) es un dispositivo similar a un motor de corriente continua que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y mantenerse estable en dicha posición.

Un servomotor es un motor eléctrico que consta con la capacidad de ser controlado, tanto en velocidad como en posición.

Los servos se utilizan frecuentemente en sistemas de radio control y en robótica, pero su uso no está limitado a estos. Es posible modificar un servomotor para obtener un motor de corriente continua que, si bien ya no tiene la capacidad de control del servo, conserva la fuerza, velocidad y baja inercia que caracteriza a estos dispositivos.

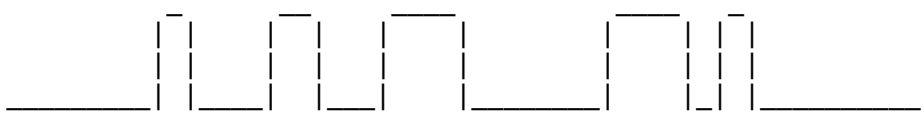
Está conformado por un motor, una caja reductora y un circuito de control. También potencia proporcional para cargas mecánicas. Un servo, por consiguiente, tiene un consumo de energía reducido.

La corriente que requiere depende del tamaño del servo. Normalmente el fabricante indica cual es la corriente que consume. La corriente depende principalmente del par, y puede exceder un amperio si el servo está enclavado, pero no es muy alto si el servo está libre moviéndose todo el tiempo.

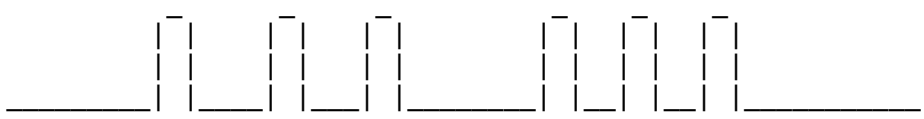
Los servomotores hacen uso de la modulación por ancho de pulsos (PWM) para controlar la dirección o posición de los motores de corriente continua. La mayoría trabaja en la frecuencia de los cincuenta hercios, así las señales PWM tendrán un periodo de veinte milisegundos. La electrónica dentro del servomotor responderá al ancho de la señal modulada. Si los circuitos dentro

del servomotor reciben una señal de entre 0,5 a 1,4 milisegundos, este se moverá en sentido horario; entre 1,6 a 2 milisegundos moverá el servomotor en sentido antihorario; 1,5 milisegundos representa un estado neutro para los servomotores estándares. A continuación se exponen ejemplos de cada caso:

Señal de ancho de pulso modulado:



Motor en sentido horario (ejemplo 0,7 ms):



Motor neutral (1,5ms):

Motor en sentido antihorario (ejemplo 1,8ms):

2.6.3. LED

El LED[27], es un dispositivo semiconductor que emite luz incoherente de espectro reducido cuando se polariza de forma directa la unión PN en la cual circula por él una corriente eléctrica . Este fenómeno es una forma de electroluminiscencia, el LED es un tipo especial de diodo que trabaja como un diodo común, pero que al ser atravesado por la corriente eléctrica, emite luz . Este dispositivo semiconductor está comúnmente encapsulado en una cubierta de plástico de mayor resistencia que las de vidrio que usualmente se emplean en las lámparas incandescentes. Aunque el plástico puede estar coloreado, es sólo por razones estéticas, ya que ello no influye en el color de la luz emitida. Usualmente un LED es una fuente de luz compuesta con diferentes partes, razón por la cual el patrón de intensidad de la luz emitida puede ser bastante complejo.

Para obtener una buena intensidad luminosa debe escogerse bien la corriente que atraviesa el LED y evitar que este se pueda dañar; para ello, hay que tener en cuenta que el voltaje de operación va desde 1,8 hasta 3,8 voltios aproximadamente (lo que está relacionado con el material de fabricación y el color de la luz que emite) y la gama de intensidades que debe circular por él varía según su aplicación. Los valores típicos de corriente directa de polarización de un LED están comprendidos entre los 10 y 20 miliamperios (mA) en los diodos de color rojo y de entre los 20 y 40 miliamperios (mA) para los otros LED. Los diodos LED tienen enormes ventajas sobre las lámparas

27 LED: Light-Emitting Diode, Diodo Emisor de Luz.

indicadoras comunes, como su bajo consumo de energía, su mantenimiento casi nulo y con una vida aproximada de 100,000 horas. Para la protección del LED en caso haya picos inesperados que puedan dañarlo. Se coloca en paralelo y en sentido opuesto un diodo de silicio común

En general, los LED suelen tener mejor eficiencia cuanto menor es la corriente que circula por ellos, con lo cual, en su operación de forma optimizada, se suele buscar un compromiso entre la intensidad luminosa que producen (mayor cuanto más grande es la intensidad que circula por ellos) y la eficiencia (mayor cuanto menor es la intensidad que circula por ellos).

El LED al ser un diodo posee las mismas terminales de estos, es decir ánodo y cátodo. En la figura 2.9. se observa el diagrama y pines de un LED.

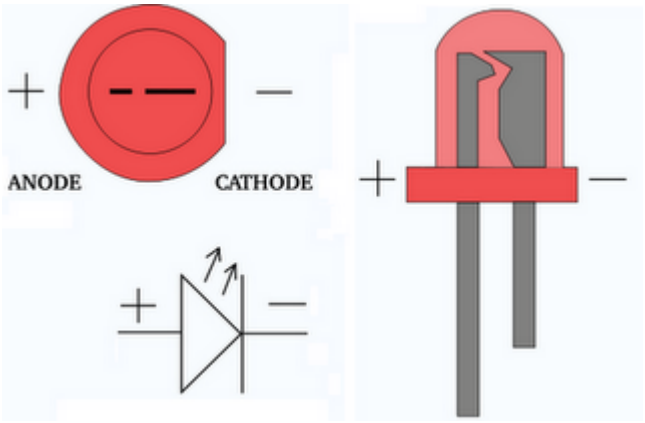


Figura 2.9. Diagrama esquemático del LED

LED RGB[28]

Los LED RGB son diodos emisores de luz que emiten diferentes tipos de colores a través del mismo encapsulado. Existen 2 tipos de RGB, el uno de 2 pines que tiene un ánodo y un cátodo al igual que un LED convencional, que

28 RGB: Red Green Blue, Rojo Verde Azul.

varía el color de acuerdo al voltaje aplicado. Y el de 4 pines que es el más usado por dar la posibilidad de mostrar una variedad amplia de colores al aplicar voltaje en uno o mas pines a la vez; posee 3 ánodos y un cátodo común. En la figura 2.10 se observa un LED RGB de 4 pines.

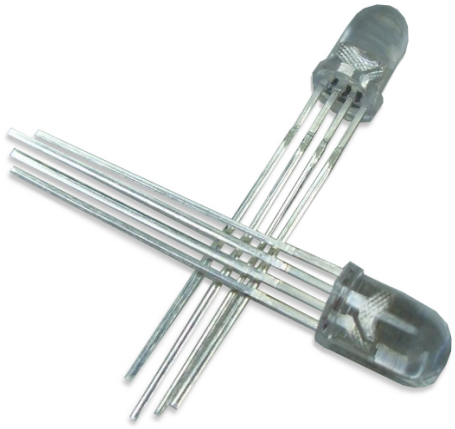


Figura 2.10. LED RGB de 4 pines

2.6.4. Matriz de LED

Una matriz de LED no es más que un grupo de LEDs unidos entre sus cátodos en forma de filas y todos sus ánodos en forma de columnas.

Se enciende un LED de la matriz enviando un 1 lógico (5 V) a la fila correspondiente y un 0 lógico (GND) a la columna donde se encuentre el LED a encenderse.

Las matrices conservan las características técnicas de los LEDs convencionales, es por eso que se tienen matrices unicolor, bicolores o matrices RGB.

En al figura 2.11 se observa la distribución de filas y columnas de una matriz de 8x8.

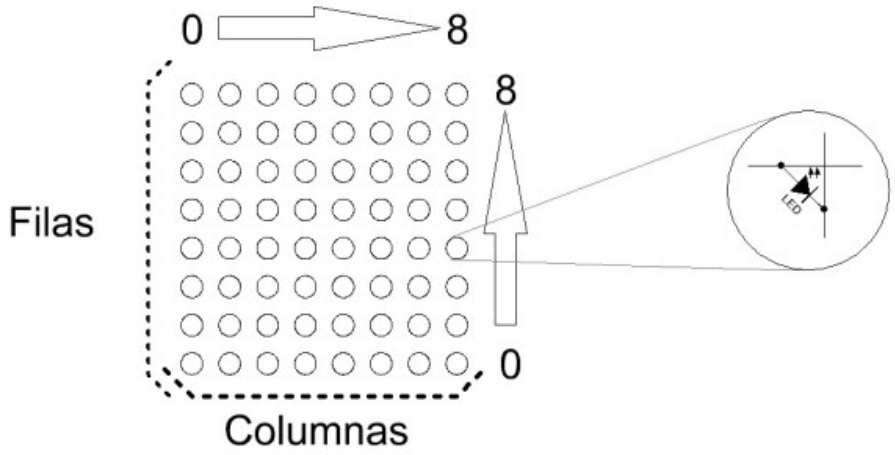


Figura 2.11. Filas y columnas de una matriz de LED 8x8

2.6.5. Sensores

Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, pH, etc. Una magnitud eléctrica puede ser una resistencia eléctrica (como en una RTD[29]), una capacidad eléctrica (como en un sensor de humedad), una Tensión eléctrica (como en un termopar), una corriente eléctrica (como en un foto transistor), etc.

Un sensor diferencia de un transductor en que el sensor está siempre en contacto con la variable de instrumentación con lo que puede decirse también que es un dispositivo que aprovecha una de sus propiedades con el fin de

29 RTD: Resistence temperature detector, Detector resistivo de temperatura

adaptar la señal que mide para que la pueda interpretar otro dispositivo. Como por ejemplo el termómetro de mercurio que aprovecha la propiedad que posee el mercurio de dilatarse o contraerse por la acción de la temperatura. Un sensor también puede decirse que es un dispositivo que convierte una forma de energía en otra.

Características de un sensor

- Rango de medida: dominio en la magnitud medida en el que puede aplicarse el sensor.
- Precisión: es el error de medida máximo esperado.
- *Offset* o desviación de cero: valor de la variable de salida cuando la variable de entrada es nula. Si el rango de medida no llega a valores nulos de la variable de entrada, habitualmente se establece otro punto de referencia para definir el *offset*.
- Linealidad o correlación lineal.
- Sensibilidad de un sensor: suponiendo que es de entrada a salida y la variación de la magnitud de entrada.
- Resolución: mínima variación de la magnitud de entrada que puede apreciarse a la salida.
- Rapidez de respuesta: puede ser un tiempo fijo o depender de cuánto varíe la magnitud a medir. Depende de la capacidad del sistema para seguir las variaciones de la magnitud de entrada.

- Derivas: son otras magnitudes, aparte de la medida como magnitud de entrada, que influyen en la variable de salida. Por ejemplo, pueden ser condiciones ambientales, como la humedad, la temperatura u otras como el envejecimiento (oxidación, desgaste, etc.) del sensor.
- Repetibilidad: error esperado al repetir varias veces la misma medida.

Un sensor es un tipo de transductor que transforma la magnitud que se quiere medir o controlar, en otra, que facilita su medida. Pueden ser de indicación directa (e.g. un termómetro de mercurio) o pueden estar conectados a un indicador (posiblemente a través de un convertidor analógico a digital, un computador y un display) de modo que los valores detectados puedan ser leídos por un humano.

Por lo general, la señal de salida de estos sensores no es apta para su lectura directa y a veces tampoco para su procesado, por lo que se usa un circuito de acondicionamiento, como por ejemplo un puente de Wheatstone, amplificadores y filtros electrónicos que adaptan la señal a los niveles apropiados para el resto de los circuitos.

2.6.5.1. Sensores de temperatura

Los sensores de temperatura se dividen en 3 grupos de acuerdo a sus características:

- Termopares
- Termistores
- Detectores Resistivos de Temperatura (RTD)

Termopares

El dispositivo más común para la medición de temperaturas de procesos industriales es el termopar. Un termopar es un par de alambres de metales diferentes unidos en una malla completa. Los alambres distintos tienen dos puntos de unión, uno en cada extremo de la malla. Una unión, llamada unión caliente, es sujeta a una alta temperatura. La otra unión, llamada unión fría, es sujeta a baja temperatura. Al hacer esto, se crea un pequeño voltaje neto en la malla. Este voltaje es proporcional a la diferencia entres las dos temperaturas de las uniones.

Lo que ocurre en una malla de termopar es que se produce un pequeño voltaje en cada unión de los metales distintos, debido a un confuso fenómeno llamado *efecto Seebeck*[30]. Entre mayor sea la temperatura en la unión, mayor será el voltaje producido por esa unión. Es más, la relación entre voltaje y la temperatura es aproximadamente lineal.

Termistores y Detectores resistivos de temperatura (RTD)

Además de usar el voltaje en un termopar para medir eléctricamente la temperatura, también es posible usar el cambio de resistencia que ocurre en muchos materiales a medida que cambia su temperatura. Los materiales usados para este propósito caen en dos categorías, los metales puros y los metales oxidados.

30 El efecto Seebeck es una propiedad termoeléctrica descubierta en 1821 por el físico alemán Thomas Johann Seebeck. Este efecto provoca la conversión de una diferencia de temperatura en electricidad.

Los metales puros tienen un coeficiente de resistencia de temperatura positivo bastante constante. El coeficiente de resistencia de temperatura, generalmente llamado coeficiente de temperatura es la razón de cambio de resistencia al cambio de temperatura. Un coeficiente positivo significa que la resistencia aumenta a medida que aumenta la temperatura. Si el coeficiente es constante, significa que el factor de proporcionalidad entre la resistencia y la temperatura es constante y que la resistencia y la temperatura se graficarán en una línea recta. Cuando se usa un alambre de metal puro para la medición de temperatura, se le refiere como ***detector resistivo de temperatura,*** o ***RTD.***

Cuando se usan óxidos metálicos para la medición de temperatura, el material de óxido metálico es conformado en formas que semejan pequeños bulbos o pequeños capacitores. El dispositivo formado así se llama ***termistor***. Los termistores tienen coeficiente de temperatura negativos grandes que no son constantes. En otras palabras, el cambio de resistencia por unidad de cambio de temperatura es mucho mayor que para un metal puro, pero el cambio es en otra dirección: la resistencia disminuye a medida que aumenta la temperatura.

2.6.5.2. Sensores Infrarrojos

El sensor infrarrojo es un dispositivo electrónico capaz de medir la radiación electromagnética infrarroja de los cuerpos en su campo de visión. Todos los cuerpos reflejan una cierta cantidad de radiación, esta resulta invisible para los ojos pero no para estos aparatos electrónicos, ya que se encuentran en el rango del espectro justo por debajo de la luz visible.

Los rayos infrarrojos(IR) entran dentro del foto transistor donde encontramos un material piroeléctrico, natural o artificial, normalmente formando una lámina delgada dentro del nitrato de galio (GaN), nitrato de Cesio (CsNO3), derivados de la fenilpirazina, y ftalocianina de cobalto. Normalmente están integrados en diversas configuraciones(1,2,4 píxeles[31] de material piroeléctrico). En el caso de parejas se acostumbra a dar polaridades opuestas para trabajar con un amplificador diferencial, provocando la auto-cancelación de los incrementos de energía de IR y el desacoplamiento del equipo.

Sensores pasivos

Están formados únicamente por el fototransistor con el cometido de medir las radiaciones provenientes de los objetos.

Sensores activos

Se basan en la combinación de un emisor y un receptor próximos entre ellos, normalmente forman parte de un mismo circuito integrado. El emisor es un diodo LED infrarrojo (IR) y el componente receptor el fototransistor.

En la figura 2.12 se observa el diagrama esquemática del IR activo, conformado por el emisor y el receptor; y en la figura 2.13 se tiene el encapsulado típico del IR activo.

31 Un **píxel** o **pixel**, plural **píxeles** (acrónimo del inglés *picture element*, "elemento de imagen") es la menor unidad homogénea en color que forma parte de una imagen digital, ya sea esta una fotografía, un fotograma de vídeo o un gráfico.

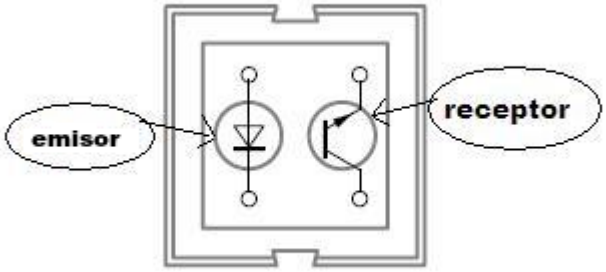


Figura 2.12. Diagrama esquemático del IR activo

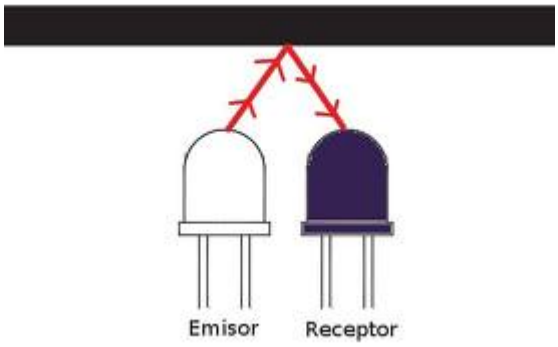


Figura 2.13. Encapsulado del IR activo

2.6.6. Comunicación Serial RS-232

La primera razón por la cual la transmisión paralela de datos no se usa exclusivamente es el rango limitado de distancia sobre el cual es posible transmitir datos en un bus paralelo. Aunque hay técnicas que permiten extender el rango de la transmisión paralela, estas son complejas y costosas. En consecuencia, la transmisión serial se usa frecuentemente, cuando los datos se deben transmitir a una distancia significativa. Puesto que los datos seriales viajan a lo largo de un solo camino y se transmiten un bit a la vez, el costo del

cable para una larga distancia es relativamente bajo; además, las unidades de transmisión y recepción están limitadas a recibir justamente una señal, y son más simples y menos costosas. Existen dos modos de operación para la transmisión serial: **simplex,** el cual corresponde a una transmisión en un solo sentido; y **duplex**, el cual permite la transmisión en cualquier dirección. La transmisión simplex requiere sólo de un receptor y un transmisor, a cada extremo del enlace; de otra parte, la transmisión duplex puede suceder de una de dos maneras: **medio-duplex ("half duplex")** y **duplex-completo (full duplex).** En la primera, aunque la transmisión puede ocurrir en ambas direcciones, ni puede ocurrir simultáneamente en ambas direcciones; en el segundo caso, ambos extremos pueden simultáneamente transmitir y recibir. La transmisión duplex-completa se implementa usualmente por medio de cuatro alambres.

La tasa de datos de una línea de transmisión serial se mide en bits por segundo, ya que los datos se transmiten un bit cada vez. La unidad de 1 bit/s se denomina **baudio**; por lo tanto, frecuentemente se hace referencia a la tasa en baudios de una transmisión serial. La tasa en baudios se puede trasladar a una tasa de transmisión paralela en palabras por segundo, si se conoce la estructura de la palabra; por ejemplo, si una palabra consta de 10 bits (bit de arranque y de parada más 8 bits para la palabra de datos) y la transmisión tiene lugar a 1200 baudios, 120 palabras se transmiten cada segundo. Las tasas de datos típicos de la transmisión serial están estandarizadas; las más comunes (familiares a los usuarios de conexiones módem de computadoras

personales) son 300, 600, 1200 y 2400 baudios. Las tasas pueden ser tan bajas como 50 baudios o tan altas como 19200 baudios.

Tal como en la transmisión paralela, la transmisión serial puede también ocurrir sincrónicamente o asincrónicamente. En el caso serial, es verdad que las transmisión asíncrona es mas económica, pero no la más rápida. También se requiere un protocolo handshake[32] para la transmisión serial asíncrona. El esquema más popular de codificación de datos para la transmisión serial es, una vez más, el código ASCII, que consiste en una palabra de 7 bits más un **bit de paridad**, para un total de 8 bits por caracter. El papel del bit de paridad es permitir la detección de errores, en el evento de una recepción (o transmisión) errónea de un bit. Para ver esto, se analiza la secuencia de los eventos del protocolo handshake para una transmisión seriales asíncrona y el uso de bits de paridad para corregir errores. En sistemas serial asincrónicos, el protocolo handshake se realiza usando bits de arranque y parada al inicio y final de caracter que se transmite. El inicio de la transmisión de una palabra serial asincrónica se anuncia por el bit de “arranque”, el cual es siempre un bit de estado 0. Para los siguientes 5 a 8 tiempos de bits sucesivos (dependiendo del código y el número de bits que especifican la longitud de palabra en el código), la línea se conmuta a los estados 1 y 0 requeridos para representar el caracter enviado. Después del último bit de los datos y del bit de paridad, hay uno o más bits en estado 1, indicando “disponible” (idle). El periodo asociado con esta transmisión se denomina intervalo de bit de “parada”.

32 Especifica el protocolo de control utilizado para establecer una comunicación con un puerto serie para un objeto SerialPort.

La transmisión serial de datos ocurre frecuentemente de acuerdo con la **norma RS-232** y sus características principales son las siguientes:

- Las señales de datos se codifican de acuerdo con la convención lógica negativa usando niveles de voltaje de -3 a -15 V para un 1 lógico y +3 a +15 V para un 0 lógico.
- Las señales de control usan una convención lógica positiva (opuesta a la de las señales de datos).
- La máxima capacitancia en paralelo (shunt) de la carga no puede exceder a 2500 pF; estom en efecto, limita la longitud máxima de los cables usados en la conexión.
- La carga resistiva debe estar entre 300 Ω y 3 kΩ.
- Tres alambres se usan para la transmisión de datos. Un alambre se usa para recibir y otro para transmitir datos; el tercer alambre es una línea de señal de retorno (tierra). Además, hat 22 alambres que se pueden usar para un variedad de procesos de control entre el DTE[33] y el DCE[34].
- La parte “macho” del conector se asigna al DTE, y la parte “hembra” añ DCE.
- La tasa de baudios se limita por la longitud del cable; para un cable de 17 m, se permite cualquier tasa de 50 baudios a 19.2 kbaudios. Si se desea un cable más largo, la máxima tasa de baudios disminuye de acuerdo con la longitud del cable.

33 DTE: Data terminal equipment, Equipo terminal de datos.
34 DCE: Data communication equipment. Equipo de comunicación de datos.

- Los datos seriales pueden ser codificados de acuerdo con cualquier código, aunque el código ASCII es, sin duda, el más popular.

2.6.7. Comunicación USB

El **USB** (*Bus de serie universal*), como su nombre lo sugiere, se basa en una arquitectura de tipo serial. Sin embargo, es una interfaz de entrada/salida mucho más rápida que los puertos seriales estándar. La arquitectura serial se utilizó para este tipo de puerto por dos razones principales:

- La arquitectura serial le brinda al usuario una velocidad de reloj mucho más alta que la interfaz paralela debido a que este tipo de interfaz no admite frecuencias demasiado altas (en la arquitectura de alta velocidad, los bits que circulan por cada hilo llegan con retraso y esto produce errores).
- Los cables seriales resultan mucho más económicos que los cables paralelos.

Estándares USB

A partir de 1995, el estándar USB se ha desarrollado para la conexión de una amplia gama de dispositivos.

El estándar **USB 1.0** ofrece dos modos de comunicación:

- 12 Mb/s en modo de alta velocidad,
- 1,5 Mb/s de baja velocidad.

El estándar **USB 1.1** brinda varias aclaraciones para los fabricantes de dispositivos USB, pero no cambia los rasgos de velocidad.

El estándar **USB 2.0** permite alcanzar velocidades de hasta 480 Mbit/s.

La mejor manera de determinar si un dispositivo es de USB de alta o baja velocidad es consultar la documentación del producto, siempre y cuando los conectores sean los mismos.

La compatibilidad entre USB 1.0, 1.1 y 2.0 está garantizada. Sin embargo, el uso de un dispositivo USB 2.0 en un puerto USB de baja velocidad (es decir 1.0 ó 1.1) limitará la velocidad a un máximo de 12 Mbit/s. Además, es probable que el sistema operativo muestre un mensaje que indique que la velocidad será restringida.

Tipos de conectores

Existen dos tipos de conectores USB:

- Los conectores conocidos como **tipo A** (figura 2.14), cuya forma es rectangular y se utilizan, generalmente, para dispositivos que no requieren demasiado ancho de banda (como el teclado, el ratón, las cámaras Web, etc.);
- Los conectores conocidos como **tipo B** (figura 2.14)poseen una forma cuadrada y se utilizan principalmente para dispositivos de alta velocidad (discos duros externos, etc.).

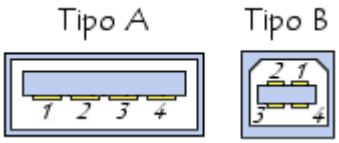


Figura 2.14. Tipos de conectores USB

1. Fuente de alimentación de +5 V (*VBUS*) máximo 100 mA
2. Datos (*D*-)
3. Datos (*D*+)
4. Conexión a tierra (*GND*)

Funcionamiento del USB

Una característica de la arquitectura USB es que puede proporcionar fuente de alimentación a los dispositivos con los que se conecta, con un límite máximo de 15 V por dispositivo. Para poder hacerlo, utiliza un cable que consta de cuatro hilos (la conexión a tierra *GND*, la alimentación del *BUS* y dos hilos de datos llamados *D*- y *D*+) (figura 2.15).

