UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TEMA:

DESARROLLO DE APLICACIONES PRACTICAS PARA MICROCONTROLADORES ATMEL BAJO LA PLATAFORMA DE ENTRENAMIENTO ARDUINO

Previa la obtención del Título

INGENIERO EN TELECOMUNICACIONES

ELABORADO POR:

Marco Leonardo Robayo Alcocer

Guayaquil, 30 de Agosto del 2014

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por el Sr.

Marco Leonardo Robayo Alcocer como requerimiento parcial para la obtención del título de INGENIERO EN TELECOMUNICACIONES.

Guayaquil, 30 de Agosto del 2014

TUTOR

_____

Mgs. Washington Medina

DIRECTOR DE CARRERA

_____

MsC. Miguel Heras Sànchez

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

INGENIERÍA EN TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

MARCO LEONARDO ROBAYO ALCOCER

DECLARÓ QUE:

El proyecto de tesis denominado “Desarrollo de Aplicaciones prácticas para Microcontroladores ATMEL bajo la plataforma de entrenamiento ARDUINO” ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros conforme las citas que constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Guayaquil, 30 de Agosto del 2014

EL AUTOR

MARCO LEONARDO ROBAYO ALCOCER

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

INGENIERÍA EN TELECOMUNICACIONES

AUTORIZACIÓN

Yo, MARCO LEONARDO ROBAYO ALCOCER

Autorizamos a la Universidad Católica de Santiago de Guayaquil, la publicación, en la biblioteca de la institución del proyecto titulado: “Desarrollo de Aplicaciones prácticas para Microcontroladores ATMEL bajo la plataforma de entrenamiento ARDUINO”, cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Guayaquil, 30 de Agosto del 2014

EL AUTOR

MARCO LEONARDO ROBAYO ALCOCER

DEDICATORIA

A Mis Padres por permitirme realizar como un profesional.

A Mi Hermana por estar brindándome su apoyo y estar pendiente de mi avancé de este implementación.

A Mis Abuelos por luchar contra toda adversidad y verme como un profesional y al resto de mi familia, amigos y profesores que me han compartido sus conocimientos.

EL AUTOR

MARCO LEONARDO ROBAYO ALCOCER

AGRADECIMIENTO

A Mis Padres

A Mis Abuelos Maternos y Paternos

A Mi profesor Ing. Washington medina por ser parte fundamental para la culminación de este proyecto.

EL AUTOR

MARCO LEONARDO ROBAYO ALCOCER

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

CALIFICACIÓN

_____

Mgs. Washington Medina

ÍNDICE GENERAL

Índice de Figuras

Capítulo 2:

Capítulo 3:

Índice de Tablas

Capítulo 2:

RESUMEN

Debido al progreso de los circuitos integrados y los ordenadores, el límite de la disciplina de la ingeniería formal de la Electrónica y la Ingeniería de Telecomunicaciones han tenido su vinculación, tal es así que algunos productos en el mercado están fabricados con componentes que poseen una dependencia mutua e interactiva entre la Electrónica, la Mecánica, Ingeniería Eléctrica y los sistemas automatizados de controles inteligentes, esto ha llevado a que los fabricantes de estos productos complementen en sus actividades a peritos de estas ramas.

Los Microcontroladores y la Programación forman una parte transcendental en el desarrollo de proyectos electrónicos por lo cual la presencia de una nueva tecnología que contribuya con la simplificación y optimización en el desarrollo de dispositivos controladores por medio de la plataforma Arduino basada en lenguaje de programación C y C++ simplificado.

Una plataforma de entrenamiento sobre Arduino, está orientado para el manejo de los alumnos de los primeros niveles de la carrera por su sencillez de programación y conexión, así se adentran a la electrónica tempranamente.

Para proveer las consumaciones de diseños de circuitos electrónicos una opción recomendada es el uso de módulos de entrenamiento, al poseer elementos indispensables para circuitos básicos, reduce los errores de conexión y forja en el alumno la atención en la programación.

El objetivo de este proyecto es profundizar en el diseño e implementación de una plataforma de entrenamiento con diferentes tipos de sensores que son utilizados en la disciplina de la electrónica, sin desatender como primera demanda los fundamentos y concepciones básicos de la tecnología de la misma.

CAPITULO 1: INTRODUCCIÓN DEL TRABAJO DE TITULACIÓN

1.1 Justificación

Es menester que los estudiantes además de adquirir los fundamentos teóricos que lleven al conocimiento de esta nueva tecnología con la finalidad de entender con mejor claridad los diferentes fenómenos que se presentan especialmente con los múltiples tipos de sensores que forman parte integral de este sistema lo lleven a la práctica. El proceso de las prácticas con Arduino hace posible obtener una plataforma de sencillo entendimiento y dar la noción al estudiante para asociarse al universo de los Microcontroladores. Otorgándole al alumno suficiente conocimiento para elaborar proyectos con mayor complejidad y desempeñar de mejor manera su carrera profesional.

1.2 Planteamiento del Problema

En los últimos años en el Ecuador han surgido cambios revolucionarios en cuanto al desarrollo de sistemas electrónicos. Siendo la Universidad Católica Santiago de Guayaquil una de las promotoras a esta transformación la misma que se lleva a cabo en la Facultad de Educación Técnica para el Desarrollo, en donde los estudiantes del nivel básico específico; asumen la teoría pero sin ponerla en práctica, debido a la carencia de materiales físicos. Es allí donde surge la necesidad de llevar a la experiencia, el conocimiento adquirido por medio del diseño e implementación de

una plataforma de entrenamiento dirigida a los educandos y a su adiestramiento en dispositivos electrónicos programables.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar una Plataforma de Entrenamiento Arduino dirigida a la práctica de los estudiantes del nivel básico específico de la Facultad de Educación Técnica para el Desarrollo de la Universidad Católica Santiago de Guayaquil.

1.3.2 Objetivos Específicos

- Utilizar MIKROBASIC PRO for PIC para el desarrollo de la programación del Microcontrolador dentro de la Plataforma de Entrenamiento Arduino.
- Emplear sensores de luz, de temperatura e infrarrojo y generador de tonos para la construcción de la Plataforma de Entrenamiento Arduino.
- Innovar el proceso de aprendizaje en aplicaciones prácticas de los estudiantes del nivel básico específico