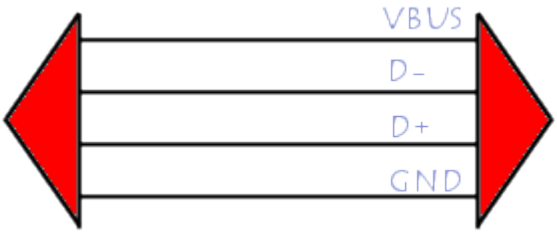


Figura 2.15. Conexión de los cables del USB

El estándar USB permite que los dispositivos se encadenen mediante el uso de una topología en bus o de estrella. Por lo tanto, los dispositivos pueden

conectarse entre ellos tanto en forma de cadena como en forma ramificada. La ramificación se realiza mediante el uso de cajas llamadas "**concentradores**" que constan de una sola entrada y varias salidas. Algunos son activos (es decir, suministran energía) y otros pasivos (la energía es suministrada por el ordenador).

La comunicación entre el host (equipo) y los dispositivos se lleva a cabo según un protocolo (lenguaje de comunicación) basado en el principio de red en anillo. Esto significa que el ancho de banda se comparte temporalmente entre todos los dispositivos conectados. El host (equipo) emite una señal para comenzar la secuencia cada un milisegundo (ms), el intervalo de tiempo durante el cual le ofrecerá simultáneamente a cada dispositivo la oportunidad de "hablar". Cuando el host desea comunicarse con un dispositivo, transmite una red (un paquete de datos que contiene la dirección del dispositivo cifrada en 7 bits) que designa un dispositivo, de manera tal que es el host el que decide "hablar" con los dispositivos. Si el dispositivo reconoce su dirección en la red, envía un paquete de datos (entre 8 y 255 bytes) como respuesta. De lo contrario, le pasa el paquete a los otros dispositivos conectados. Los datos que se intercambian de esta manera están cifrados conforme a la codificación NRZI[35].

Como la dirección está cifrada en 7 bits, 128 dispositivos (2^7) pueden estar conectados simultáneamente a un puerto de este tipo. En realidad, es recomendable reducir esta cantidad a 127 porque la dirección 0 es una dirección reservada.

---

35 NRZI: Non return to zero – inverted, No retorna a cero – invertido: Al transmitir un 0 no se produce transición y en cambio al enviar un 1 se produce una transición a nivel positivo o negativo.

Debido a la longitud máxima de 5 metros del cable entre los dos dispositivos y a la cantidad máxima de 5 concentradores (a los que se les suministra energía), es posible crear una cadena de 25 metros de longitud.

Los puertos USB admiten dispositivos **Plug and play de conexión en caliente**. Por lo tanto, los dispositivos pueden conectarse sin apagar el equipo (**conexión en caliente**). Cuando un dispositivo está conectado al host, detecta cuando se está agregando un nuevo elemento gracias a un cambio de tensión entre los hilos D+ y D-. En ese momento, el equipo envía una señal de inicialización al dispositivo durante 10 ms para después suministrarle la corriente eléctrica mediante los hilos *GND* y *VBUS* (hasta 100 mA). A continuación, se le suministra corriente eléctrica al dispositivo y temporalmente se apodera de la dirección predeterminada (dirección 0). La siguiente etapa consiste en brindarle la dirección definitiva. Para hacerlo, el equipo interroga a los dispositivos ya conectados para poder conocer sus direcciones y asigna una nueva, que lo identifica por retorno. Una vez que cuenta con todos los requisitos necesarios, el host puede cargar el driver adecuado.

2.6.8. Comunicación Ethernet

Ethernet es un estándar de redes de área local para computadores con acceso al medio por contienda CSMA/CD. CSMA/CD (Acceso Múltiple por Detección de Portadora con Detección de Colisiones), es una técnica usada en redes Ethernet para mejorar sus prestaciones. El nombre viene del concepto físico de *ether*. Ethernet define las características de cableado y señalización de nivel

físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI[36].

La Ethernet se tomó como base para la redacción del estándar internacional IEEE 802.3. Usualmente se toman Ethernet e IEEE 802.3 como sinónimos. Ambas se diferencian en uno de los campos de la trama de datos. Las tramas Ethernet e IEEE 802.3 pueden coexistir en la misma red.

Ethernet, opera a 10 Mb/s y es la base de las redes de altas velocidades, estas redes de alta velocidad incluyen **interfaz de datos de fibra distribuida (fiber-distibuted data interface, FDDI)**, la cual especifica un anillo de fibra óptica con una tasa de datos de 100 Mb/s, y el **modo de transferencia asincrónica (asynchronous transfer modo, ATM)**, un modo de transferencia orientado a paquetes que mueven datos en paquetes fijos llamados celdas. ATM no opera a velocidad fija. Una velocidad típica es de 155 Mb/s, pero hay implementaciones que corren a velocidades hasta de 2 Gb/s.

36 Modelo OSI: Open system interconnection, Modelo de interconexión de sistemas abiertos.

Capítulo III

DISEÑO E IMPLEMENTACIÓN

3.1. Introducción

Para el diseño de los módulos de entrenamiento sobre plataforma Arduino se tomó en cuenta los siguientes parámetros:

- Arduino posee E/S analógicas y digitales, así que se incluyó elementos electrónicos con el fin de utilizar y demostrar esta característica de la plataforma.
- La utilización de elementos básicos que pueden funcionar con cualquier microcontrolador.
- La utilización de elementos un poco más complejos que con otros microcontroladores es muy complicada la programación en cambio con Arduino se simplifica.
- La aplicación de la Arduino Ethernet con experimentos reales a través de la red LAN y la internet.
- Los principios de Arduino es su interactividad entre el mundo físico y el virtual, en este sentido el diseño va direccionado a la utilización de los elementos electrónicos instalados con Arduino y Processing.

3.2. Diseño del módulo de entrenamiento sobre plataforma Arduino

Basándose en los parámetros y necesidades mencionadas en la introducción, los módulos de entrenamiento incluyen las siguientes etapas:

- Arduino Uno
- Arduino Ethernet Shield (Desmontable)
- Switchs
- Leds RGB
- Matriz de leds de 8 x 8
- LCD
- Sensor de temperatura
- Sensor infrarrojo (Tx - Rx)
- Servomotor
- Motor paso a paso
- Potenciómetros
- Parlante
- Protoboard

El diseño de cada etapa se lo realizó de manera independiente y sin relación alguna entre estas, con esto se facilita la implementación y distribución de los elementos en el módulo de entrenamiento, es decir el voltaje de alimentación y la tierra no son comunes entre las etapas del módulo.

Para los diagramas esquemáticos se utilizó el programa Fritzing[37]. Fritzing permite realizar diagramas de circuitos electrónicos, visualización en PCB y protoboard, además es software libre y creado bajo el espíritu de Processing y Arduino.

37 www.fritzing.org

Para los cálculos de los diferentes valores necesarios del diseño se tomó en cuenta los siguientes parámetros:

- Voltaje de salida de cada pin: 5 V
- Corriente de salida de cada pin: 40 mA

Estos parámetros de voltaje y corriente están dados por las características técnicas de la placa Arduino (tabla 2.1).

Microcontrolador	ATmega368
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines E/S digitales	14 (6 proporcionan salida PWM)
Pines de entrada analógica	6
Intensidad por pin	40 mA
Intensidad en pin 3.3V	50 mA
Memoria Flash	32 KB de las cuales 2 KB las usa el gestor de arranque(bootloader)
SRAM	2 KB
EEPROM	1 KB
Velocidad de reloj	16 MHz

3.2.1. Diseño de la etapa de los switchs

Todo interruptor o switch representa una entrada digital para cualquier sistema electrónico. Según se han revisado los módulos de entrenamiento con PIC existentes en los laboratorios de electrónica de la Universidad Israel y de acuerdo a los manuales y experimentos diseñados para la plataforma Arduino, con 3 switch se tendrá una cantidad óptima de operación.

A la salida de cada switch se conecta una resistencia pull – down de 10 KΩ, en el caso de que el switch esté en estado abierto, envía al pin de entrada del Arduino una señal lógica de 0 V ó LOW, caso contrario, cuando el switch es presionado se lee en el pin de entrada del Arduino una señal lógica de 5 V ó HIGH, sin esta resistencia se quemaría el pin debido a que se sobrepasaría la cantidad de corriente que éste soporta, que es de 40 mA. Ahora bien, con la resistencia de 10 KΩ se obtiene solamente una corriente de 0.5 mA que está muy por debajo del límite superior de corriente pero que es suficiente para entender y aceptar el nivel lógico de 5 V ó HIGH. Ver figura 3.1.

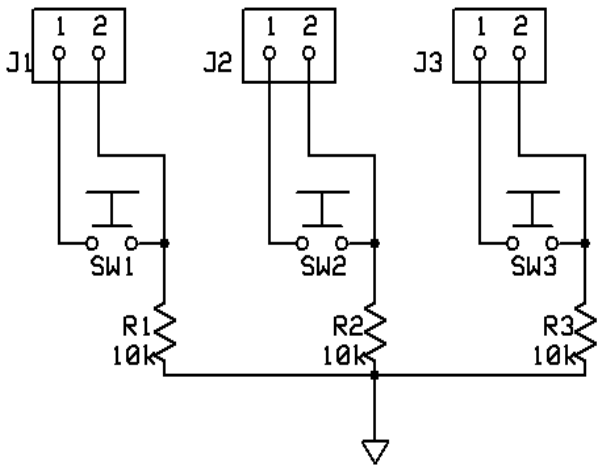


Figura 3.1. Diagrama esquemático de la etapa de los switchs

3.2.2. Diseño de la etapa de los LEDs RGB

Los LEDs son elementos básicos que se utilizan para comenzar el aprendizaje de programación de microcontroladores, además es un elemento de salida que con Arduino puede funcionar como digital o analógico, por este motivo se vuelve imprescindible su implementación en el módulo de entrenamiento.

El LED RGB está constituido por 3 LEDs (rojo, verde y azul) en un solo encapsulado en el cual internamente se encuentra constituido por un cátodo común y 3 ánodos independientes, con esto se tiene 4 pines de conexión.

En la tabla 3.1, extraída del Anexo 1, se tiene las características técnicas del LED RGB, donde se puede verificar los siguientes valores de importancia para el funcionamiento correcto del LED:

- I_F : Corriente del LED = 20 mA
- V_F :Voltaje del LED = 2.0 V (rojo); 3.2 V (verde y azul)

ITEM	COLOR	SYMBOL	CONDITION	MIN	TYPE	MAX	UNIT
Forward Voltage	Red	V_F	$I_F = 20\,mA$	1.8	2.0	2.2	V
	Green			3.0	3.2	3.2	
	Blue			3.0	3.2	3.2	

Tabla 3.1. Características técnicas del led RGB

La corriente necesaria I_F debe ser igual a 20 mA y los pines de salidas del Arduino entregan hasta 40 mA, en este caso la placa entrega sin inconvenientes la corriente requerida al LED.

El LED trabaja con los voltajes 2 V (rojo) y 3.2 V (verde y azul), y el voltaje de los pines de salida del Arduino es de 5 V, con lo que si se conecta directo al LED podría quemarse, en este caso es necesario ocupar una resistencia limitadora.

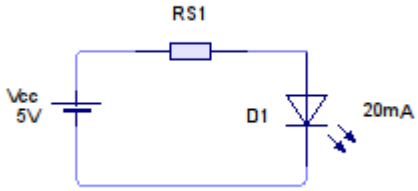


Figura 3.2. Circuito serie de un led

Para el cálculo de la resistencia limitadora (RS1) de la figura 3.2. se aplica la siguiente fórmula:

$$RS1 = \frac{Vcc - V_F}{I_F}$$

Donde:

RS1: Resistencia limitadora

Vcc: Voltaje del pin de salida digital de Arduino (5 V)

V_F : Voltaje del LED (1.8 V)

I_F : Corriente del LED (20 mA)

Como se observa en la tabla 3.1 los valores de voltaje varían en cada color del LED RGB, para los cálculos se usó el caso más crítico que corresponde al color rojo con V_F = 1.8 V.

$$RS1 = \frac{5V - 1.8V}{0.02A}$$

$$RS1 = \frac{3.2V}{0.02A}$$

$$RS1 = 160\ \Omega$$

La resistencia calculada es de 160 Ω, pero no existen resistencias comerciales con ese valor y por seguridad es recomendable usar de un ohmiaje mayor debido a la variación por la tolerancia; bajo estos fundamentos la RS1 a usarse es de 220 Ω.

En la figura 3.3 se muestra el diagrama esquemático de la etapa de leds RGB utilizando la resistencia calculada.

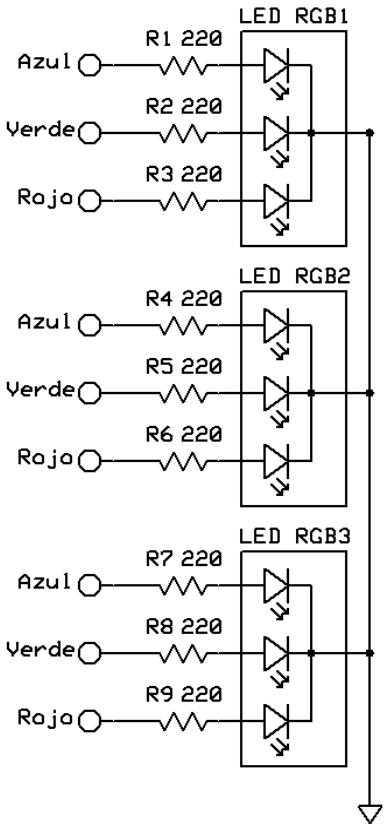


Figura 3.3. Diagrama esquemático de la etapa de leds RGB

3.2.2. Diseño de la etapa de matriz de LEDs 8x8

La matriz de LEDs de 8x8 en su interior consta de 64 LEDs unidos entre sus cátodos en forma de filas y todos sus ánodos en forma de columnas, obteniendo 8 filas y 8 columnas; con esto se puede encender cualquier LED de la matriz enviando un 1 lógico (5 V) a la fila correspondiente y un 0 lógico (GND) a la columna donde se encuentre el LED a encenderse. Si queremos encender toda la matriz (64 LEDs) se debe enviar 5 V a las 8 filas y poner en GND a las 8 columnas, en este caso se deben manipular las 16 variables (8 filas y 8 columnas) a través de las salidas de un microcontrolador, siendo un valor alto de salidas a controlar. Además si los 64 LEDs se encienden se tendría el siguiente consumo de corriente:

$$I_T = 64 * I_F$$

$$I_T = 64 * 0.02A$$

$$I_T = 1.28A$$

Es una cantidad de corriente que no se puede soportar con ningún microcontrolador, por lo que se utilizan etapas de potencia formados por transistores, aumentando aun más el diseño de cualquier circuito.

Arduino posee shields diseñadas para aplicaciones específicas, para el caso de manipulación de matrices se tiene la placa Rainbowduino (figura 3.4)

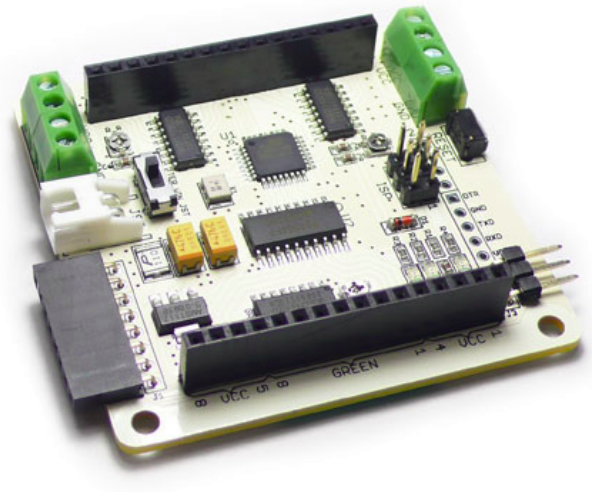


Figura 3.4. Placa Rainbowduino[38]

La placa Rainbowduino permite a través de la programación de Arduino manejar hasta 192 LEDs, es decir una matriz de 8x8 tricolor, además no existe problema con lo que respecta a la corriente ya que está diseñada para el caso de consumo más critico. El diseño de la Rainbowduino se basó en experimentos anteriores incluidos en la librería Arduino que usa el MAX7219 (Anexo 2). Las características eléctricas del MAX7219 se observa en la tabla 3.2.

PARAMETER	SYM	CONDITIONS	MIN	TYP	MAX	UNIT
Operating Supply Voltage	V+		4.0		5.0	V
Shutdown Supply Current	I+	All digital inputs at V+ or GND, TA = +25°C			150	µA
Operating Supply Current	I+	RSET = open circuit			8	mA
		All segments and decimal point on, ISEG_ = -40mA		330		
Display Scan Rate	f_{OSC}	8 digits scanned	500	800	1300	Hz
Digit Drive Sink Current	I_{DIGIT}	V+ = 5V, VOUT = 0.65V	320			mA
Segment Drive Source Current	I_{SEG}	TA = +25°C, V+ = 5V, VOUT = (V+ - 1V)	-30	-40	-45	mA

38 http://www.seeedstudio.com/depot/rainbowduino-led-driver-platform-atmega-328-p-371.html

Segment Drive Current Matching	ΔI_{SEG}			3.0	%
Digit Drive Source Current	I_{DIGIT}	Digit off, VDIGIT = (V+ - 0.3V)	-2		mA
Segment Drive Sink Curren	I_{SEG}	Segment off, VSEG = 0.3V	5		mA
LOGIC INPUTS					
Input Current DIN, CLK, LOAD, CS	I_{IH} , I_{IL}	VIN = 0V or V+	-1	1	µA
Logic High Input Voltage	V_{IH}		3.5		V
Logic Low Input Voltage	V_{IL}			8	V
Output High Voltage	V_{OH}	DOUT, ISOURCE = -1mA	V+-1		V
Output Low Voltage	V_{OL}	DOUT, ISINK = 1.6mA		0.4	V
Hysteresis Voltage	ΔV_I	DIN, CLK, LOAD, CS		1	V

Tabla 3.2. Características eléctricas del MAX7219

Como se observa en la figura 3.5 y según las características técnicas del MAX7219, se acopla sin inconvenientes a una matriz de 8x8 unicolor o a 8 displays de 7 segmentos de cátodo común. Además su voltaje de alimentación es de 4 a 5 V, la placa Arduino en su pin de salida de Vcc provee 5 V, con lo cual se puede alimentar sin inconvenientes al MAX7219.

El MAX7219 se puede conectar en cascada para formar paneles de matrices de LEDs a manera de pantallas, y todos pueden ser manejados por una placa Arduino.

La matriz de LEDs ocupada para el módulo de entrenamiento se observa en la figura 3.6 y el diagrama esquemático en la figura 3.7. Es una matriz de 8x8 bicolor, para el diseño sólo se ocupó los pines de color rojo, se pueden utilizar los pines de color verde con otro MAX7219 y ser manipulados por la misma placa Arduino.

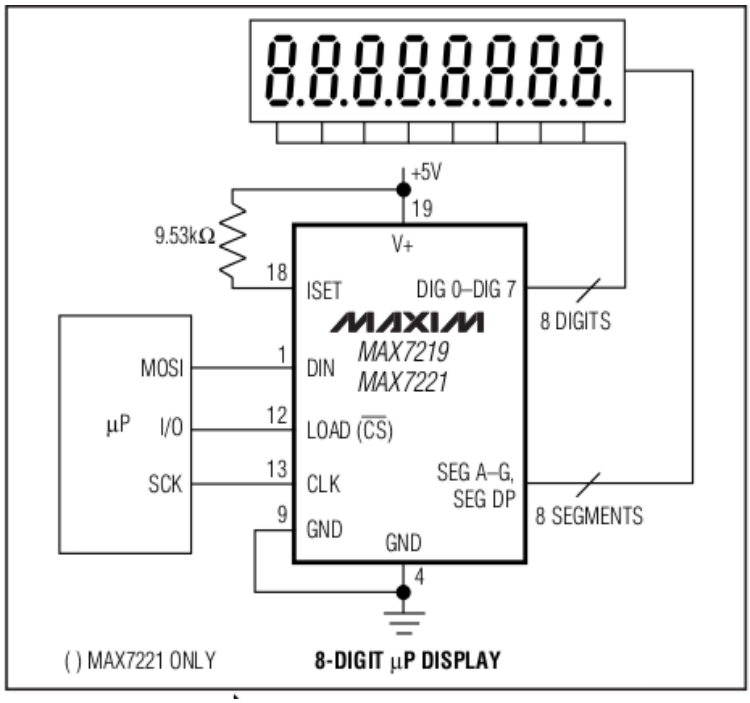


Figura 3.5. Aplicación típica del MAX7219

Figura 3.6. Matriz de 8x8 bicolor

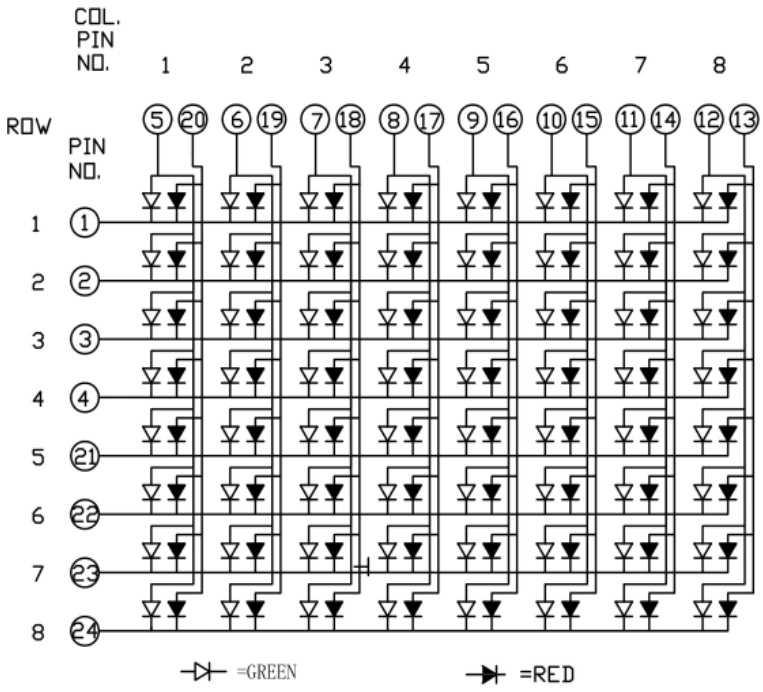


Figura 3.7. Diagrama de matriz de 8x8 bicolor

Para el diseño se reemplazó la resistencia limitadora de corriente de 9,53 KΩ que especifica el datasheet del MAX7219 por un valor comercial de 10 KΩ. Se agregó 2 capacitores de 10 µF y de 0.01 µF al ingreso de la alimentación del MAX7219 para evitar señales de ruido en la alimentación que se reflejaban en intermitencia de los leds debido a la multiplexación de los datos transmitidos a la matriz.

En el diagrama esquemático de la figura 3.8, se puede observar las conexiones del MAX7219 a la matriz.

Al momento del diseño de la PCB de esta etapa presentó problemas por la cantidad de pistas de cobre existentes, por lo cual se utilizó regletas de conexión para facilitar la implementación (ver figura 3.9)

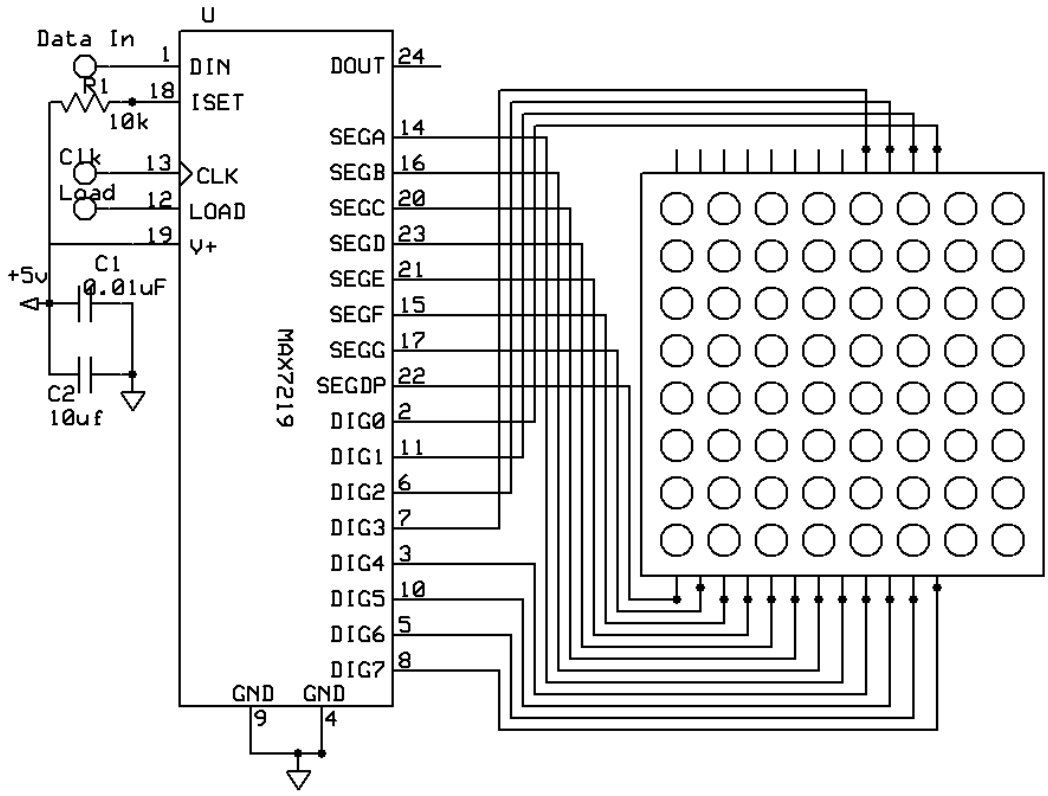


Figura 3.8. Diagrama esquemático de la etapa de matriz 8x8

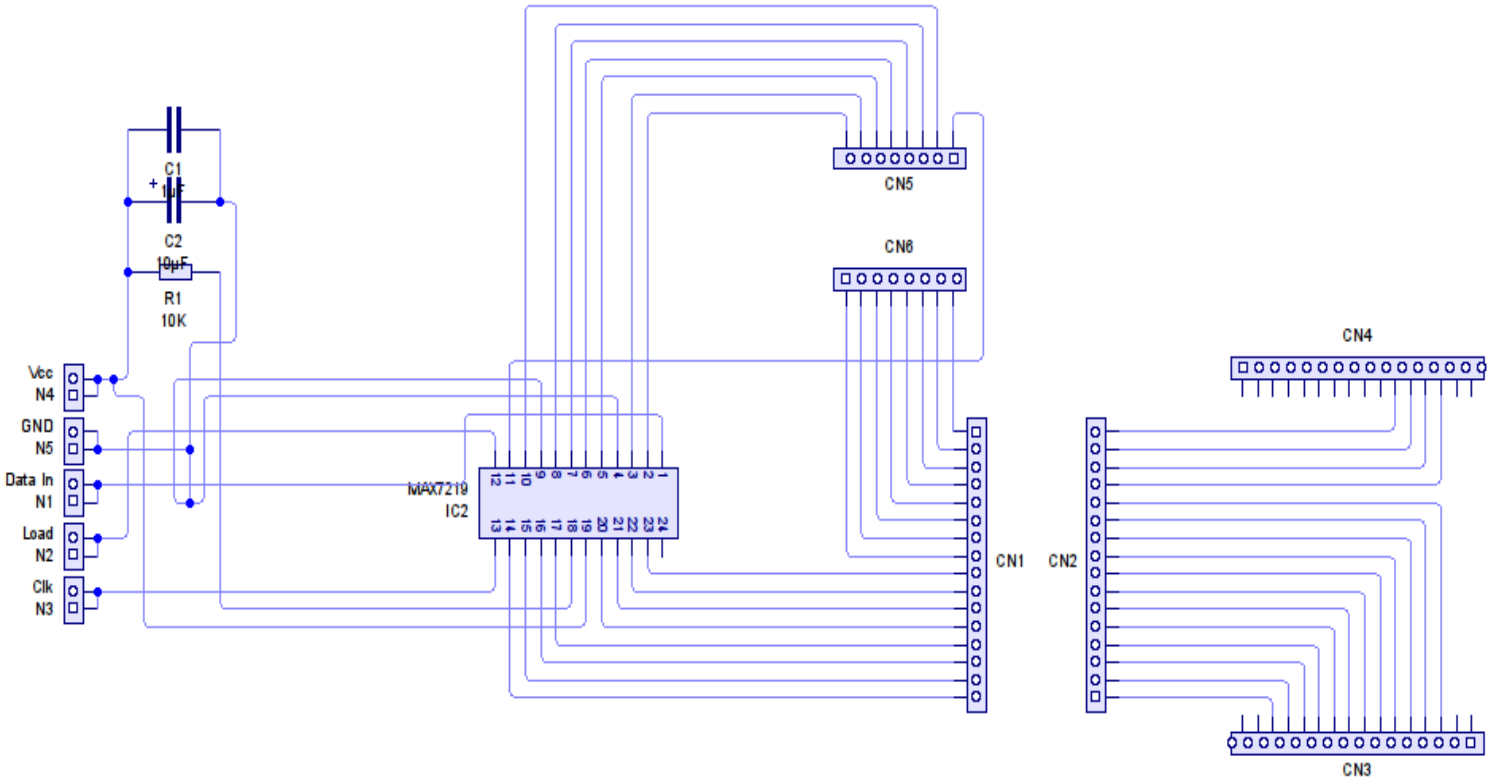


Figura 3.9. Diagrama de conectores de la etapa de matriz 8x8.

3.2.3. Diseño de la etapa del LCD

La pantalla LCD permite visualizar caracteres definidos dependiendo de la programación que se realice en la placa Arduino. Por ejemplo, si se está monitoreando la temperatura de un sensor a través del Arduino y se quiere observar los valores obtenidos, los 2 mejores métodos son:

- Monitoreo Serial: Se puede ver la temperatura a través del puerto serie que se tiene en el IDE de Arduino.
- Pantalla LCD: Se visualiza los valores de temperatura por medio de la LCD sin tener que usar el IDE de Arduino.

Para el diseño de esta etapa se se utilizó el LCD de 16x2 sin Backlight[39], el Backlight es útil en lugares oscuros ya que la pantalla provee iluminación adicional que ayuda a visualizar de mejor manera los caracteres desplegados, pero los módulos de entrenamiento serán utilizados en las aulas de la Universidad Israel que poseen buena iluminación por lo cual el Backlight no es necesario.

En la tabla 3.3 se observa las características eléctricas del LCD y en la tabla 3.4 la distribución de pines, extraídas del Anexo 4.

En la figura 3.10 se tiene el esquemático de los pines del LCD, como se ve tiene 16 pines pero no todos serán utilizados en la implementación del módulo de entrenamiento, al no ser imprescindibles en su conexión.

39 Backlight: Retroiluminación, es un tipo de iluminación de pantallas LCD.

ELECTRICAL CONDITION						
ITEM	**SYMBOL**	**CONDITION**	**STANDARD VALUE**			**UNIT**
			MIN	**TYP**	**MAX**	
Input Voltage	VDD	VDD=+5V	4.7	5.0	5.3	V
		VDD=+3V	2.7	3.0	5.3	V
Supply Current	IDD	VDD=5V	-	1.2	3.0	mA
Recommended LC Driving Voltage for Normal Temp. Version Module	VDD-V0	- 20°C	-	-	-	V
		0°C	4.2	4.8	5.1	
		25°C	3.8	4.2	4.6	
		50°C	3.6	4.0	4.4	
		70°C	-	-	-	
Led Forward Voltage	VF	25°C	-	4.2	4.6	V
Led Foreard Current	IF	25°C Array	-	130	260	mA
		Edge	-	20	40	
EL Power Supply Current	IEL	Vel: 110VAC: 400HZ	-	-	5.0	mA

Tabla 3.3. Características eléctricas del LCD 16x2

PIN NUMBER	SYMBOL	FUCTION
1	Vss	GND
2	Vdd	+ 3V or + 5V
3	Vo	Contrast Adjustment
4	RS	H/L Register Select Signal
5	R/W	H/L Read/Write Signal
6	E	H → L Enable Signal
7	DB0	H/L Data Bus Line
8	DB1	H/L Data Bus Line
9	DB2	H/L Data Bus Line
10	DB3	H/L Data Bus Line
11	DB4	H/L Data Bus Line
12	DB5	H/L Data Bus Line
13	DB6	H/L Data Bus Line
14	DB7	H/L Data Bus Line
15	A/Vee	+ 4.2V for LED/Negative Voltage Output
16	K	Power Supply for B/L (OV)

Tabla 3.4. Pines del LCD 16x2

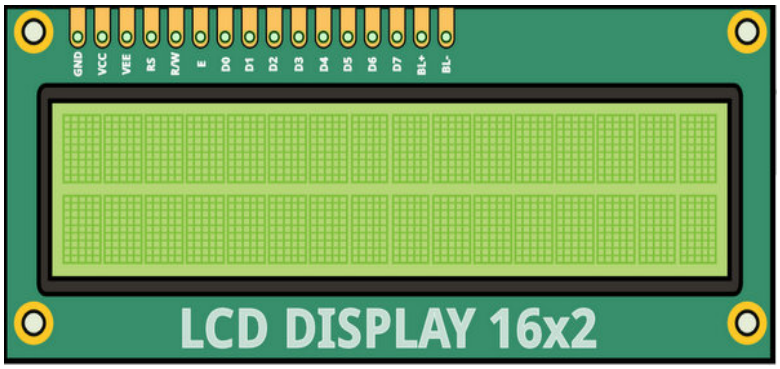


Figura 3.10. Esquema de los pines del LCD 16x2

Como se observa en la tabla 3.4 el voltaje de alimentación del LCD es de 5V y el consumo de corriente es de 1.2 mA. Ambos valores son entregados por la Arduino y permite la conexión directa sin circuitos de acoplamiento entre el LCD y la Arduino.

En la tabla 3.5 indica que el pin 3 (Vo) regula el contraste de la pantalla, para lo cual se conectó un potenciómetro de 10 KΩ para regular este parámetro. Los pines 15 y 16 son los que regulan el Blacklight, en el caso del diseño de los módulos de entrenamiento se utilizaron LCD sin contraste Blacklight, así que estos pines quedan sin conexión. El diagrama esquemático se observa en la figura 3.11.

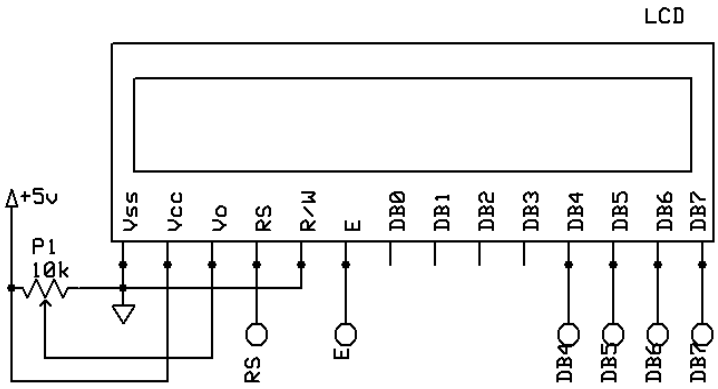


Figura 3.11. Diagrama esquemático de la etapa de LCD

3.2.4. Diseño de la etapa del sensor de temperatura

Un sensor de temperatura es un elemento que funciona como entrada analógica para la placa Arduino, para el diseño su utilizó el LM35 (figura 3.12, extraída del Anexo 5) por las ventajas de conexión y porque el voltaje de alimentación esta entre los 4 a 20 V, con lo cual se conecta directamente a Vcc del Arduino sin necesidad de circuitos adicionales.

La escala de medición del sensor varía en 0 a +10.0 mV/°C, que en temperatura es de +2°C a +150°C, otra variable que facilita la elaboración de experimentos prácticos.

En la figura 3.13 se muestra el diagrama esquemático de esta etapa, como se puede ver, no es necesario ninguna conexión extra para el funcionamiento del sensor.

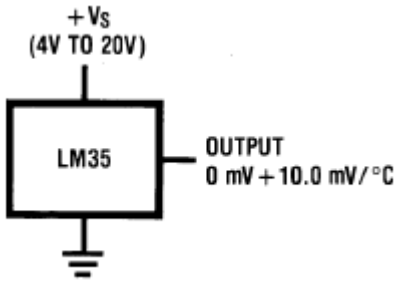


Figura 3.12. Aplicación típica del LM35

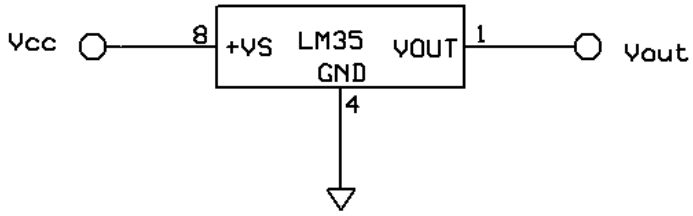


Figura 3.13. Diagrama esquemático de la etapa del sensor de temperatura

3.2.5. Diseño de la etapa del sensor infrarrojo (Tx – Rx)

Los elementos usados para el diseño son un LED IR que hace el trabajo de transmisor y un fototransistor que hace de receptor.

El LED IR utilizado es de encapsulado transparente y con las características eléctricas de la tabla 3.5 extraída del Anexo 6.

Part Number	Lens color	Forward voltage V_F (V)		Condition I_F	Reverse current I_R (μA)	Condition V_R	Optical Power I_e (mW/sr)	Condition	Peak wavelength λP (nm)	Condition I_F	Spectrum half width $\Delta\lambda$ (nm)	Condition I_F	Chip material	Dimension A (mm)
		typ	max	(mA)	max	(V)	typ		typ	(mA)	typ	(mA)		
SID303C	Clear	1.3	1.5	50	10	5	80	(Constant	940	50	50	50	GaAs	$3.0^{\pm0.5}$
SID313BP	Transparent light purple	1.3	1.5	50	10	5	130	voltage)	940	50	50	50		$3.6^{\pm0.5}$
SID1003BQ	Transparent light navy blue	1.3	1.5	50	10	5	180	V_{CC}=3V	940	50	50	50		
SID307BR	Transparent dark navy blue	1.3	1.5	50	10	5	200	R=2.2Ω	940	50	50	50		$4.2^{\pm0.5}$
SID1G307C	Clear	1.5	1.8	50	10	5	50	I_F=50mA	850	50	40	50		

Tabla 3.5. Características eléctricas del LED IR

Las características eléctricas (voltaje) del IR son similares a las del LED convencional y si se observa la figura 3.14, se puede notar que el rango de infrarrojo se encuentra sobre el rango de la luz visible roja, basándose en esto y la similitud de características los cálculos se realizan con los valores del LED rojo de la tabla 3.1.

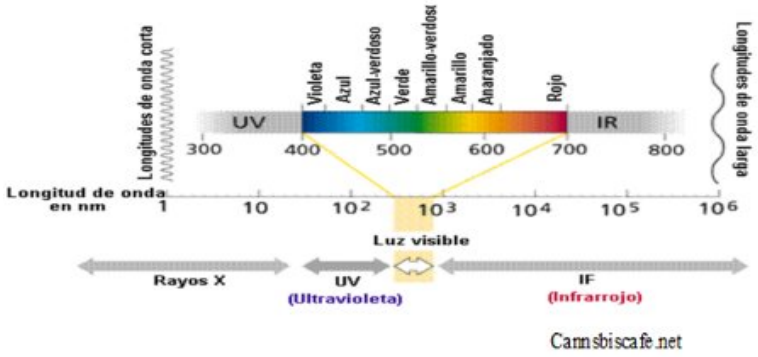


Figura 3.14. Rango de luz visible[40]

Donde la a fórmula aplicada es:

$$RS1 = \frac{Vcc - V_F}{I_F}$$

Así se tiene que RS1 = 160 Ω, bajo el mismo criterio que la etapa de LEDs RGB, la resistencia a usarse es de 220 Ω.

En la figura 3.15 se observa el lóbulo de direccionamiento del IR, en todos los casos de los diferentes IR se tiene un mayor porcentaje de señal si se encuentra direccionado a 90°.

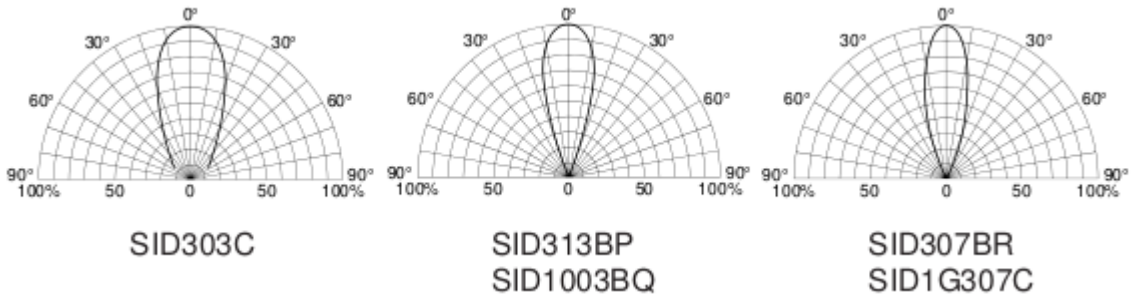


Figura 3.15. Direccionamiento del lóbulo del IR

40 http://www.biologia.edu.ar/plantas/fotosint1.htm

Para que el fototransistor entre en operación debe recibir la señal externa de algún elemento, en este caso, del IR. Según se observa en la figura 3.16, la ganancia o cantidad de señal recibida depende de la posición de la emitida, donde a 0° se obtiene una señal completa y a 90° no se recibe señal, por este motivo y tomando en cuenta el direccionamiento del lóbulo del IR la mejor recepción para transmisión y recepción se tiene cuando los elementos se encuentran frente a frente. El diagrama esquemático se observa en la figura 3.17.

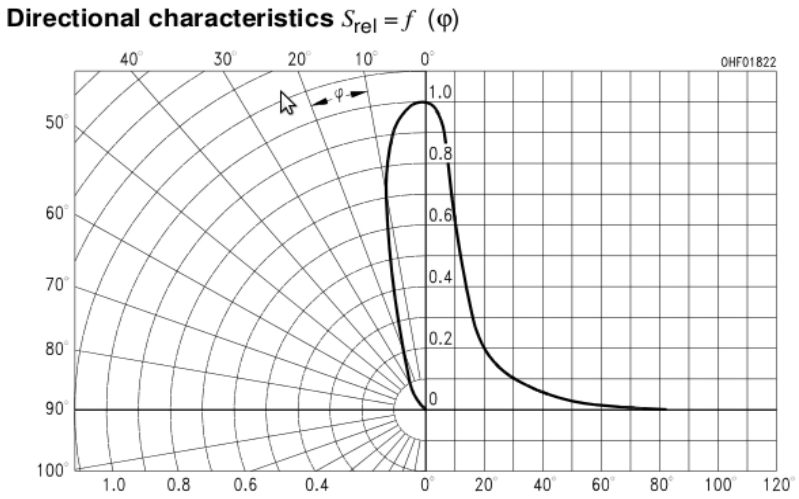


Figura 3.16. Características direccionales del fototransistor

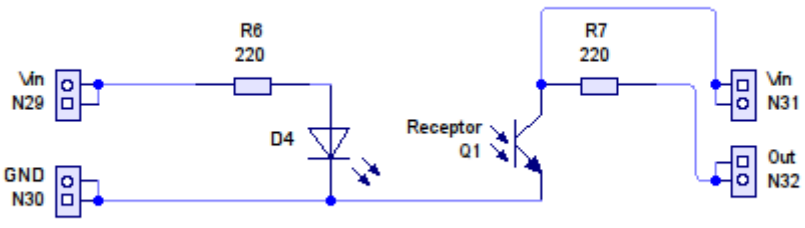


Figura 3.17. Diagrama esquemático de la etapa del sensor infrarrojo

3.2.6. Diseño de la etapa del servomotor

El servomotor o servo, es un dispositivo similar a un motor de corriente continua que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y mantenerse estable en dicha posición. Un servomotor es un motor eléctrico que consta con la capacidad de ser controlado, tanto en velocidad como en posición.

Los servomotores hacen uso de la modulación por ancho de pulsos (PWM) para controlar la dirección o posición de los motores de corriente continua. La mayoría trabaja en la frecuencia de los cincuenta hercios, así las señales PWM tendrán un periodo de veinte milisegundos. La electrónica dentro del servomotor responderá al ancho de la señal modulada. Si los circuitos dentro del servomotor reciben una señal de entre 0,5 a 1,4 milisegundos, este se moverá en sentido horario; entre 1,6 a 2 milisegundos moverá el servomotor en sentido antihorario; 1,5 milisegundos representa un estado neutro para los servomotores estándares.

Para el diseñó se utilizó el servo DY-S0209-38g con giro de 180°, que tiene las siguientes características técnicas:

- Voltaje: 4.8 a 6.0 V
- Velocidad: 0.18 sec/60°
- Torque: 3.5 Kg x cm
- Tamaño: 40.8 x 20.1 x 38.0 mm
- Peso 38 gr

La información de los cables de alimentación es:

- Cable Naranja: Señal de control
- Cable Rojo: 4.8 a 6.0 V
- Cable Café: GND

Por las características del servo, no es necesario alguna conexión adicional para manejarlo con la Arduino. El diagrama esquemático del servomotor se observa en la figura 3.18.

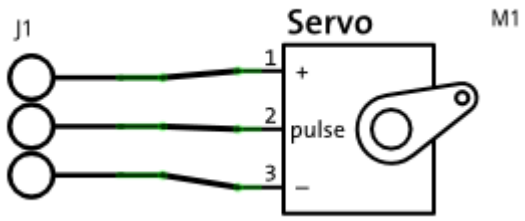


Figura 3.18. Diagrama esquemático de la etapa del servomotor

3.2.7. Diseño de la etapa del motor paso a paso

Para el diseño de la etapa del motor paso a paso, se tomó en cuenta los siguientes parámetros:

- Motor unipolar o bipolar
- Número de pasos para el giro de 360°
- Alimentación del motor

Los motores pap pueden ser unipolar y bipolar y su diferencia radica en que los unipolares generan su giro energizando el cable común y enviando secuencialmente a tierra el resto de cables; en cambio los bipolares requieren

de un cambio de flujo de corriente a través de las bobinas para generar su giro.

Los motores pap para generar el giro de 360° lo hacen a través de pasos, a mayor número de pasos, mayor es la precisión; en la tabla 3.6 se puede observar los pasos con relación a los grados de giro.

Grados que gira por impulso (°)	Números de pasos para llegar a 360°
0,72	500
1,8	200
3,75	96
7,5	48
15	24
90	4

Tabla 3.6. Número de pasos con relación a los grados de giro

De acuerdo a los parámetros mencionados y para el acoplamiento con la placa Arduino, se utilizó un motor pap unipolar de 5 V y de 7.5° de las impresoras Epson.

El consumo de corriente del motor en funcionamiento fue un poco alto y en momentos el giro no era el correcto, por lo cual fue necesario la implementación de un circuito de control, para este tipo de aplicaciones es recomendable el uso del buffer ULN2003A (ver figura 3.19, extraída del Anexo 7), ya que maneja cargas de hasta 500 mA.

El ULN2003A posee internamente 7 circuitos darlington, además es indispensable conectar la tierra al pin 8 y 5 V al pin 9 (pin de los cátodos comunes).

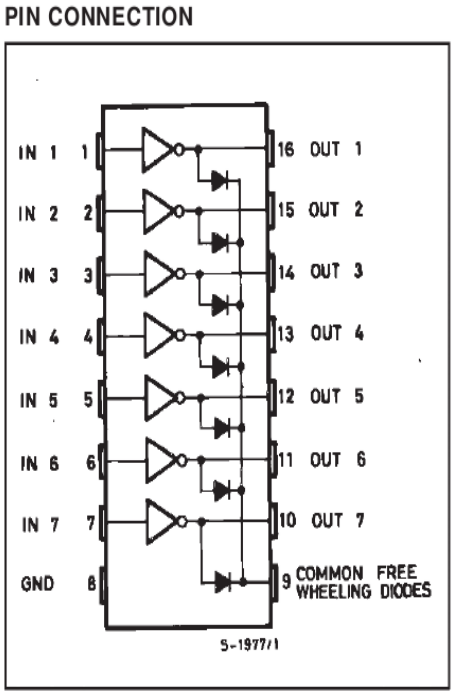


Figura 3.19. Esquema del ULN2003A

El motor pap utilizado es de 6 hilos, 2 de los cuales son comunes y los otro 4 controlan el giro, estos 4 hilos están conectados a las salidas del ULN2003A para proveer mayor corriente, como se ve en el diagrama esquemático de la etapa del motor paso a paso de la figura 3.20.

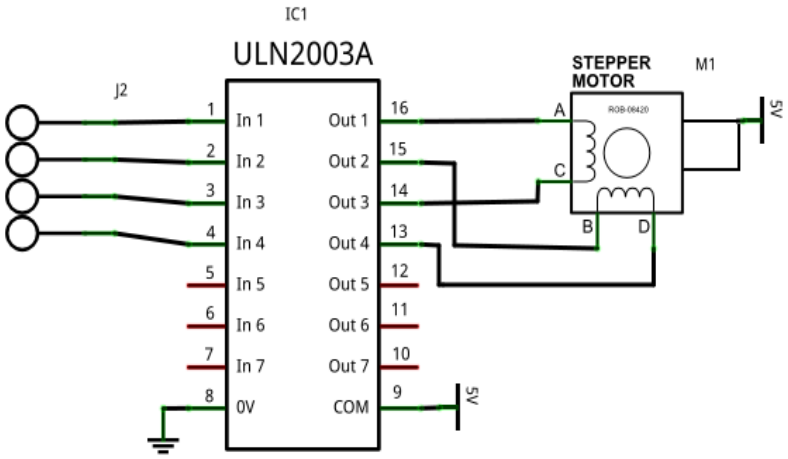


Figura 3.20. Diagrama esquemático de la etapa del motor paso a paso

3.2.8. Diseño de la etapa de los potenciómetros

El potenciómetro es utilizado como un elemento analógico de entrada, se utiliza el de un valor de 10 KΩ que al conectar a Vcc y variar la resistencia se tiene un divisor de voltaje, voltaje que la placa Arduino recibe e interpreta el valor de voltios a bits en el rango de 0 a 1023.

En la figura 3.21 se observa el diagrama esquemático de la etapa de los potenciómetros, en el cual se tiene 2 potenciómetros para la utilización en las prácticas que se necesite más de una entrada analógica.

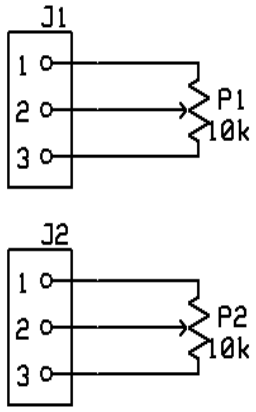


Figura 3.21. Diagrama esquemático de la etapa de los potenciómetros

3.2.9 Diseño de la etapa del parlante

Las salidas PWM del Arduino simulan a salidas analógicas y pueden ser usadas tanto en LEDs variando la intensidad, en servomotores generando señales de pulso, en parlantes generando frecuencias para obtener sonido, etc. En el caso del parlante puede funcionar como salida analógica (melodías) o como digital (tono), por eso se incluyó en el diseño el parlante por permitir la utilización de salidas tanto digitales y analógicas.

En la figura 3.22 se observa el diagrama esquemático de la etapa del parlante.

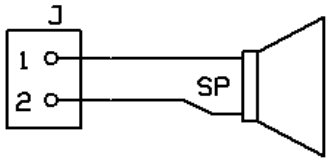


Figura 3.22. Diagrama esquemático de la etapa del parlante

3.3. Implementación del módulo de entrenamiento sobre plataforma Arduino

Con los diagramas esquemáticos diseñados de cada etapa se obtuvo el diseño general del módulo de entrenamiento (figura 3.23).

Para la implementación de las etapas en la placa PCB, se tomó en cuenta que los motores, potenciómetros y el parlante ocupan mucho espacio, por lo que fue necesario utilizar conectores externos para facilitar la ubicación de cada elemento.

Al convertir el diagrama esquemático de todas las etapas en un modelo PCB varias pistas de cobre no podían ser completadas por la gran cantidad de conexiones, por lo cual se utilizó conectores externos para reducir el numero de pistas, con esto se logró la conectividad total en toda la PCB. Además para todas las terminales de conexión se incluyó conectores dobles en el caso de que alguno se dañe se tiene otro de respaldo.

También se enumeró todos los conectores con la nomenclatura **N (N1, N2, N3, ... Nn)** con el fin de identificar todos los elementos, tanto en tipo, valor y cantidad; esto se observa en la figura 3.24, esta figura es el diagrama esquemático y de conectores del módulo de entrenamiento

La placa PCB utilizada es de 24x20 cm, tamaño que permite la distribución de cada etapa, en la tabla 3.7 se observa las dimensiones de cada una de estas.

Etapa	Ancho (cm)	Alto (cm)
Switch	7	3
LEDs RGB	6.5	6.5
Matriz 8x8	17.3	6.5
LCD	10	5.8
Temperatura	4.5	2.3
Sensor infrarrojo	7	3
Servomotor	5	1.8
Motor paso a paso	5	4
Parlante	4.5	1.2
Potenciómetro	4.5	1.5

Tabla 3.7. Dimensiones de las etapas en el PCB

Con la distribución y los esquemas necesarios se realizó el diagrama del circuito impreso de la PCB (figura 3.25). Con este diagrama a través del software de diseño se puede ver la distribución de los elementos electrónicos, debiéndose tener en cuenta que por disposición de espacios la matriz y la pantalla LCD fueron reemplazados por conectores externos (figura 3.26)

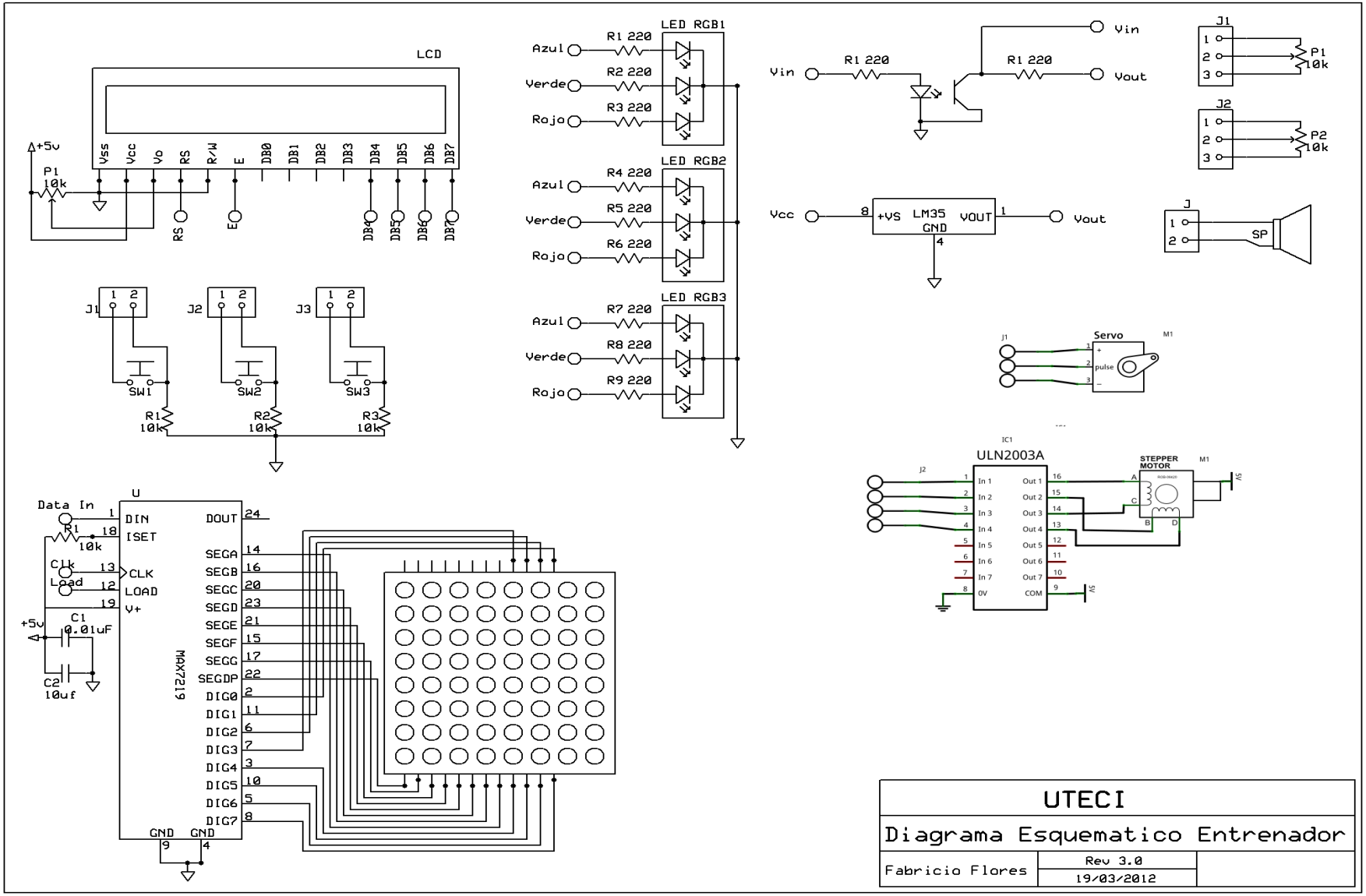


Figura 3.23. Diagrama esquemático del módulos de entrenamiento

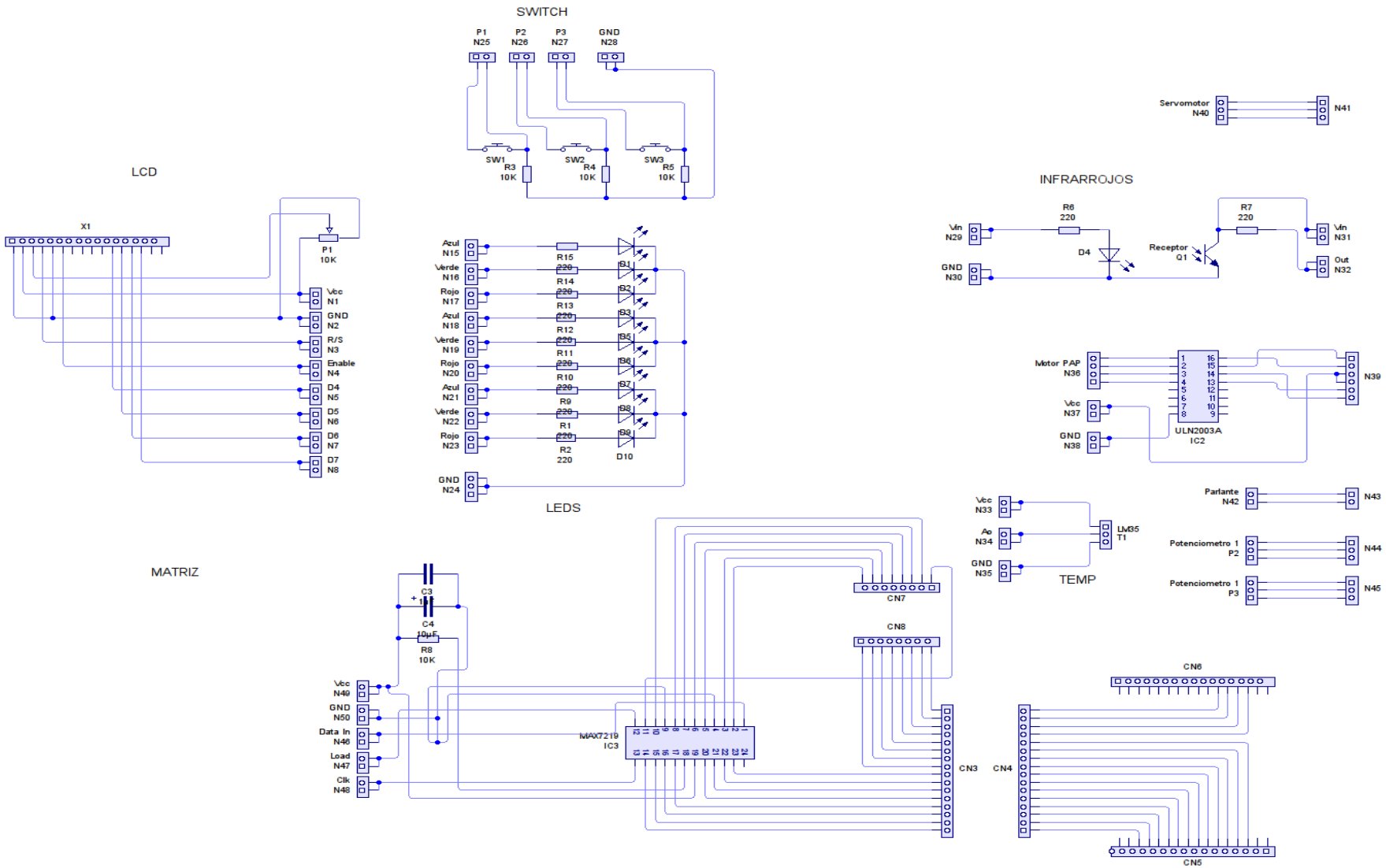
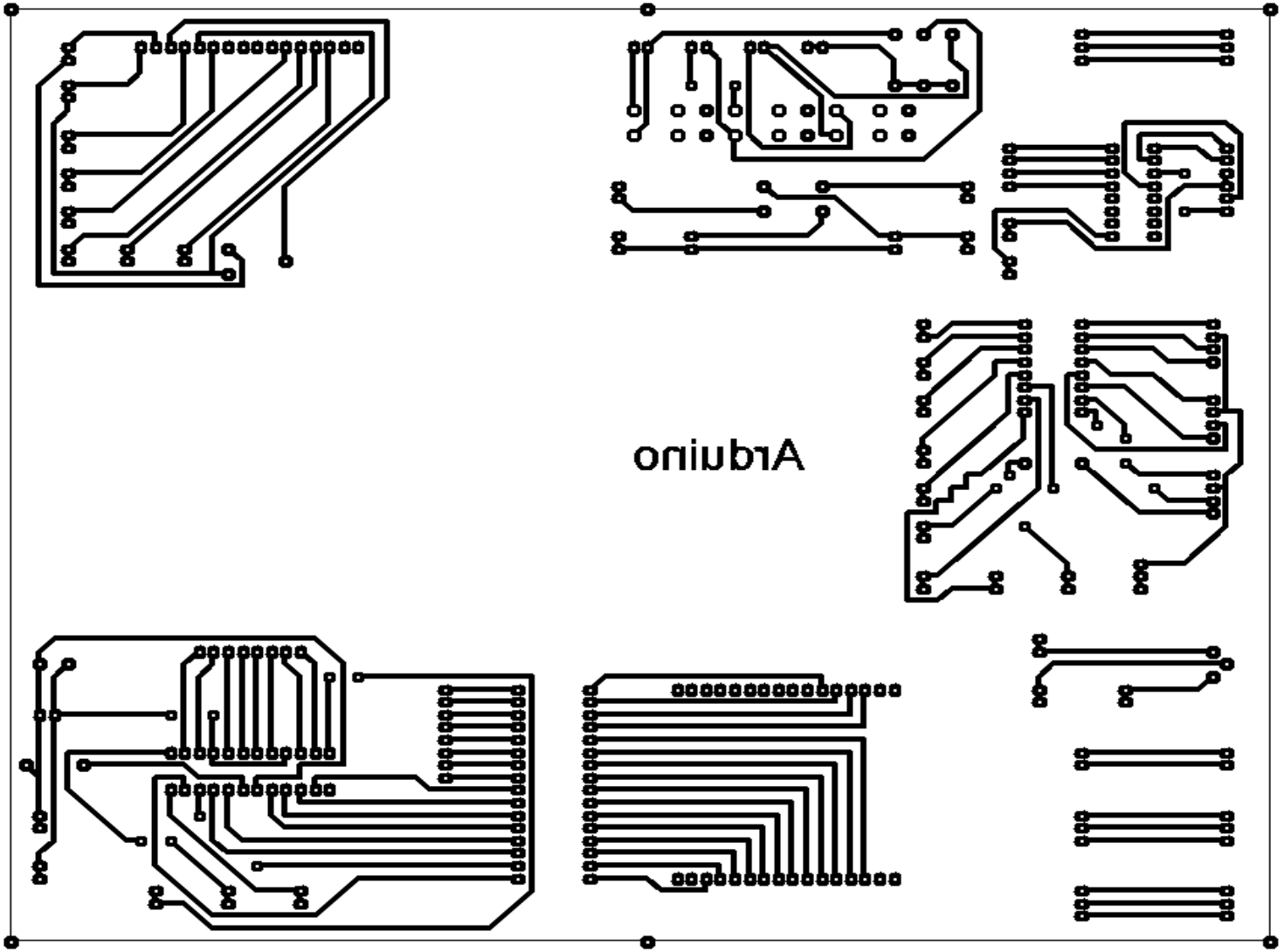
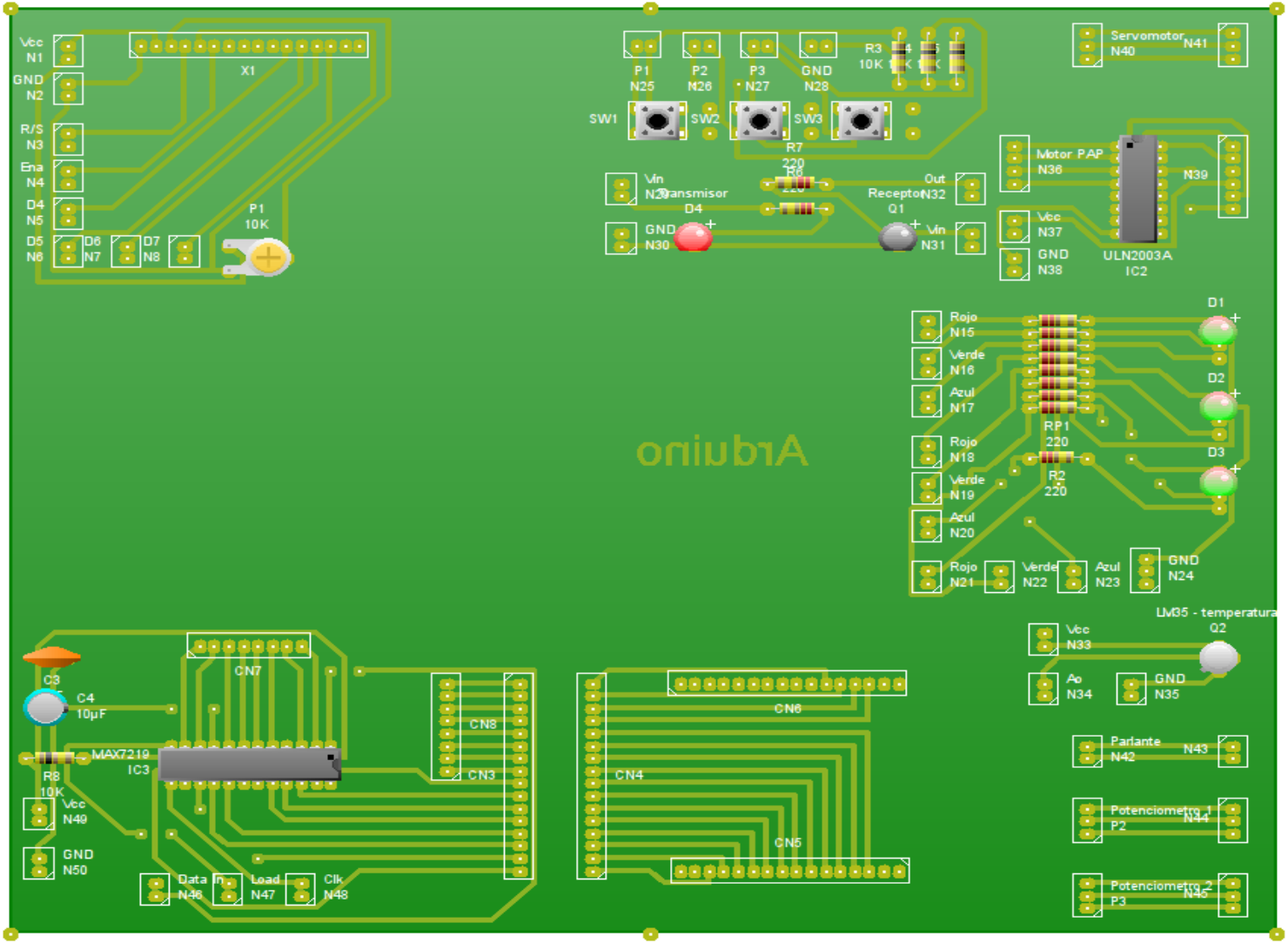


Figura 3.24. Diagrama de conectores del módulo de entrenamiento



Escala 1 : 1.5

Figura 3.25. Diagrama del circuito impreso del PCB



Escala 1 : 1.5

Figura 3.26. Diagrama de distribución de los elementos electrónicos

3.3.1. Elementos electrónicos utilizados por las etapas del diseño

Etapa de los switchs

- 3 Switchs
- 3 Resistencias de 10 KΩ
- 4 Conectores de 2 pines

Etapa de los LEDs RGB

- 3 LEDs RGB
- 9 Resistencias de 220 Ω
- 9 Conectores de 2 pines
- 1 Conector de 3 pines

Etapa de la matriz de LED de 8x8

- 1 Matriz de 8x8 bicolor
- 1 Circuito integrado MAX7219
- 1 Resistencia de 10 KΩ
- 1 Capacitor electrolítico de 10 µF
- 1 Capacitor cerámico de 1 nF
- 2 Conectores externos de 16 pines
- 2 Conectores externos de 8 pines
- 5 Conectores de 2 pines

Etapa de la pantalla LCD

- 1 Pantalla LCD de 16x2 sin Backlight

- 1 Potenciómetro de 10 KΩ
- 8 Conectores de 2 pines

Etapa del sensor de temperatura

- 1 Termistor (LM35)
- 3 Conectores de 2 pines

Etapa del sensor infrarrojo (Tx – Rx)

- 1 LED IR
- 1 Fototransistor
- 2 Resistencias de 220 Ω
- 4 Conectores de 2 pines

Etapa del servomotor

- 1 Servomotor
- 2 Conectores de 3 pines

Etapa del motor PAP

- 1 Motor PAP
- 1 Circuito Integrado ULN2003A
- 1 Conector de 6 pines
- 1 Conector de 4 pines
- 2 Conectores de 2 pines

Etapa de los potenciómetros

- 2 Potenciómetros de 10KΩ
- 4 Conectores de 3 pines

Etapa del parlante

- 1 Parlante de 8 Ω
- 2 Conectores de 2 pines

3.3.2. Elaboración de la PCB y montaje de los componentes

Con el diagrama del circuito impreso del PCB, se imprime en una impresora láser en papel transfer o térmico tanto el diagrama como el diseño de nombres de los componentes (screen de elementos). A través de temperatura se adhiere la impresión del diagrama de pistas del circuito a la baquelita para proceder a quemarla por medios químicos, en este caso con ácido férrico, el resultado se observa en la figura 3.27.

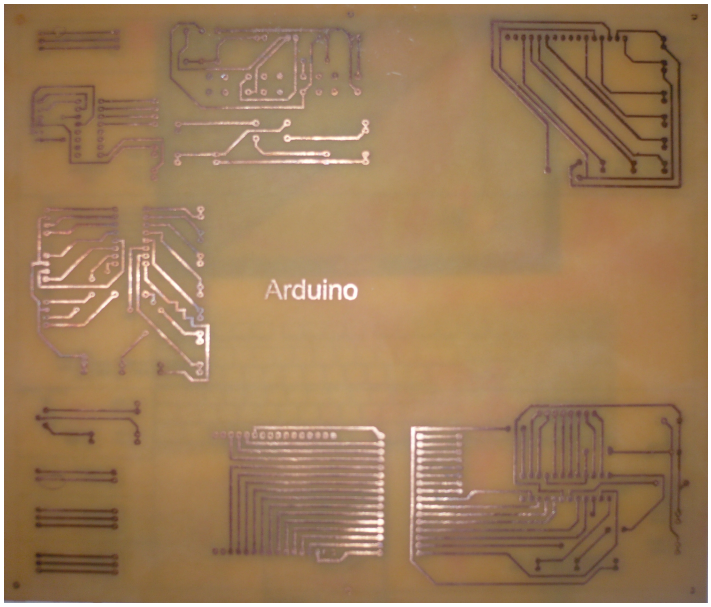


Figura 3.27. Placa PCB impresa el diagrama esquemática

Disuelto el cobre en el ácido se limpia la placa y se retira con materiales ásperos, como la esponja lavaplatos, el barniz de la baquelita, esto con el fin de

adherir de mejor manera la hoja térmica con los nombres de los componentes (figura 3.28). Cuando se imprime la hoja de componentes se lo hace con efecto espejo para invertir los nombres en el papel transfer, una vez adherida con temperatura en la baquelita se ve los nombres y espacios de los componentes en el sentido correcto.

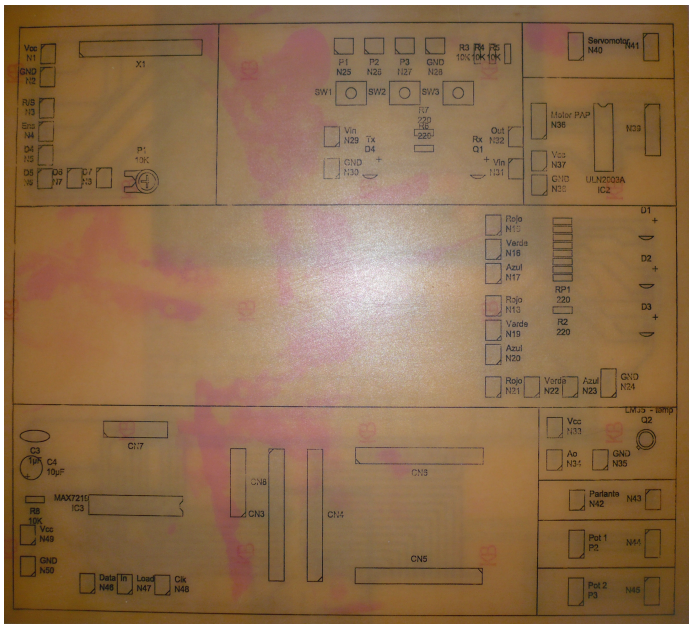


Figura 3.28. Placa PCB impresa nombres de componentes.

Lista la placa PCB, se perfora en la ubicación de los elementos y se realiza agujeros adicionales para la sujeción de la Arduino a la PCB y de ésta a la estructura de madera. Se utiliza separadores metálicos de 10 mm para la separación de placas. Se suelda todos los elementos para obtener el módulo de entrenamiento como se ve en la figura 3.29.

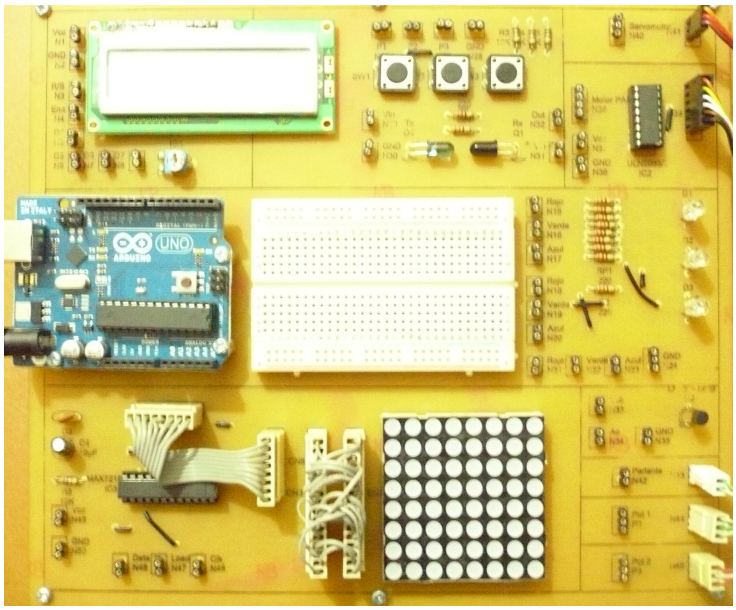


Figura 3.29. Placa PCB del módulo de entrenamiento sobre plataforma Arduino

Como se mencionó anteriormente, los motores, potenciómetros y parlante se ubicaron en un módulo aparte con el fin de mejorar la distribución de todos los elementos. Conectados ambos módulos y sujetados en la estructura de madera se tiene el módulo de entrenamiento sobre plataforma Arduino (figura 3.30).

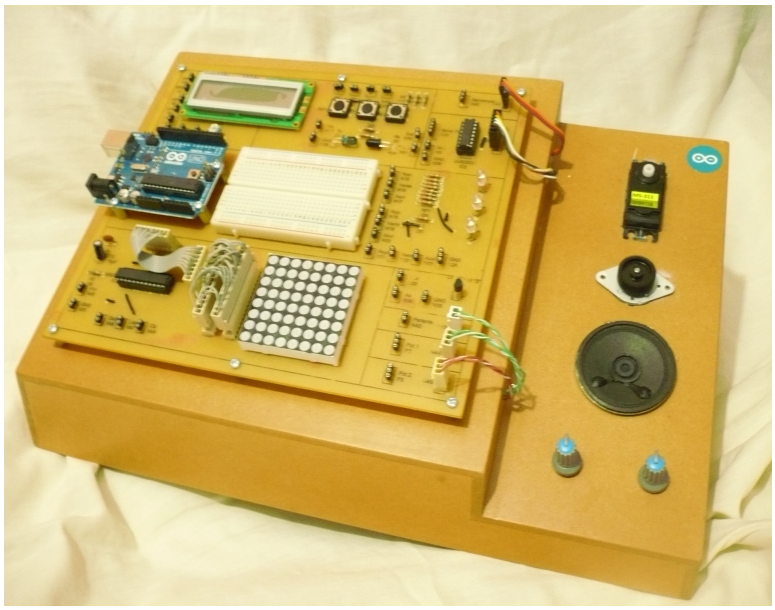


Figura 3.30. Módulo de entrenamiento sobre plataforma Arduino

3.3.3. Nomenclatura y distribución de los componentes en el módulo

El módulo de entrenamiento se encuentra listo para usarse, pero hay que tomar en cuenta las siguientes observaciones:

1. Todas las etapas fueron diseñadas para alimentarse directamente de la placa Arduino, por lo cual cada etapa tiene conectores de Vcc y GND independientes.
2. Todos los conectares poseen numeración y nomenclatura.

Etapa de los switchs

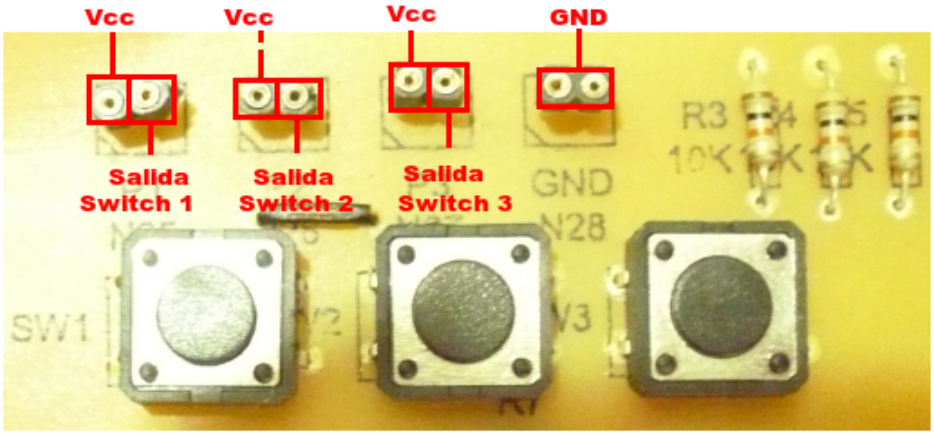


Figura 3.31. Conexión de los pines de la etapa del switch

Descripción de la figura 3.31:

La tierra es común para los 3 switch, pero Vcc es independiente para cada uno. Se tiene un conector con 2 pines, el primero se conecta al Vcc de la placa Arduino y el segundo es la salida del switch.

Etapa del LED RGB

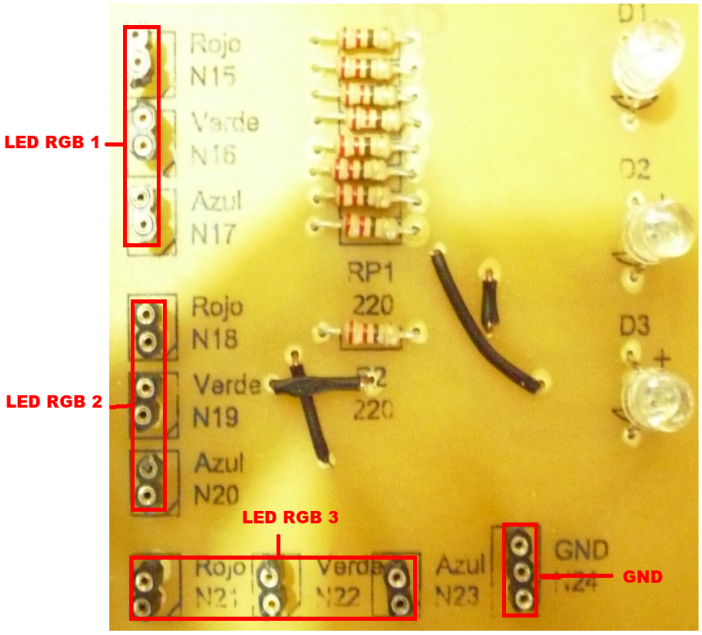


Figura 3.32. Conexión de los pines de la etapa del LED RGB

Descripción de la figura 3.32:

La tierra es común para los 3 LEDs RGB, cada LED tiene 3 conectores de 2 pines cada uno para los colores respectivos.

Etapa de la matriz de LED de 8x8

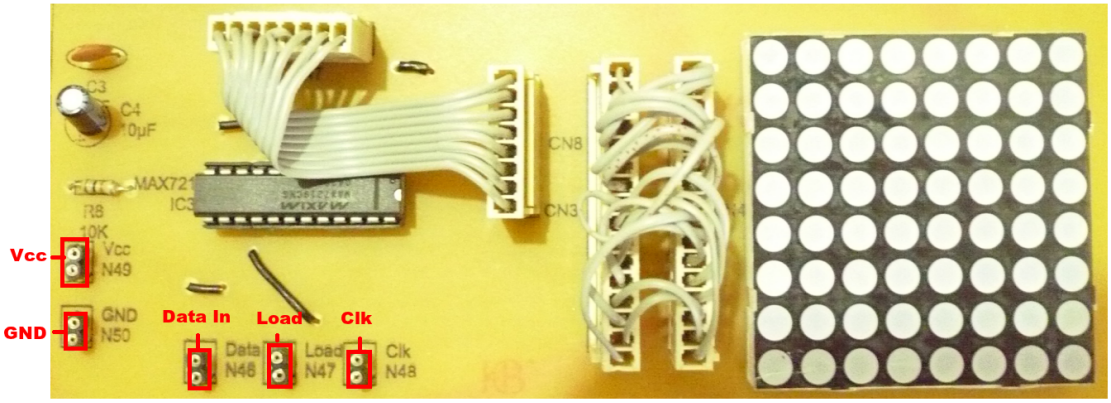


Figura 3.33. Conexión de los pines de la etapa de la matriz

Descripción de la figura 3.33:

Están identificados los conectores de alimentación y los 3 conectores de datos (Data In, Load y Clk).

Etapa de la pantalla LCD

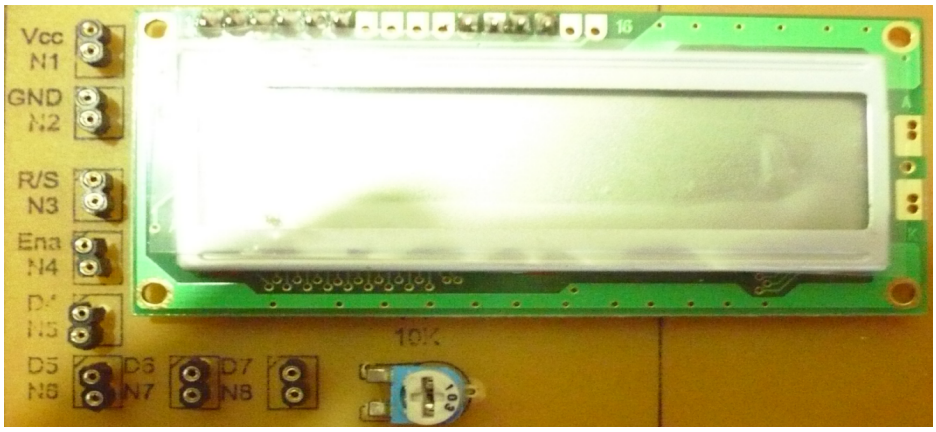


Figura 3.34. Conexión de los pines de la etapa de la pantalla LCD

Descripción de la figura 3.34:

Los conectores están identificados tanto los de alimentación como los de la entrada a la LCD.

Etapa del sensor de temperatura

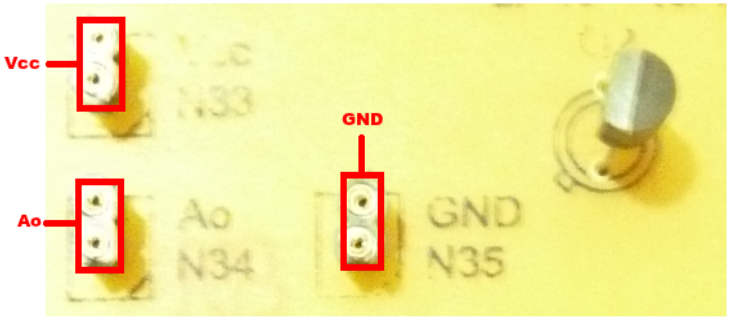


Figura 3.35. Conexión de los pines de la etapa del sensor de temperatura

Descripción de la figura 3.35:

Los conectores de alimentación están identificados (Vcc y GND), el otro conector es el Ao que es el de la salida del sensor.

Etapa del sensor infrarrojo

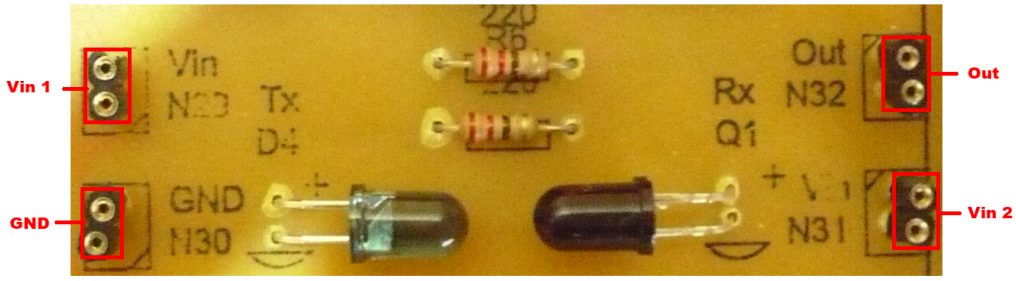


Figura 3.36. Conexión de los pines de la etapa del sensor infrarrojo

Descripción de la figura 3.36:

- GND es común para el Rx y el Tx.
- Vin 1 es el conector en donde ingresa la señal (voltaje) al LED IR.
- Vin 2 es el conector en donde se alimenta el fototransistor.
- OUT es la salida de la señal (voltaje) que se obtiene del fototransistor.

Etapa del servomotor

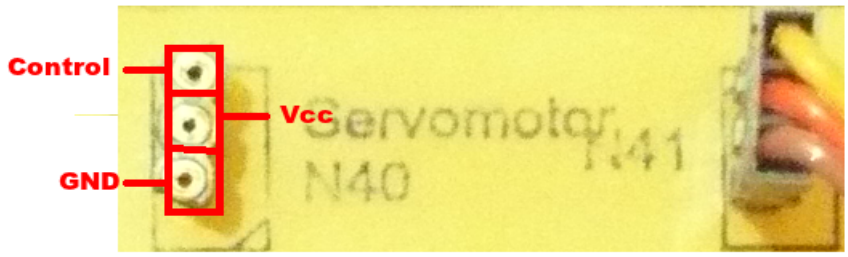


Figura 3.37. Conexión de los pines de la etapa del servomotor

Descripción de la figura 3.37:

Se tiene un conector con 3 pines, el primero pin es el de **Control** que maneja el servomotor el segundo es el de Vcc y el tercero GND.

Etapa del motor PAP

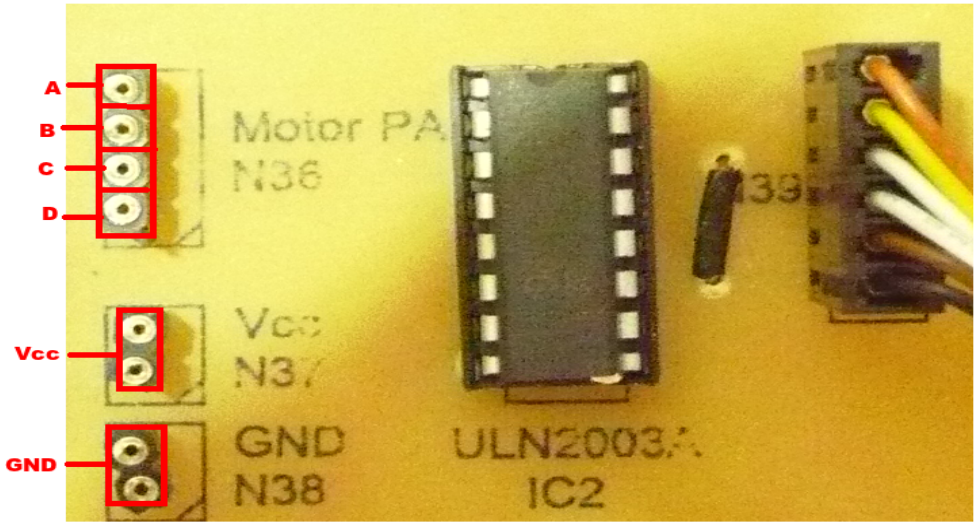


Figura 3.38. Conexión de los pines de la etapa del motor PAP

Descripción de la figura 3.38:

Se tiene los 2 conectores de alimentación (Vcc y GND) y un conector de 4 pines que son las entradas hacia el ULN2003A y de manera recíproca al motor PAP, los pines se encuentran en orden descendente de las bobinas, tal ubicación es A, B,C y D.

Etapa de los potenciómetros

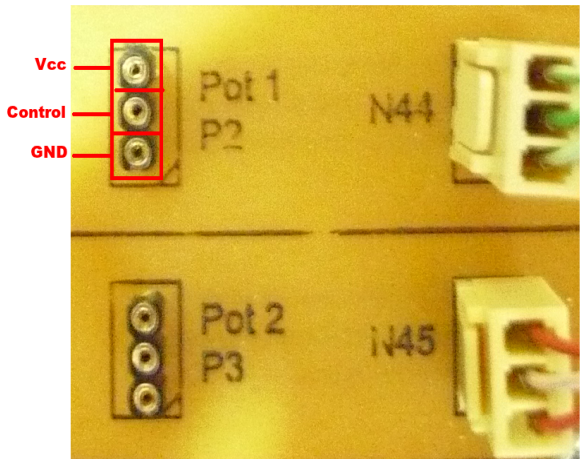


Figura 3.39. Conexión de los pines de la etapa del potenciómetro

Descripción de la figura 3.39:

Los 2 potenciómetros están conectados bajo la misma configuración. En el conector de 3 pines, el primer pin es el de Vcc, el segundo el de control y el tercero de GND.

Etapa del parlante

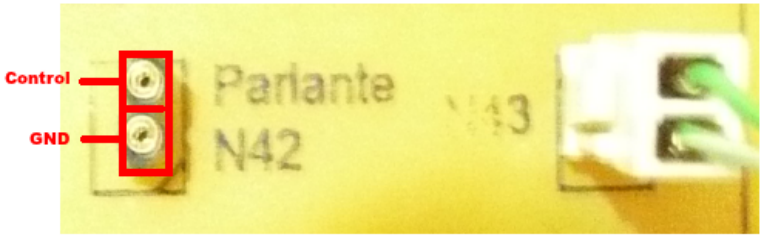


Figura 3.40. Conexión de los pines del parlante

Descripción de la figura 3.40:

Se tiene un conector de 2 pines, el primer pin es el de control y el segundo se conecta a GND.

3.4. Instalación del software Arduino – IDE para Arduino

La plataforma Arduino es compatible con la mayor parte de sistemas operativos (Windows, Mac OS X, GNU/Linux, Android, etc.) y el IDE es sin costo al ser software libre. En el caso de Android la manera de configurar el Arduino es a través del programa ECLIPSE[41] en el cual se puede agregar las librerías de Arduino.

3.4.1. Instalación del IDE en Windows

El IDE puede ser instalado en Windows XP, Vista y 7. En versiones anteriores no es soportado, se puede instalar pero presenta dificultades al compilar.

Para instalar el IDE se realiza los siguientes pasos:

1. Descargar el entorno Arduino de la página web http://arduino.cc/en/Main/Software, en el sitio en inglés se tiene las actualizaciones más recientes, se encuentra disponible la versión 0022 del IDE.
2. Conectar la placa Arduino al puerto USB, en el caso del Vista y el 7 automáticamente reconoce la conexión y extrae los drivers de la descarga realizada. Para XP hay que seguir los siguientes pasos:
    - Al abrirse el diálogo de instalación de Nuevo Hardware, indicar que no se conecte a Windows Update y continuar, (ver figura 3.41).

41 http://developer.android.com/sdk/index.html

Figura 3.41. Asistente para Nuevo Hardware para XP – Paso 1

- Seleccionar “Instalar de una lista o ubicación específica”
- Buscar el programa descargado de la página web de Arduino, y continuar (figura 3.42).

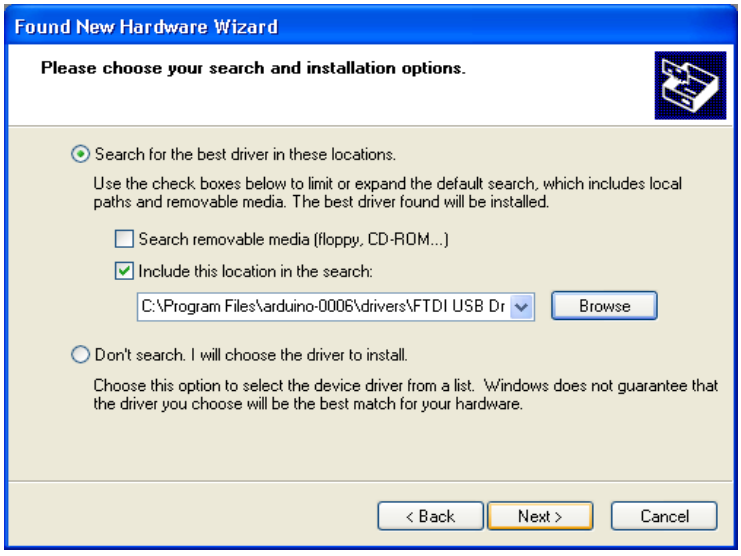


Figura 3.42. Asistente para Nuevo Hardware para XP – Paso 2

- El asistente buscará el driver y encontrará el USB Serial Converter,

dar finalizar y el IDE para Arduino estará instalado (figura 3.43).

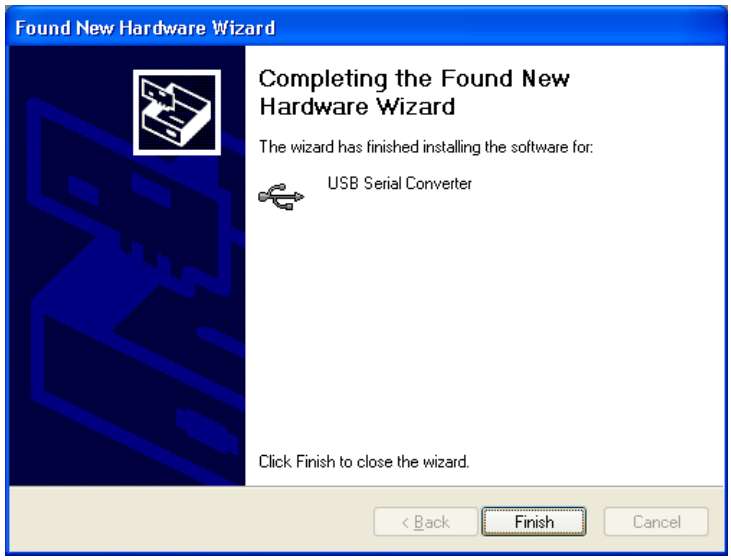


Figura 3.43. Asistente para Nuevo Hardware para XP – Paso 3

3.4.2. Instalación del IDE en Mac OS X

Para la instalación del IDE para Arduino se descarga de la página web http://arduino.cc/en/Main/Software, en esta dirección se encuentra disponible la última versión del software. Al descargar la aplicación se genera una carpeta con 2 controladores como se ve en la figura 3.44.

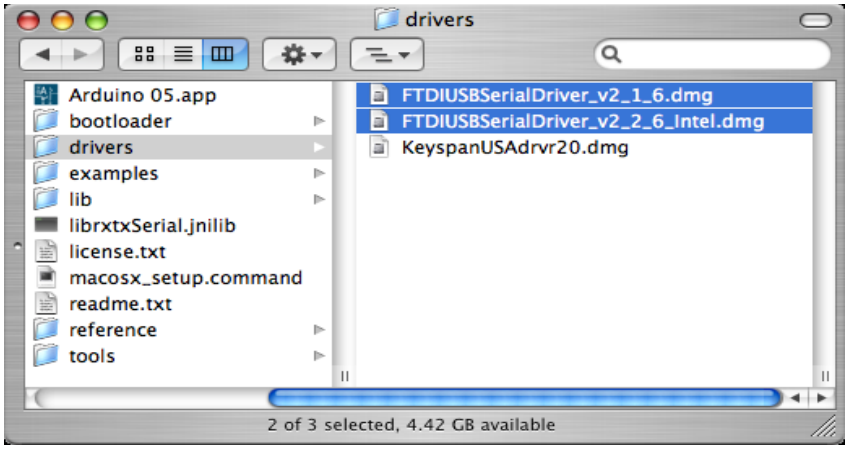


Figura 3.44. Instalación de drivers en Mac OS x

Al conectar el USB de Arduino, automáticamente reconoce los drivers y procede a la instalación.

3.4.3. Instalación del IDE en GNU/Linux

Se puede instalar el IDE en las versiones GNU/Linux con interfaz gráfica que en este caso es Ubuntu 11.10 Oneiric Ocelot, la versión recomendada para usarse es la de 32 bits por ser el sistema estable y recomendado por el servidor de Ubuntu.

Para la instalación del IDE para Arduino existen 2 métodos:

- A través del repositorio de Arduino para Ubuntu.
- A través del centro de software de Ubuntu, para las versiones 10.10 y posteriores.

3.4.3.1. Instalación del IDE a través del repositorio de Arduino[42]

Se siguen los siguientes pasos dentro de la terminal de Ubuntu:

1. <<sudo add-apt-repository ppa:arduino-ubuntu-team>>. Añade el repositorio de Ubuntu a los orígenes del software de la computadora.
2. <<sudo apt-get update>>. Actualiza los orígenes del software de la computadora y por lo tanto de los repositorios.
3. <<sudo apt-get install arduino>>. Instala el IDE para Arduino y todas sus dependencias.

Con esto ya se tiene instalado el software para el Arduino, sólo falta crear el icono en el lanzador para mayor facilidad. El icono se ancla dando clic con el

42 http://cosaspedroruiz.wordpress.com/2010/09/05/instalar-arduino-en-ubuntu-10-04-y-10-10/

botón secundario y escogiendo la opción de “Mantener en el lanzador” (figura 3.45).

Figura 3.45. Icono del IDE en el lanzador

3.4.3.2. Instalación del IDE a través del centro de software de Ubuntu

A partir de la versión de Ubuntu 10.10 se puede descargar el IDE y los drivers a través del centro de software. Para Ubuntu 11.10 se dispone del IDE 0022, que es la última actualización. Para la instalación se siguen los siguientes pasos:

1. Se ingresa al centro de software y se escribe en la búsqueda la palabra Arduino, se despliega la opción de instalación del IDE (ver figura 3.46).

Figura 3.46. IDE para Arduino en el centro de software de Ubuntu

2. Se procede a la instalación, siempre que se va a hacer alguna modificación, Ubuntu solicita la clave de administrador (ver figura 3.47).

Figura 3.47. Autenticación de administrador

Al igual que en la instalación a través de los repositorios de Arduino, falta crear el icono en el lanzador para mayor facilidad, como se ve en la figura 3.45. Al ingresar al IDE se despliega el entorno de Arduino, al conectar la placa a través del USB automáticamente reconoce todos los drivers y corre el programa. Ahora sólo resta configurar la placa a usarse y el puerto asignado.

Para configurar la placa Arduino se selecciona *Tools* > *Board* > Se escoge la placa conectada (figura 3.48).

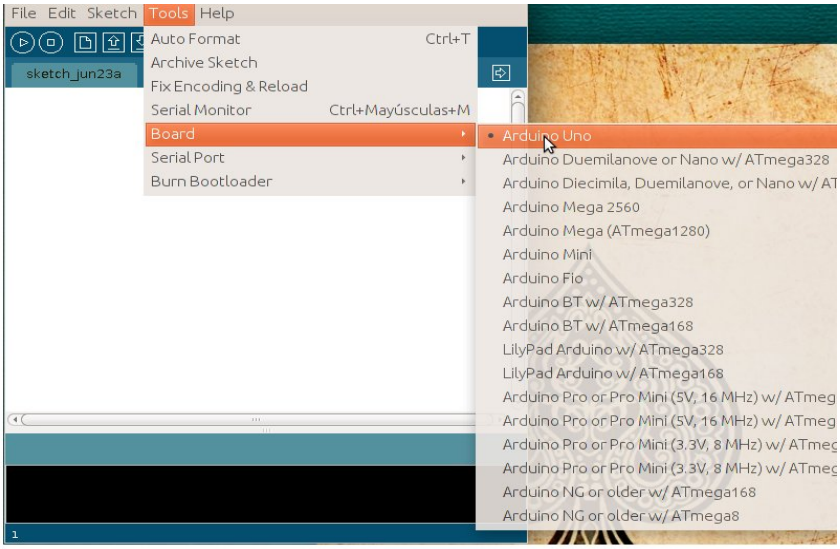


Figura 3.48. Selección de la placa Arduino a través del IDE

Para configurar el puerto serial se selecciona *Tools* > *Serial Port* > Se escoge el puerto serial configurado, que generalmente no es más de uno, para Ubuntu se tiene algo así: dev/tty/USB0 ó dev/tty/S1.

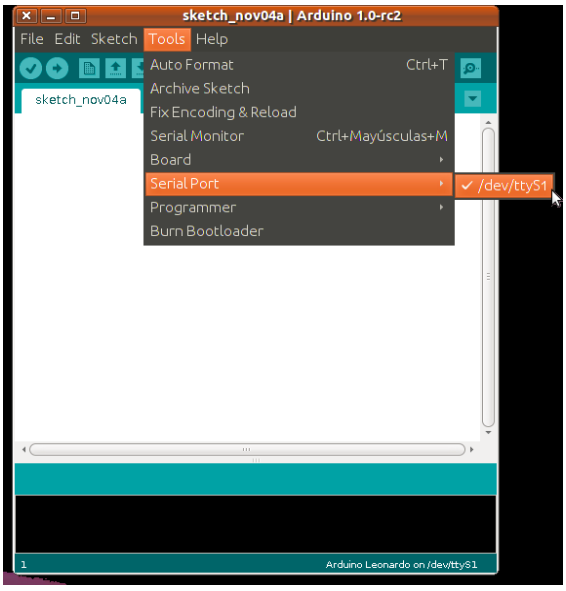


Figura 3.49. Selección del puerto conectado a Arduino

3.5. Pruebas y resultados

Las pruebas del módulo de entrenamiento sobre plataforma Arduino van relacionadas con el diseño de las prácticas, al ser un proyecto orientado a la programación no existe un margen de error en el funcionamiento de cada etapa.

Cada etapa fue comprobada bajo las variables posibles de operación analógica y digital, además al momento de realizar las prácticas se rediseño varias veces, no con el fin de tener el sketch más corto, sino el más entendible.

El proyecto fue planteado para que la programación de la Arduino sea a través de software libre, en este caso Ubuntu 11.10 Oneiric Ocelot, pero en este sistema operativo existieron los siguientes inconvenientes:

- El Ubuntu 11.10 no se encuentra en una versión definitiva, así que existen problemas no frecuentes en el IDE, este suele colgarse, cerrarse o no detectar el puerto USB. Esta situación no se presenta en versiones anteriores del SO (Ubuntu 10.10 Maverick Meerkat).
- En el proceso de instalación del IDE para Arduino se siguen los pasos mencionados en el Tema 3.4.3, cuando llega el momento de conectar la placa reconoce automáticamente el nuevo dispositivo, el problema se presenta al cambiar las placas de diferente diseño (Arduino Uno, Arduino Duemilanove) estas no son reconocidas. Sólo detecta el modelo de la placa conectada por primera vez.

- No permite la creación de librerías, se las puede escribir en el IDE pero al momento de almacenarlas en la carpeta respectiva no se tiene acceso. Este inconveniente no se presenta en Windows.

Los resultados obtenidos en la implementación del módulo de entrenamiento con relación a las prácticas fueron los esperados al no presentar ningún problema en la parte física y en la programación, los resultados fueron los siguientes:

- El voltaje teórico que provee la placa Arduino es de 5 V y en todos los experimentos el valor medido no baja de 4.8 V tanto para entradas y salidas, es decir el rango de error es de un 4%, porcentaje muy bajo que no afecta en ningún momento los resultados finales de las prácticas.
- Las prácticas realizadas fueron 19, que utilizan todos los componentes instalados en el módulo de entrenamiento y no se han visto afectadas por el uso continuo sobre todo en los conectores.
- Se pueden realizar varios experimentos simultáneos sin preocuparse del consumo de la corriente, para evitar problemas de bajo abastecimiento del USB se utilizan adaptadores de pared con una corriente de 650 mA, además se pueden conectar la fuente de energía externa y el USB a la vez, en este caso trabaja con el voltaje proveído por el adaptador y la PC solo sirve para la configuración del Arduino. La placa Arduino posee la protección necesaria cuando se trabaja en este modo por si se presentase alguna sobrecarga proveniente del adaptador.

Capítulo IV

ANÁLISIS FINANCIERO

4.1. Matriz FODA

La matriz FODA refleja los puntos fuertes y débiles del proyecto TTP, en la tabla 4.1 se observa detalladamente.

Fortalezas	**Debilidades**
Arduino es de sencillo manejo y bajo costo, sus placas son de open hardware. Se puede configurar a través software libre. Es una tecnología nueva que garantiza escalabilidad para futuro. Es el primer módulo de entrenamiento diseñado para Arduino.	Resistencia a cambiar del manejo de PIC a Arduino. Los módulos de entrenamiento de PIC tienen un buen mercado, ingresar nuevos módulos con otra tecnología puede ser muy complicado.
Oportunidades	**Amenazas**
PIC no ha desarrollado ningún producto nuevo, ni ha implementado una plataforma original. Arduino se sigue innovando y se tiene actualizaciones constantes en software y hardware. A nivel mundial Arduino es la plataforma más vendida, dando un buen mercado para los módulos de entrenamiento.	La creación de Arduino ha dado cabida a la creación de nuevas plataformas, creciendo la competencia significativamente. Los módulos de entrenamiento son susceptibles a ser copiados en su diseño, por no haber restricciones en este sentido.

Tabla 4.1. Matriz FODA

4.2. Análisis de costo - beneficio

El análisis financiero se realiza de acuerdo a los costos generados debido a la investigación y desarrollo del proyecto, en la tabla 4.2, se observa estos valores y detalles.

Actividad	Detalle	Costo (USD)
Investigación	Se investigó los elementos electrónicos a ser incluidos en el módulo, de acuerdo a los carentes en el laboratorio de la U. Israel y que puedan dar funcionalidad a la Arduino. Tiempo: 6 horas	40
Diseño	El diseño va orientado a las necesidades obtenidas en la investigación. El diseño comprende los diagramas esquemática y de PCB. Tiempo: 20 horas	60
Adquisición de materiales	El costo total de los materiales utilizados para realizar un módulo de entrenamiento se observa en la tabla 4.3, con su respectivo detalle	144.22
Implementación	Con el diseño y los materiales se procede a construir los módulos, con las especificaciones del diseño. Tiempo: 20 horas	60
	TOTAL:	304.22

Tabla 4.2. Costo del desarrollo e implementación del módulo de entrenamiento

Cantidad	Descripción	V. Unit (USD)	V. Total (USD)
PROGRAMADOR			
1	Arduino Uno	28.85	28.85
ETAPA DE SWITCH			
3	Switchs	0.35	1.05
3	Resistencias de 10 KΩ	0.03	0.09
1	Regleta maquinada	1.45	1.45
ETAPA DE LEDS RGB			
3	Leds RGB	1.90	5.70

9	Resistencias de 220 Ω	0.03	0.27
1	Regleta maquinada	1.45	1.45
ETAPA MATRIZ DE LED 8x8			
1	Matriz de led 8x8 bicolor	8.83	8.83
1	MAX7219	10.80	10.80
1	Resistencia de 10 KΩ	0.03	0.03
1	Capacitor de 1 µF	0.10	0.10
1	Capacitor electrolítico de 10 µF	0.25	0.25
1	Conector GP8	0.85	0.85
1	Conector GP16	1.20	1.20
1	Regleta maquinada	1.45	1.45
ETAPA DEL LCD			
1	Pantalla LCD	5.80	5.80
1	Potenciómetro miniatura de 10 KΩ	0.30	0.30
1	Regleta maquinada	1.45	1.45
ETAPA DEL SENSOR DE TEMPERATURA			
1	Termistor LM35	2.50	2.50
1	Regleta maquinada	1.45	1.45
ETAPA SENSOR INFRARROJO			
1	Diodo Infrarrojo (Tx)	0.40	0.40
1	Diodo Infrarrojo (Rx)	0.70	0.70
2	Resistencia de 220 Ω	0.03	0.06
1	Regleta maquinada	1.45	1.45
ETAPA DEL SERVOMOTOR			
1	Servomotor	13.50	13.50
1	Regleta maquinada	1.45	1.45
ETAPA DEL MOTOR PASO A PASO			
1	Motor paso a paso	4.50	4.50
1	Driver ULN2003A	1.80	1.80
1	Zócalo de 16 pines	0.35	0.35
1	Regleta maquinada	1.45	1.45
OTROS			
1	Parlante	1.00	1.00
2	Potenciómetros de 10 KΩ	0.45	0.45
2	Conector GP3	0.40	0.80
1	Conector GP2	0.30	0.30
1	Baquelita A4	4.40	4.40

2	Láminas transferibles	2.80	5.60
3	Fundas de cloruro férrico	0.50	1.50
1	Protoboard	2.50	2.50
9	Separadores de placas metálicos de 10 mm	0.30	2.70
4	Tornillos de 3 mm x 45 mm	0.11	0.44
1	Soporte modular de madera	20.00	20.00
IMPORTACIÓN			15.00
		TOTAL	144.22

Tabla 4.3. Costo de materiales utilizados en el módulo de entrenamiento

Como se observa en las tablas anteriores, el valor total del módulo es de 304.22 dólares americanos, de los cuales 100 dólares son de investigación y diseño, que para los próximos módulos estos valores ya son asumidos por el primero, que según análisis económico, en este valor se encuentra la ganancia.

El proyecto contempla la entrega de 5 módulos de entrenamiento sobre plataforma Arduino, por tal motivo el costo total es el siguiente:

Costo Total = 5 x Costo de materiales + Costo (investigación, diseño e implementación) + Costo Arduino Ethernet Shield

Costo Total = 5 x 144.22 + 160 + 39.95

Costo Total = 721.11 + 160 + 39.95

Costo Total = 921.05 dólares

El valor de 921.05 dolares es el costo del proyecto completo a ser entregado a la Universidad Tecnológica Israel.

Para la comercialización de los módulos de entrenamiento el valor es el obtenido en la tabla 4.2. En el mercado no existen módulos de Arduino, así que

no se tiene una referencia del precio con el cual evaluar el porcentaje del costo y obtener algún parámetro financiero.

Capítulo V

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- La plataforma Arduino permite introducir a la programación y a la electrónica de una manera sencilla y básica a los estudiantes de primeros niveles de la Facultad de Electrónica, ya que no necesitan conocimientos específicos, así cuando reciban la asignatura de microcontroladores ya tendrán nociones de programación, además hay que tener presente de que Arduino no es un reemplazo del PIC, ya que PIC es un microcontrolador y Arduino una plataforma de programación que tiene como parte central un microcontrolador Atmega.

- Los módulos de entrenamiento facilitan el aprendizaje ya que se tiene los elementos físicos y los manuales para la programación, además los módulos implementados no sólo pueden ser usados por Arduino sino por cualquier otro microcontrolador.

- Arduino funciona sin inconvenientes en cualquier sistema operativo, sobre todo este proyecto va encaminado al software libre (Ubuntu) en el cual no se ha tenido problemas significativos y en cierto modo la instalación del IDE es más sencilla que en otros SO.

- Arduino permite una amplia conectividad con diferentes tipos de Shields aumentando la opción de funcionamiento y operatividad.

- Teniendo el conocimiento de programación de Arduino y el alcance que se le puede dar a esta plataforma, se facilita la elaboración de proyectos de TTP de los estudiantes de la Universidad Israel.

5.2. Recomendaciones

- Al momento de realizar las conexiones físicas de la Arduino debe estar desconectada de la PC o de la fuente de alimentación externa para evitar descargas y daños en la placa.

- El módulo de entrenamiento posee 2 sensores (infrarrojos y temperatura), se recomienda realizar prácticas con otros sensores disponibles para comprobar las variables de programación que se le puede dar a la plataforma.

- En la mayoría de casos se ocupó elementos básicos para el entendimiento de la programación, lo aconsejable es ocupar elementos más complejos para aprovechar la capacidad de la plataforma.

- La Arduino Ethernet Shield, es la única Shield fabricada bajo licencia Arduino, pero existen Shields creadas por usuarios o empresas

independientes y soportadas con librerías específicas; se recomienda la implementación de proyectos con estas Shields, que en el acoplamiento y escalabilidad de la Arduino es en donde radica su mayor virtud que le ha dado ventaja sobre otras plataformas.

Bibliografía

1. REYES, Carlos

   2004, Aprenda rápido a programar Microcontroladores PIC

   Gráficas Ayerve C.A., Ecuador

2. ENRÍQUEZ, Rafael

   2009, Guía de Usuario de Arduino

   España

3. EVANS, Brian

   2007, Arduino Notebook: A Beginner's Reference Written

   Estados Unidos

4. SARRIÓ, Juanma

   Arduino + [P5, GRASSHOPEER, FIREFLY]

   España

5. DOMÍNGUEZ, Fernando

   Curso de Microcontroladores PIC

   España

6. MALONEY, Timothy

   1997, Electrónica Industrial Moderna

   Prentice Hall, México

7. RIZZONI, Giorgio

   2002, Principios y aplicaciones de ingeniería eléctrica

   McGraw Hill, Colombia

Referencias de páginas web

1. http://www.arduino.cc/es/
2. http://www.wiring.org.co
3. http://www.processing.org
4. http://es.wikipedia.org/wiki/Arduino
5. http://arduinoarts.com/2011/08/the-arduino-uno-anatomy/
6. http://ayudaelectronica.com/%C2%BFque-son-los-arduino-shields/
7. http://es.wikipedia.org/wiki/Motor_el%C3%A9ctrico
8. http://es.wikipedia.org/wiki/Servomotor
9. http://blog.bricogeek.com/noticias/tutoriales/video-tutorial-arduino-led-rgb-y-pulsadores/
10. http://www.seeedstudio.com/depot/rainbowduino-led-driver-platform-atmega-328-p-371.html
11. http://developer.android.com/sdk/index.html
12. http://www.sherkhan.net/blogs/frikadas/?p=397
13. www.pachube.com

Glosario de términos

- Android: Es un sistema operativo móvil basado en Linux.
- Arduino: Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar.
- Atmel: Es una compañía de semiconductores, fundada en 1984. Su línea de productos incluye microcontroladores.
- Avr-gcc: Es un software desarrollado para configurar herramientas Atmel.
- Baudio: Es una unidad de medida, usada en telecomunicaciones, que representa la cantidad de veces que cambia el estado de una señal en un periodo de tiempo, tanto para señales digitales como para señales analógicas.
- Bit: Es un dígito del sistema de numeración binario.
- Byte: Un byte es la unidad fundamental de datos en los ordenadores personales, un byte son ocho bits contiguos.
- Bootloader: Gestor de arranque del sistema.- Es el programa básico que escucha al puerto serie y así poder descargar programas desde la IDE.
- CAD: Computer-aided design, Diseño asistido por computadora.
- FTDI: Es un semiconductor que transforma la comunicación RS-232 a transmisión serial TTL.

- Fritzing: Es un programa de automatización de diseño electrónico libre que busca ayudar a diseñadores y artistas para que puedan pasar de prototipos (usando, por ejemplo, placas de pruebas) a productos finales.
- IDE: (Integrated Development Environment) Entorno de desarrollo integrado, es un programa informático compuesto por un conjunto de herramientas de programación.
- I²C: Es un bus de comunicaciones en serie.
- Linux: Es un núcleo libre de sistema operativo basado en Unix.
- LCD: Pantalla de cristal líquido.
- LED: (Light-Emitting Diode) Diodo emisor de luz.
- PCB: Printed Circuit Board, Placa de Circuito Impreso.
- PIC: Peripheral Interface Controller, Controlador de Interfases Periféricas
- Processing: Es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java.
- PWM: Modulación por ancho de pulso.
- RS-232: Es una interfaz de transmisión de datos serial.
- Shields: Un shield es una placa impresa que se pueden conectar en la parte superior de la placa Arduino para ampliar sus capacidades, pudiendo ser apilada una encima de la otra.
- Sketch: Programa o rutina escrita en el IDE para Arduino.
- SPI: Comunicación serial sincrónica.

- Ubuntu: Es un sistema operativo mantenido por Canonical y la comunidad de desarrolladores. Utiliza un núcleo Linux, y su origen está basado en Debian.
- USB: El Universal Serial Bus (bus universal en serie USB) es un estándar industrial que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre ordenadores y periféricos y dispositivos electrónicos.
- Wiring: Es un lenguaje de programación de código abierto para microcontroladores.
- Xbee: Son módulos de radio frecuencia que trabajan en la banda de 2.4 GHz con protocolo de comunicación 802.15.4

Anexo 1

LED RGB

深圳市昱申科技有限公司
CHINA YOUNG SUN LED TECHNOLOGY CO., LTD.
TEL: (86) 755-28079401 28079402 28079403 28079404 28079405
FAX: (86) 755-28079407 E-mail: info@100LED.com Web: www.100LED.com

Model No.: YSL-R596CR3G4B5W-F12
RED/GREEN/BLUE Triple Color LED white diffused lens

Applications:

- Moving Message Display
- Full Color Display
- Banking Board
- Score Boards
- Digital Display

LED Chip Absolute Maximum Ratings: (Ta=25℃)

Pararmeter	Symbol	Red	Green	Blue	Unit
Forward curret	I_F	20	20	20	mA
Peak forward current(Duty Cycle=$\frac{1}{10}$,1KHz)	I_{FP}	30	30	30	mA
Reverse voltage (V_R=5V)	I_R	10	10	10	μA
Operating temp	T_{OPR}	-25 - 85	-25 - 85	-25 - 85	℃
Storage temp	T_{STG}	-30-85	-30-85	-30-85	℃
Peak Emission Wavelenength	λP_H	625	520	467.5	nm

* Soldering Bath: not more than 5 seconds @260℃.The bottom ends of the plastic reflector should be at least 2mm above the solder surface

Soldering Iron: not more than 3 seconds @300℃ under 30W

LED Chip Typical Electircal & Optical Chara Qeristics: (Ta=25℃)

ITEMS	Color	Symbol	Condition	Min.	Typ.	Max.	Unit
Forward Voltage	Red	V_F	I_F=20mA	1.8	2.0	2.2	V
	Green			3.0	3.2	3.4	
	Blue			3.0	3.2	3.4	
Luminous Intensity	Red	I_V	I_F=20mA	- - -	- - -	2800	mcd
	Green			- - -	- - -	6500	
	Blue			- - -	- - -	1200	
Wavelenength	Red	$\Delta\lambda$	I_F=20mA	620	623	625	nm
	Green			520	521	522.5	
	Blue			465	466	467.5	
Light Degradation after 1000 hours	Red	-4.68% ~ -8.27%					
	Green	-11.37% ~ -15.30%					
	Blue	-8.23% ~ -16.81%					

Address: 5/F, Building B, Anzhilong Indl., Qinghua East Road., Longhua Town, Shenzhen CHINA. 518109

深 圳 市 昱 申 科 技 有 限 公 司
CHINA YOUNG SUN LED TECHNOLOGY CO., LTD.
TEL: (86) 755-28079401 28079402 28079403 28079404 28079405
FAX: (86) 755-28079407 E-mail: info@100LED.com Web: www.100LED.com

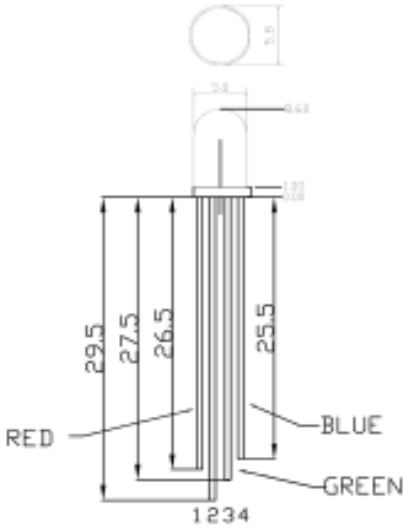
Light Degradation in mcd: (I_F=20mA)

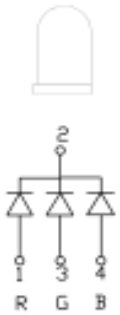
Hours / Colors	Light Degradation in mcd after Different Hours					
	216 Hrs	360 Hrs	792 Hrs	1104 Hrs	1992 Hrs	2328 Hrs
Red	1.52%	-1.22%	-3.10%	-4.68%	-5.72%	-8.27%
Green	-8.02%	-9.78%	-10.25%	-11.37%	-13.79%	-15.30%
Blue	3.13%	-0.33%	-3.84%	-8.23%	-14.32%	-16.81%

Mechanical Dimensions:

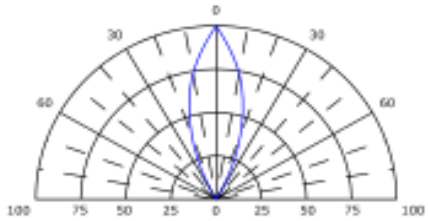
- All dimension are in mm, tolerance is ±0.2mm unless otherwise noted
- An epoxy meniscus may extend about 1.5mm down the leads.
- Burr around bottom of epoxy may be 0.5mm Maximum

Unit: mm





Viewing Angle Drawing



Anexo 2

MAX7219

19-4452; Rev 4; 7/03

MAXIM

Serially Interfaced, 8-Digit LED Display Drivers

General Description

The MAX7219/MAX7221 are compact, serial input/output common-cathode display drivers that interface microprocessors (µPs) to 7-segment numeric LED displays of up to 8 digits, bar-graph displays, or 64 individual LEDs. Included on-chip are a BCD code-B decoder, multiplex scan circuitry, segment and digit drivers, and an 8x8 static RAM that stores each digit. Only one external resistor is required to set the segment current for all LEDs. The MAX7221 is compatible with SPI™, QSPI™, and MICROWIRE™, and has slew-rate-limited segment drivers to reduce EMI.

A convenient 4-wire serial interface connects to all common µPs. Individual digits may be addressed and updated without rewriting the entire display. The MAX7219/MAX7221 also allow the user to select code-B decoding or no-decode for each digit.

The devices include a 150µA low-power shutdown mode, analog and digital brightness control, a scan-limit register that allows the user to display from 1 to 8 digits, and a test mode that forces all LEDs on.

For applications requiring 3V operation or segment blinking, refer to the MAX6951 data sheet.

Applications

Bar-Graph Displays
Industrial Controllers
Panel Meters
LED Matrix Displays

Features

- ♦ 10MHz Serial Interface
- ♦ Individual LED Segment Control
- ♦ Decode/No-Decode Digit Selection
- ♦ 150µA Low-Power Shutdown (Data Retained)
- ♦ Digital and Analog Brightness Control
- ♦ Display Blanked on Power-Up
- ♦ Drive Common-Cathode LED Display
- ♦ Slew-Rate Limited Segment Drivers for Lower EMI (MAX7221)
- ♦ SPI, QSPI, MICROWIRE Serial Interface (MAX7221)
- ♦ 24-Pin DIP and SO Packages

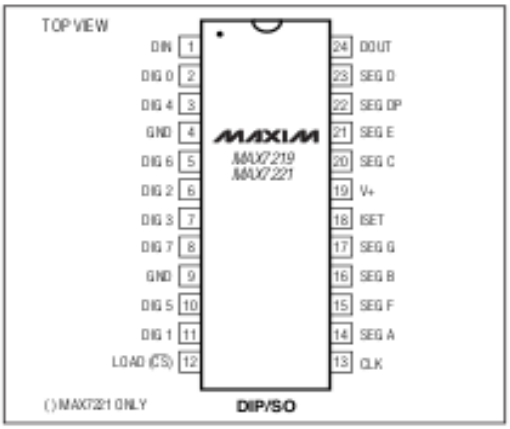
Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX7219CNG	0°C to +70°C	24 Narrow Plastic DIP
MAX7219CWG	0°C to +70°C	24 Wide SO
MAX7219C/D	0°C to +70°C	Dice*
MAX7219ENG	-40°C to +85°C	24 Narrow Plastic DIP
MAX7219EWG	-40°C to +85°C	24 Wide SO
MAX7219ERG	-40°C to +85°C	24 Narrow CERDIP

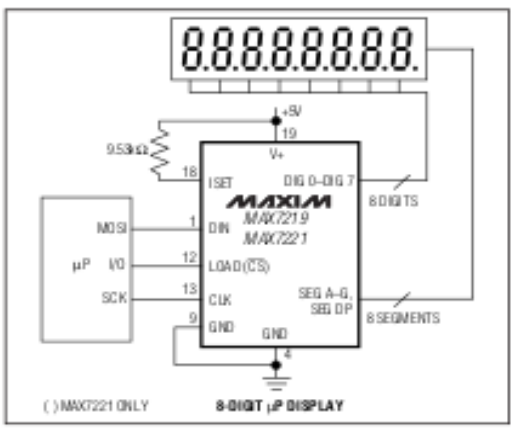
Ordering Information continued at end of data sheet.

**Dice are specified at T_A = +25°C.*

Pin Configuration



Typical Application Circuit



SPI and QSPI are trademarks of Motorola Inc. MICROWIRE is a trademark of National Semiconductor Corp.

Serially Interfaced, 8-Digit LED Display Drivers

ABSOLUTE MAXIMUM RATINGS

Voltage (with respect to GND)
- V+ ..... -0.3V to 6V
- DIN, CLK, LOAD, $\overline{CS}$ ..... -0.3V to 6V
- All Other Pins ..... -0.3V to (V+ + 0.3V)

Current
- DIG 0–DIG 7 Sink Current 500mA
- SEG A–G, DP Source Current 100mA

Continuous Power Dissipation (T_A = +85°C)
- Narrow Plastic DIP (derate 13.3mW/°C above +70°C) 1066mW
- Wide SO (derate 11.8mW/°C above +70°C) 941mW
- Narrow CERDIP (derate 12.5mW/°C above +70°C) 1000mW

Operating Temperature Ranges (T_{MIN} to T_{MAX})
- MAX7219C_G/MAX7221C_G ..... 0°C to +70°C
- MAX7219E_G/MAX7221E_G ..... -40°C to +85°C

Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10s) ..... +300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

(V+ = 5V ±10%, R_{SET} = 9.53kΩ ±1%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Operating Supply Voltage	V+		4.0		5.5	V
Shutdown Supply Current	I+	All digital inputs at V+ or GND, T_A = +25°C			150	µA
Operating Supply Current	I+	R_{SET} = open circuit			8	mA
		All segments and decimal point on, $I_{SEG\_}$ = -40mA		330		
Display Scan Rate	f_{OSC}	8 digits scanned	500	800	1300	Hz
Digit Drive Sink Current	I_{DIGIT}	V+ = 5V, V_{OUT} = 0.65V	320			mA
Segment Drive Source Current	I_{SEG}	T_A = +25°C, V+ = 5V, V_{OUT} = (V+ - 1V)	-30	-40	-45	mA
Segment Current Slew Rate (MAX7221 only)	$\Delta I_{SEG}/\Delta t$	T_A = +25°C, V+ = 5V, V_{OUT} = (V+ - 1V)	10	20	50	mA/µs
Segment Drive Current Matching	ΔI_{SEG}			3.0		%
Digit Drive Leakage (MAX7221 only)	I_{DIGIT}	Digit off, V_{DIGIT} = V+			-10	µA
Segment Drive Leakage (MAX7221 only)	I_{SEG}	Segment off, V_{SEG} = 0V			1	µA
Digit Drive Source Current (MAX7219 only)	I_{DIGIT}	Digit off, V_{DIGIT} = (V+ - 0.3V)	-2			mA
Segment Drive Sink Current (MAX7219 only)	I_{SEG}	Segment off, V_{SEG} = 0.3V	5			mA

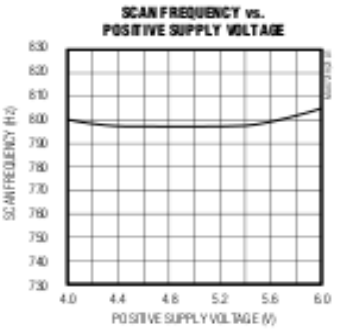
ELECTRICAL CHARACTERISTICS (continued)

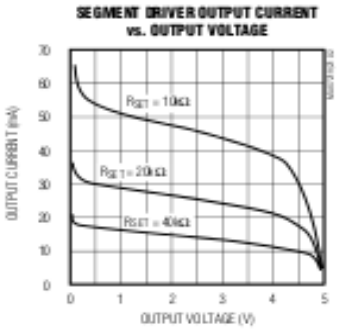
(V+ = 5V ±10%, R_{SET} = 9.53kΩ ±1%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

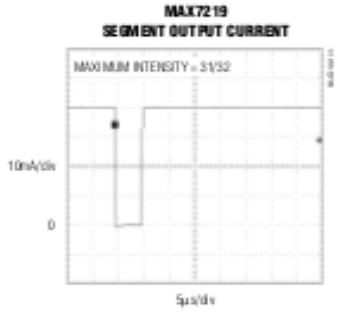
PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
LOGIC INPUTS						
Input Current DIN, CLK, LOAD, $\overline{CS}$	I_{IH}, I_{IL}	V_{IN} = 0V or V+	-1		1	µA
Logic High Input Voltage	V_{IH}		3.5			V
Logic Low Input Voltage	V_{IL}				0.8	V
Output High Voltage	V_{OH}	DOUT, I_{SOURCE} = -1mA	V+ - 1			V
Output Low Voltage	VOL	DOUT, ISINK = 1.6mA			0.4	V
Hysteresis Voltage	ΔV_I	DIN, CLK, LOAD, $\overline{CS}$		1		V
TIMING CHARACTERISTICS						
CLK Clock Period	t_{CP}		100			ns
CLK Pulse Width High	t_{CH}		50			ns
CLK Pulse Width Low	tCL		50			ns
$\overline{CS}$ Fall to SCLK Rise Setup Time (MAX7221 only)	t_{CSS}		25			ns
CLK Rise to $\overline{CS}$ or LOAD Rise Hold Time	tCSH		0			ns
DIN Setup Time	t_{DS}		25			ns
DIN Hold Time	t_{DH}		0			ns
Output Data Propagation Delay	t_{DO}	C_{LOAD} = 50pF			25	ns
Load-Rising Edge to Next Clock Rising Edge (MAX7219 only)	t_{LDCK}		50			ns
Minimum $\overline{CS}$ or LOAD Pulse High	t_{CSW}		50			ns
Data-to-Segment Delay	t_{DSPD}				2.25	ms

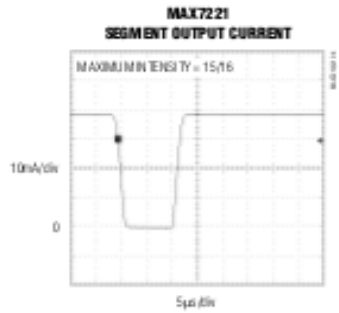
Typical Operating Characteristics

(V+ = +5V, T_A = +25°C, unless otherwise noted.)







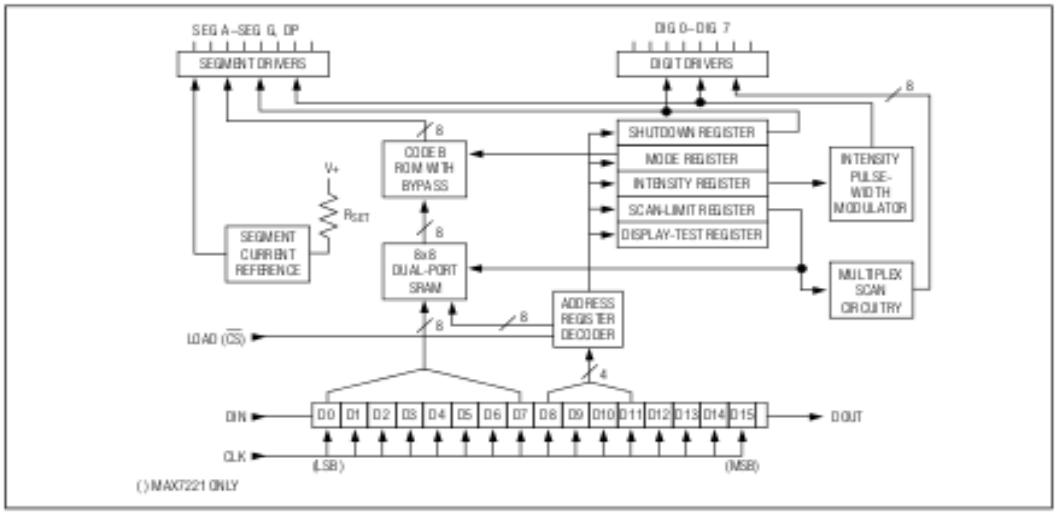


Serially Interfaced, 8-Digit LED Display Drivers

Pin Description

PIN	NAME	FUNCTION
1	DIN	Serial-Data Input. Data is loaded into the internal 16-bit shift register on CLK's rising edge.
2, 3, 5–8, 10, 11	DIG 0–DIG 7	Eight-Digit Drive Lines that sink current from the display common cathode. The MAX7219 pulls the digit outputs to V+ when turned off. The MAX7221's digit drivers are high-impedance when turned off.
4, 9	GND	Ground (both GND pins must be connected)
12	LOAD (MAX7219)	Load-Data Input. The last 16 bits of serial data are latched on LOAD's rising edge.
	$\overline{CS}$ (MAX7221)	Chip-Select Input. Serial data is loaded into the shift register while $\overline{CS}$ is low. The last 16 bits of serial data are latched on $\overline{CS}$'s rising edge.
13	CLK	Serial-Clock Input. 10MHz maximum rate. On CLK's rising edge, data is shifted into the internal shift register. On CLK's falling edge, data is clocked out of DOUT. On the MAX7221, the CLK input is active only while $\overline{CS}$ is low.
14–17, 20–23	SEG A–SEG G, DP	Seven Segment Drives and Decimal Point Drive that source current to the display. On the MAX7219, when a segment driver is turned off it is pulled to GND. The MAX7221 segment drivers are high-impedance when turned off.
18	ISET	Connect to V_{DD} through a resistor (R_{SET}) to set the peak segment current (Refer to *Selecting R_{SET} Resistor and Using External Drivers* section).
19	V+	Positive Supply Voltage. Connect to +5V.
24	DOUT	Serial-Data Output. The data into DIN is valid at DOUT 16.5 clock cycles later. This pin is used to daisy-chain several MAX7219/MAX7221's and is never high-impedance.

Functional Diagram



Anexo 3

MATRIZ

深 圳 市 昱 申 科 技 有 限 公 司
CHINA YOUNG SUN LED TECHNOLOGY CO., LTD.
TEL: (86) 755-28079401 28079402 28079403 28079404 28079405
FAX: (86) 755-28079407 E-mail: info@100LED.com Web: www.100LED.com

Model No.: YSM-2088CR3G2C
8mm Pitch RED/GREEN Double Color Dot Matrix

Applications:

- Moving Message Display
- Full Color Display
- Banking Board
- Score Boards
- Digital Display

LED Chip Absolute Maximum Ratings: (Ta=25℃)

Parameter	Symbol	Red	Green	Unit
Forward curret	I_F	20	20	mA
Peak forward current(Duty Cycle=1/10,1KHz)	I_{FP}	30	30	mA
Reverse voltage (VR=5V)	I_R	10	10	μA
Operating temp	T_{OPR}	-25 - 85	-25 - 85	℃
Storage temp	T_{STG}	-30-85	-30-85	℃
Peak Emission Wavelenength	λ PH	20	30	nm

* Soldering Bath: not more than 5 seconds @260℃.The bottom ends of the plastic reflector should be at least 2mm above the solder surface
Soldering Iron: not more than 3 seconds @300℃ under 30W

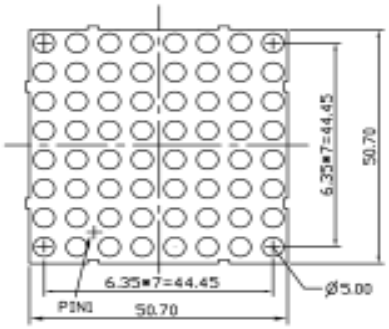
LED Chip Typical Electrical & Optical Characteristics: (Ta=25℃)

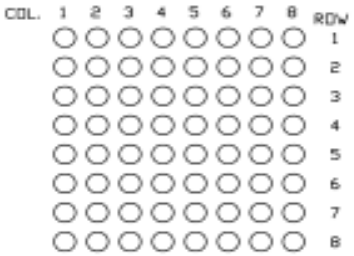
ITEMS	Color	Symbol	Condition	Min.	Typ.	Max.	Unit
Forward Voltage	Red	V_F	I_F=20mA	1.9	2.0	2.2	V
	Green			3.2	3.3	3.5	
Luminous Intensity	Red	I_V	I_F=20mA	200	210	220	mcd
	Green			280	300	330	
Wavelenength	Red	Δλ	I_F=20mA	620	---	625	nm
	Green			515	---	517.5	
Light Degradation after 1000 hours	Red	-4.68% ~ -8.27%					
	Green	-11.37% ~ -15.30%					

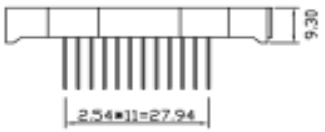
深圳市昱申科技有限公司
CHINA YOUNG SUN LED TECHNOLOGY CO., LTD.
TEL: (86) 755-28079401 28079402 28079403 28079404 28079405
FAX: (86) 755-28079407 E-mail: info@100LED.com Web: www.100LED.com

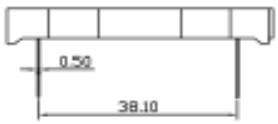
Mechanical Dimensions:

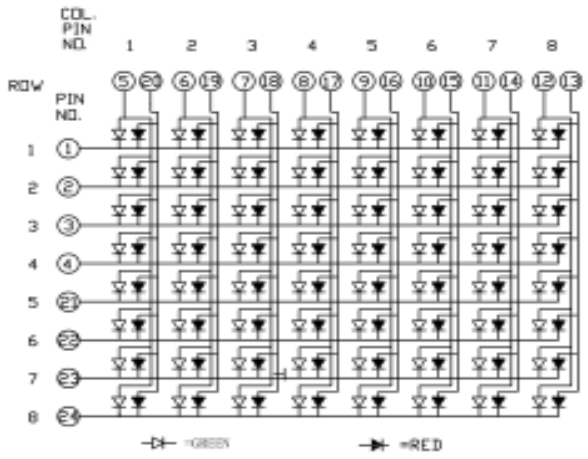
- All dimension are in mm, tolerance is ±0.2mm unless otherwise noted
- An epoxy meniscus may extend about 1.5mm down the leads.











Unit: mm

Address: 5/F, Building B, Anzhilong Indl., Qinghua East Road., Longhua Town, Shenzhen CHINA. 518109

Anexo 4

LCD

VISHAY

LCD-016M002B

Vishay

16 x 2 Character LCD

FEATURES

- 5 x 8 dots with cursor
- Built-in controller (KS 0066 or Equivalent)
- + 5V power supply (Also available for + 3V)
- 1/16 duty cycle
- B/L to be driven by pin 1, pin 2 or pin 15, pin 16 or A,K (LED)
- N.V. optional for + 3V power supply

MECHANICAL DATA		
ITEM	STANDARD VALUE	UNIT
Module Dimension	80.0 x 36.0	mm
Viewing Area	66.0 x 16.0	mm
Dot Size	0.56 x 0.66	mm
Character Size	2.96 x 5.56	mm

ABSOLUTE MAXIMUM RATING					
ITEM	SYMBOL	STANDARD VALUE			UNIT
		MIN.	TYP.	MAX.	
Power Supply	VDD-VSS	- 0.3	–	7.0	V
Input Voltage	VI	- 0.3	–	VDD	V

NOTE: VSS = 0 Volt, VDD = 5.0 Volt

ELECTRICAL SPECIFICATIONS							
ITEM	SYMBOL	CONDITION		STANDARD VALUE			UNIT
				MIN.	TYP.	MAX.	
Input Voltage	VDD	VDD = + 5V		4.7	5.0	5.3	V
		VDD = + 3V		2.7	3.0	5.3	V
Supply Current	IDD	VDD = 5V		–	1.2	3.0	mA
Recommended LC Driving Voltage for Normal Temp. Version Module	VDD - V0	- 20°C		–	–	–	V
		0°C		4.2	4.8	5.1	
		25°C		3.8	4.2	4.6	
		50°C		3.6	4.0	4.4	
		70°C		–	–	–	
LED Forward Voltage	VF	25°C		–	4.2	4.6	V
LED Forward Current	IF	25°C	Array	–	130	260	mA
			Edge	–	20	40	
EL Power Supply Current	IEL	Vel = 110VAC:400Hz		–	–	5.0	mA

DISPLAY CHARACTER ADDRESS CODE:

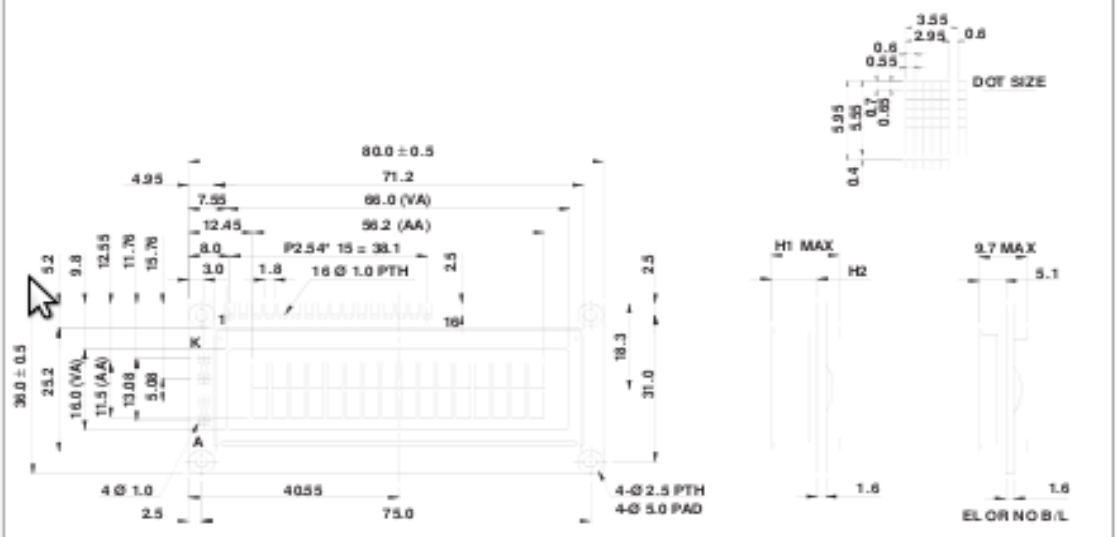
Display Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DD RAM Address	00	01														0F
DD RAM Address	40	41														4F

LCD-016M002B

Vishay — 16 x 2 Character LCD

PIN NUMBER	SYMBOL	FUNCTION
1	Vss	GND
2	Vdd	+ 3V or + 5V
3	Vo	Contrast Adjustment
4	RS	H/L Register Select Signal
5	R/$\overline{W}$	H/L Read/Write Signal
6	E	H →L Enable Signal
7	DB0	H/L Data Bus Line
8	DB1	H/L Data Bus Line
9	DB2	H/L Data Bus Line
10	DB3	H/L Data Bus Line
11	DB4	H/L Data Bus Line
12	DB5	H/L Data Bus Line
13	DB6	H/L Data Bus Line
14	DB7	H/L Data Bus Line
15	A/Vee	+ 4.2V for LED/Negative Voltage Output
16	K	Power Supply for B/L (0V)

DIMENSIONS in millimeters



LED - H/L B/L	HIGH	LOW
H1	13.2	12.1
H2	8.6	7.5

For Technical Questions, Contact: Displays@Vishay.com

Anexo 5

LM35

National Semiconductor

December 1994

LM35/LM35A/LM35C/LM35CA/LM35D Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-202 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load

Connection Diagrams

TO-46
Metal Can Package*

TL/H/5516–1

*Case is connected to negative pin (GND)

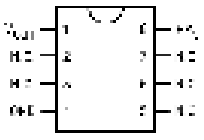
Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

TO-92
Plastic Package

TL/H/5516–3

Order Number LM35CZ, LM35CAZ or LM35DZ
See NS Package Number Z03A

SO-8
Small Outline Molded Package

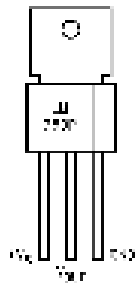


TL/H/5516–21

Top View
N.C. = No Connection

Order Number LM35DM
See NS Package Number M08A

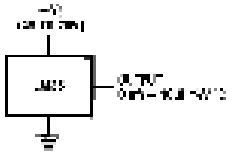
TO-202
Plastic Package



TL/H/5516–24

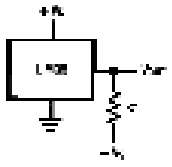
Order Number LM35DP
See NS Package Number P03A

Typical Applications



TL/H/5516–3

FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)



TL/H/5516–4

Choose $R_1 = -V_S/50\ \mu A$

$V_{OUT} = +1,500$ mV at +150°C
$= +250$ mV at +25°C
$= -550$ mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

TRI-STATE® is a registered trademark of National Semiconductor Corporation.

  RRD-B30M75/Printed in U.S.A.

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp., TO-46 Package,	−60°C to +180°C
TO-92 Package,	−60°C to +150°C
SO-8 Package,	−65°C to +150°C
TO-202 Package,	−65°C to +150°C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300°C
TO-92 Package, (Soldering, 10 seconds)	260°C
TO-202 Package, (Soldering, 10 seconds)	+230°C
SO Package (Note 12):	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55°C to +150°C
LM35C, LM35CA	−40°C to +110°C
LM35D	0°C to +100°C

Electrical Characteristics (Note 1) (Note 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	$T_A = +25°C$	±0.2	±0.5		±0.2	±0.5		°C
	$T_A = -10°C$	±0.3			±0.3		±1.0	°C
	$T_A = T_{MAX}$	±0.4	±1.0		±0.4	±1.0		°C
	$T_A = T_{MIN}$	±0.4	±1.0		±0.4		±1.5	°C
Nonlinearity (Note 8)	$T_{MIN} \le T_A \le T_{MAX}$	**±0.18**		**±0.35**	**±0.15**		**±0.3**	°C
Sensor Gain (Average Slope)	$T_{MIN} \le T_A \le T_{MAX}$	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/°C
Load Regulation (Note 3) $0 \le I_L \le 1$ mA	$T_A = +25°C$	±0.4	±1.0		±0.4	±1.0		mV/mA
	$T_{MIN} \le T_A \le T_{MAX}$	**±0.5**		**±3.0**	**±0.5**		**±3.0**	mV/mA
Line Regulation (Note 3)	$T_A = +25°C$	±0.01	±0.05		±0.01	±0.05		mV/V
	$4V \le V_S \le 30V$	**±0.02**		**±0.1**	**±0.02**		**±0.1**	mV/V
Quiescent Current (Note 9)	$V_S = +5V$, $+25°C$	56	67		56	67		µA
	$V_S = +5V$	**105**		**131**	**91**		**114**	µA
	$V_S = +30V$, $+25°C$	56.2	68		56.2	68		µA
	$V_S = +30V$	**105.5**		**133**	**91.5**		**116**	µA
Change of Quiescent Current (Note 3)	$4V \le V_S \le 30V$, $+25°C$	0.2	1.0		0.2	1.0		µA
	$4V \le V_S \le 30V$	**0.5**		**2.0**	**0.5**		**2.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.5**	**+0.39**		**+0.5**	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, $I_L = 0$	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	$T_J = T_{MAX}$, for 1000 hours	±0.08			±0.08			°C

Note 1: Unless otherwise noted, these specifications apply: $-55°C \le T_J \le +150°C$ for the LM35 and LM35A; $-40° \le T_J \le +110°C$ for the LM35C and LM35CA; and $0° \le T_J \le +100°C$ for the LM35D. $V_S = +5$Vdc and $I_{LOAD} = 50$ µA, in the circuit of *Figure 2*. These specifications also apply from +3°C to T_{MAX} in the circuit of *Figure 1*. Specifications in **boldface** apply over the full rated temperature range.

Note 2: Thermal resistance of the TO-46 package is 400°C/W, junction to ambient, and 24°C/W junction to case. Thermal resistance of the TO-92 package is 180°C/W junction to ambient. Thermal resistance of the small outline molded package is 220°C/W junction to ambient. Thermal resistance of the TO-202 package is 65°C/W junction to ambient. For additional thermal resistance information see table in the Applications section.

Electrical Characteristics (Note 1) (Note 6) (Continued)

Parameter	Conditions	LM35			LM35C, LM35D			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy, LM35, LM35C (Note 7)	$T_A = +25°C$	±0.4	±1.0		±0.4	±1.0		°C
	$T_A = -10°C$	±0.5			±0.5		±1.5	°C
	$T_A = T_{MAX}$	±0.8	±1.5		±0.8		±1.5	°C
	$T_A = T_{MIN}$	±0.8		±1.5	±0.8		±2.0	°C
Accuracy, LM35D (Note 7)	$T_A = +25°C$				±0.6	±1.5		°C
	$T_A = T_{MAX}$				±0.9		±2.0	°C
	$T_A = T_{MIN}$				±0.9		±2.0	°C
Nonlinearity (Note 8)	$T_{MIN} \le T_A \le T_{MAX}$	**±0.3**		**±0.5**	**±0.2**		**±0.5**	°C
Sensor Gain (Average Slope)	$T_{MIN} \le T_A \le T_{MAX}$	**+10.0**	**+9.8, +10.2**		**+10.0**		**+9.8, +10.2**	mV/°C
Load Regulation (Note 3) $0 \le I_L \le 1$ mA	$T_A = +25°C$	±0.4	±2.0		±0.4	±2.0		mV/mA
	$T_{MIN} \le T_A \le T_{MAX}$	**±0.5**		**±5.0**	**±0.5**		**±5.0**	mV/mA
Line Regulation (Note 3)	$T_A = +25°C$	±0.01	±0.1		±0.01	±0.1		mV/V
	$4V \le V_S \le 30V$	**±0.02**		**±0.2**	**±0.02**		**±0.2**	mV/V
Quiescent Current (Note 9)	$V_S = +5V$, $+25°C$	56	80		56	80		µA
	$V_S = +5V$	**105**		**158**	**91**		**138**	µA
	$V_S = +30V$, $+25°C$	56.2	82		56.2	82		µA
	$V_S = +30V$	**105.5**		**161**	**91.5**		**141**	µA
Change of Quiescent Current (Note 3)	$4V \le V_S \le 30V$, $+25°C$	0.2	2.0		0.2	2.0		µA
	$4V \le V_S \le 30V$	**0.5**		**3.0**	**0.5**		**3.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.7**	**+0.39**		**+0.7**	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, $I_L = 0$	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	$T_J = T_{MAX}$, for 1000 hours	±0.08			±0.08			°C

Note 3: Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

Note 4: Tested Limits are guaranteed and 100% tested in production.

Note 5: Design Limits are guaranteed (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.

Note 6: Specifications in **boldface** apply over the full rated temperature range.

Note 7: Accuracy is defined as the error between the output voltage and 10mv/°C times the device's case temperature, at specified conditions of voltage, current, and temperature (expressed in °C).

Note 8: Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the device's rated temperature range.

Note 9: Quiescent current is defined in the circuit of *Figure 1*.

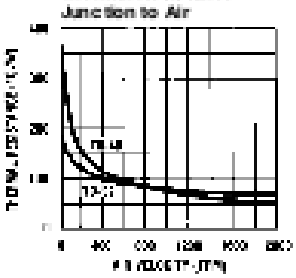
Note 10: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 1.

Note 11: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

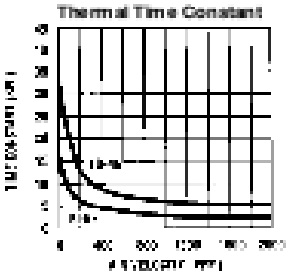
Note 12: See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" or the section titled "Surface Mount" found in a current National Semiconductor Linear Data Book for other methods of soldering surface mount devices.

Typical Performance Characteristics

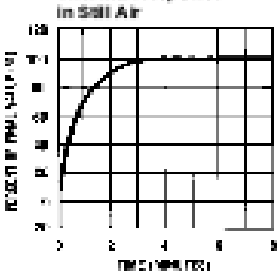
Thermal Resistance Junction to Air



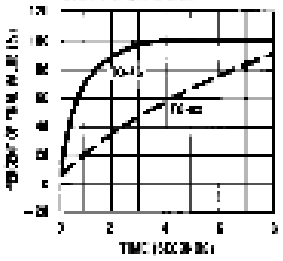
Thermal Time Constant



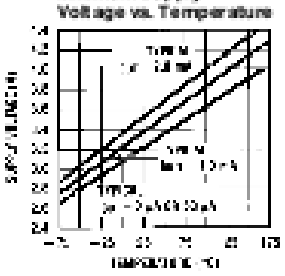
Thermal Response in Still Air



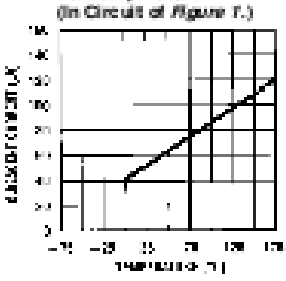
Thermal Response in Stirred Oil Bath



Minimum Supply Voltage vs. Temperature

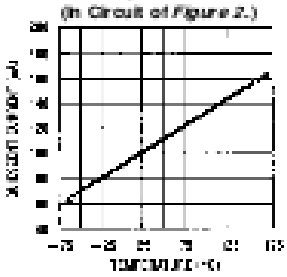


Quiescent Current vs. Temperature (In Circuit of *Figure 1.*)

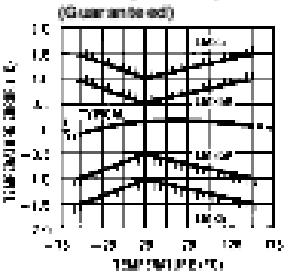


TL/H/5818–17

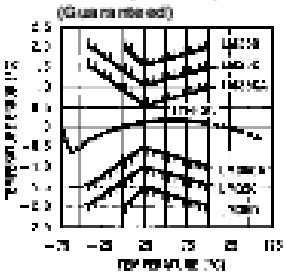
Quiescent Current vs. Temperature (In Circuit of *Figure 2.*)



Accuracy vs. Temperature (Guaranteed)

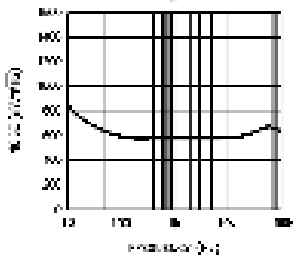


Accuracy vs. Temperature (Guaranteed)

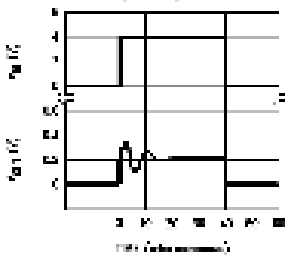


TL/H/5818–18

Noise Voltage



Start-Up Response



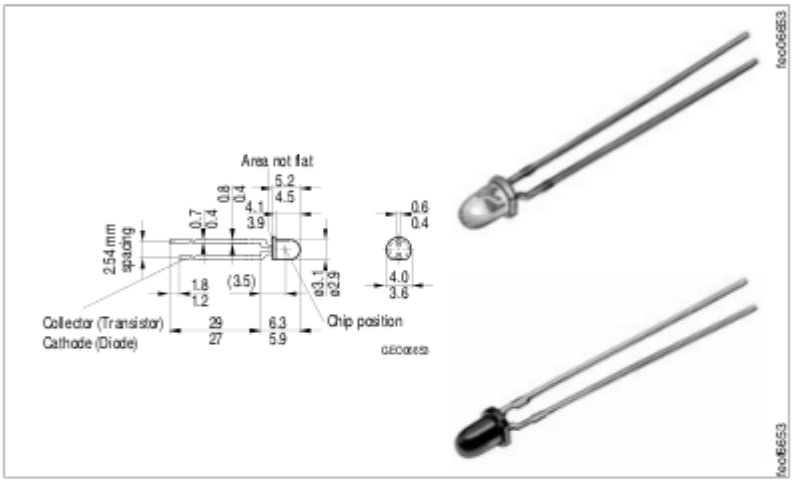
TL/H/5818–22

Anexo 6

INFRARROJO

SIEMENS

NPN-Silizium-Fototransistor **SFH 309**
Silicon NPN Phototransistor **SFH 309 FA**



Maße in mm, wenn nicht anders angegeben/Dimensions in mm, unless otherwise specified

Wesentliche Merkmale

- Speziell geeignet für Anwendungen im Bereich von 380 nm bis 1180 nm (SFH 309) und bei 880 nm (SFH 309 FA)
- Hohe Linearität
- 3 mm-Plastikbauform im LED-Gehäuse
- Gruppiert lieferbar

Anwendungen

- Lichtschranken für Gleich- und Wechsellichtbetrieb
- Industrieelektronik
- "Messen/Steuern/Regeln"

Features

- Especially suitable for applications from 380 nm to 1180 nm (SFH 309) and of 880 nm (SFH 309 FA)
- High linearity
- 3 mm LED plastic package
- Available in groups

Applications

- Photointerrupters
- Industrial electronics
- For control and drive circuits

Typ Type	Bestellnummer Ordering Code	Typ (*vorher) Type (*formerly)	Bestellnummer Ordering Codes
SFH 309	Q62702-P859	SFH 309 FA (*SFH 309 F)	Q62702-P941
SFH 309-3	Q62702-P997	SFH 309 FA-2 (*SFH 309 F-2)	Q62702-P174
SFH 309-4	Q62702-P998	SFH 309 FA-3 (*SFH 309 F-3)	Q62702-P176
SFH 309-5	Q62702-P999	SFH 309 FA-4 (*SFH 309 F-4)	Q62702-P178
SFH 309-6[1)]	Q62702-P1000	SFH 309 FA-5 (*SFH 309 F-5[1)])	Q62702-P180

[1)] Eine Lieferung in dieser Gruppe kann wegen Ausbeuteschwankungen nicht immer sichergestellt werden. Wir behalten uns in diesem Fall die Lieferung einer Ersatzgruppe vor.

[1)] Supplies out of this group cannot always be guaranteed due to unforseeable spread of yield. In this case we will reserve us the right of delivering a substitute group.

Grenzwerte
Maximum Ratings

Bezeichnung Description	Symbol Symbol	Wert Value	Einheit Unit
Betriebs- und Lagertemperatur Operating and storage temperature range	T_{op}; T_{stg}	– 40 ... + 100	°C
Löttemperatur bei Tauchlötung Lötstelle ≥ 2 mm vom Gehäuse, Lötzeit $t \leq 5$ s Dip soldering temperature ≥ 2 mm distance from case bottom, soldering time $t \leq 5$ s	T_S	260	°C
Löttemperatur bei Kolbenlötung Lötstelle ≥ 2 mm vom Gehäuse, Lötzeit $t \leq 3$ s Iron soldering temperature ≥ 2 mm distance from case bottom, soldering time $t \leq 3$ s	T_S	300	°C
Kollektor-Emitterspannung Collector-emitter voltage	V_{CE}	35	V
Kollektorstrom Collector current	I_C	15	mA
Kollektorspitzenstrom, $\tau < 10$ µs Collector surge current	I_{CS}	75	mA

Grenzwerte
Maximum Ratings (cont'd)

Bezeichnung Description	Symbol Symbol	Wert Value	Einheit Unit
Verlustleistung, T_A = 25 °C Total power dissipation	P_{tot}	165	mW
Wärmewiderstand Thermal resistance	R_{thJA}	450	K/W

Kennwerte (T_A = 25 °C, λ = 950 nm)
Characteristics

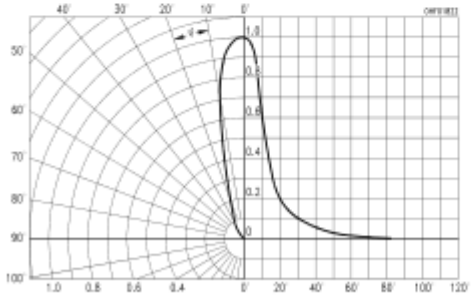
Bezeichnung Description	Symbol Symbol	Wert Value		Einheit Unit
		SFH 309	SFH 309 FA	
Wellenlänge der max. Fotoempfindlichkeit Wavelength of max. sensitivity	$\lambda_{S\ max}$	860	900	nm
Spektraler Bereich der Fotoempfindlichkeit S = 10 % von S_{max} Spectral range of sensitivity S = 10 % of S_{max}	λ	380 ... 1150	730 ... 1120	nm
Bestrahlungsempfindliche Fläche (∅ 240 µm) Radiant sensitive area	A	0.2	0.2	mm^2
Abmessung der Chipfläche Dimensions of chip area	$L \times B$ $L \times W$	0.45 × 0.45	0.45 × 0.45	mm × mm
Abstand Chipoberfläche zu Gehäuseoberfläche Distance chip front to case surface	H	2.4 ... 2.8	2.4 ... 2.8	mm
Halbwinkel Half angle	φ	± 12	± 12	Grad deg.
Kapazität, V_{CE} = 0 V, f = 1 MHz, E = 0 Capacitance	C_{CE}	5.0	5.0	pF
Dunkelstrom Dark current V_{CE} = 25 V, E = 0	I_{CEO}	1 (≤ 200)	1 (≤ 200)	nA

Die Fototransistoren werden nach ihrer Fotoempfindlichkeit gruppiert und mit arabischen Ziffern gekennzeichnet.
The phototransistors are grouped according to their spectral sensitivity and distinguished by arabian figures.

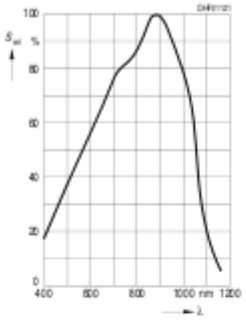
Bezeichnung Description	Symbol Symbol	Wert Value				Einheit Unit
		-2	-3	-4	-5	
Fotostrom, λ = 950 nm Photocurrent E_e = 0.5 mW/cm², V_{CE} = 5 V	I_{PCE}	0.4 ... 0.8	0.63 ... 1.25	1.0 ... 2.0	1.6 ... 3.2	mA
SFH 309: E_v = 1000 lx, Normlicht/ standard light A, V_{CE} = 5 V	I_{PCE}	1.5	2.8	4.5	7.2	mA
Anstiegszeit/Abfallzeit Rise and fall time I_C = 1 mA, V_{CC} = 5 V, R_L = 1 kΩ	t_r, t_f	5	6	7	8	µs
Kollektor-Emitter-Sättigungsspannung Collector-emitter saturation voltage $I_C = I_{PCEmin}$ [1] × 0.3, E_e = 0.5 mW/cm²	V_{CEsat}	200	200	200	200	mV

[1] I_{PCEmin} ist der minimale Fotostrom der jeweiligen Gruppe
[1] I_{PCEmin} is the min. photocurrent of the specified group

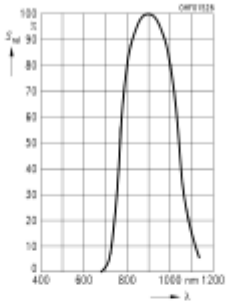
Directional characteristics $S_{rel} = f(\varphi)$



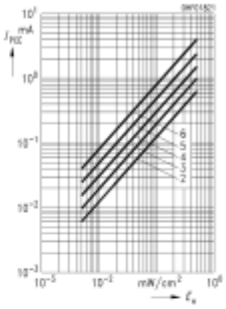
Relative spectral sensitivity,
SFH 309 $S_{rel} = f(\lambda)$



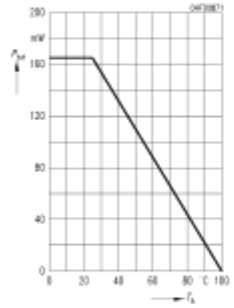
Relative spectral sensitivity,
SFH 309 FA $S_{rel} = f(\lambda)$



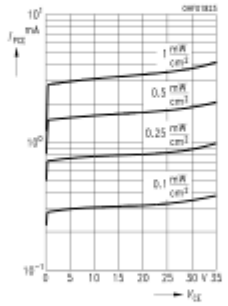
Photocurrent
$I_{PCE} = f(E_e)$, $V_{CE} = 5$ V



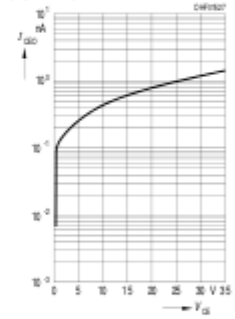
Total power dissipation
$P_{tot} = f(T_A)$



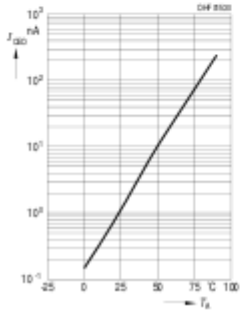
Photocurrent
$I_{PCE} = f(V_{CE})$, E_e = Parameter



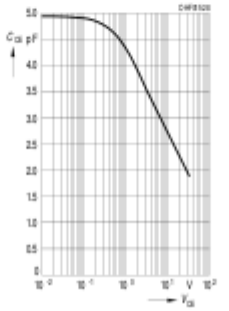
Dark current
$I_{CEO} = f(V_{CE})$, $E = 0$



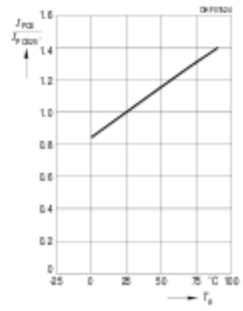
Dark current
$I_{CEO} = f(T_A)$, $V_{CE} = 25$ V, $E = 0$



Capacitance
$C_{CE} = f(V_{CE})$, $f = 1$ MHz, $E = 0$



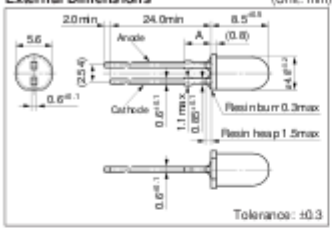
Photocurrent
$I_{PCE}/I_{PCE25°} = f(T_A)$, $V_{CE} = 5$ V



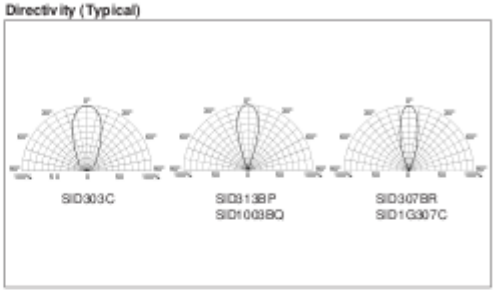
5φ Round Infrared LED

SID300/1003 Series

External Dimensions (Unit: mm)



Directivity (Typical)



Absolute maximum ratings (Ta=25°C)

Symbol	Unit	Rating	Condition
I_F	mA	150	
ΔI_F	mA/°C	−1.33	Above 25°C
I_{FP}	mA	1000	f=1kHz, tw≤10μs
V_R	V	5	
Top	°C	−30 to +85	
Tstg	°C	−30 to +100	

Electrical Optical characteristics (Ta=25°C)

Part Number	Lens color	Forward voltage V_F (V) typ	Forward voltage V_F (V) max	Condition I_F (mA)	Reverse current I_R (μA) max	Condition V_R (V)	Optical Power I_e (mW/sr) typ	Condition	Peak wavelength λP (nm) typ	Condition I_F (mA)	Spectrum half width Δλ (nm) typ	Condition I_F (mA)	Chip material	Dimension A (mm)
SID303C	Clear	1.3	1.5	50	10	5	80	(Constant	940	50	50	50	GaAs	3.0 +0.5
SID313BP	Transparent light purple	1.3	1.5	50	10	5	130	voltage)	940	50	50	50		3.6 +0.5
SID1003BQ	Transparent light navy blue	1.3	1.5	50	10	5	180	V_{CC}=3V	940	50	50	50		
SID307BR	Transparent dark navy blue	1.3	1.5	50	10	5	200	R_L=2.2Ω	940	50	50	50		4.2 +0.5
SID1G307C	Clear	1.5	1.8	50	10	5	50	I_F=50mA	850	50	40	50		

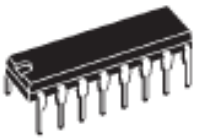
Anexo 7

ULN2003A

ULN2001A-ULN2002A
ULN2003A-ULN2004A

SEVEN DARLINGTON ARRAYS

- SEVEN DARLINGTONS PER PACKAGE
- OUTPUT CURRENT 500mA PER DRIVER (600mA PEAK)
- OUTPUT VOLTAGE 50V
- INTEGRATED SUPPRESSION DIODES FOR INDUCTIVE LOADS
- OUTPUTS CAN BE PARALLELED FOR HIGHER CURRENT
- TTL/CMOS/PMOS/DTL COMPATIBLE INPUTS
- INPUTS PINNED OPPOSITE OUTPUTS TO SIMPLIFY LAYOUT



DIP16

ORDERING NUMBERS: ULN2001A/2A/3A/4A

SO16

ORDERING NUMBERS: ULN2001D/2D/3D/4D

DESCRIPTION

The ULN2001A, ULN2002A, ULN2003 and ULN2004A are high voltage, high current darlington arrays each containing seven open collector darlington pairs with common emitters. Each channel rated at 500mA and can withstand peak currents of 600mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout.

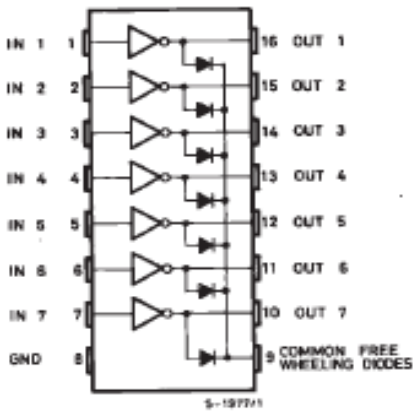
The four versions interface to all common logic families :

ULN2001A	General Purpose, DTL, TTL, PMOS, CMOS
ULN2002A	14-25V PMOS
ULN2003A	5V TTL, CMOS
ULN2004A	6–15V CMOS, PMOS

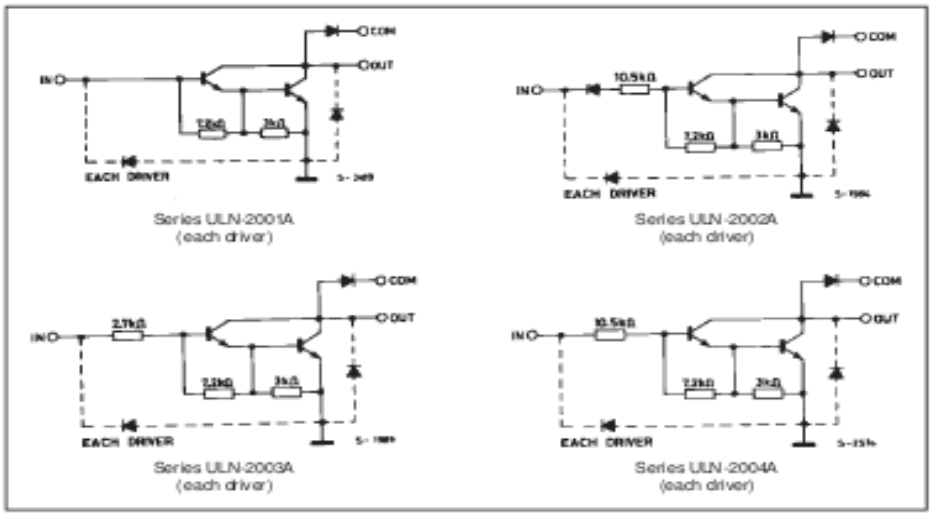
These versatile devices are useful for driving a wide range of loads including solenoids, relays DC motors, LED displays filament lamps, thermal printheads and high power buffers.

The ULN2001A/2002A/2003A and 2004A are supplied in 16 pin plastic DIP packages with a copper leadframe to reduce thermal resistance. They are available also in small outline package (SO-16) as ULN2001D/2002D/2003D/2004D.

PIN CONNECTION



SCHEMATIC DIAGRAM



Series ULN-2001A (each driver)

Series ULN-2002A (each driver)

Series ULN-2003A (each driver)

Series ULN-2004A (each driver)

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_o	Output Voltage	50	V
V_{in}	Input Voltage (for ULN2002A/D - 2003A/D - 2004A/D)	30	V
I_c	Continuous Collector Current	500	mA
I_b	Continuous Base Current	25	mA
T_{amb}	Operating Ambient Temperature Range	– 20 to 85	°C
T_{stg}	Storage Temperature Range	– 55 to 150	°C
T_j	Junction Temperature	150	°C

THERMAL DATA

Symbol	Parameter		DIP16	SO16	Unit
$R_{th\,j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	70	100	°C/W

ELECTRICAL CHARACTERISTICS (T_{amb} = 25°C unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit	Fig.
I_{CEX}	Output Leakage Current	V_{CE} = 50V T_{amb} = 70°C, V_{CE} = 50V T_{amb} = 70°C for ULN2002A V_{CE} = 50V, V_i = 6V for ULN2004A V_{CE} = 50V, V_i = 1V			50 100 500 500	µA µA µA µA	1a 1a 1b 1b
$V_{CE(sat)}$	Collector-emitter Saturation Voltage	I_C = 100mA, I_B = 250µA I_C = 200 mA, I_B = 350µA I_C = 350mA, I_B = 500µA		0.9 1.1 1.3	1.1 1.3 1.6	V V V	2 2 2
$I_{I(on)}$	Input Current	for ULN2002A, V_i = 17V for ULN2003A, V_i = 3.85V for ULN2004A, V_i = 5V V_i = 12V		0.82 0.93 0.35 1	1.25 1.35 0.5 1.45	mA mA mA mA	3 3 3 3
$I_{I(off)}$	Input Current	T_{amb} = 70°C, I_C = 500µA	50	65		µA	4
$V_{I(on)}$	Input Voltage	V_{CE} = 2V for ULN2002A I_C = 300mA for ULN2003A I_C = 200mA I_C = 250mA I_C = 300mA for ULN2004A I_C = 125mA I_C = 200mA I_C = 275mA I_C = 350mA			 13 2.4 2.7 3 5 6 7 8	V	5
h_{FE}	DC Forward Current Gain	for ULN2001A V_{CE} = 2V, I_C = 350mA	1000				2
C_I	Input Capacitance			15	25	pF	
t_{PLH}	Turn-on Delay Time	0.5 V_i to 0.5 V_o		0.25	1	µs	
t_{PHL}	Turn-off Delay Time	0.5 V_i to 0.5 V_o		0.25	1	µs	
I_R	Clamp Diode Leakage Current	V_R = 50V T_{amb} = 70°C, V_R = 50V			50 100	µA µA	6 6
V_F	Clamp Diode Forward Voltage	I_F = 350mA		1.7	2	V	7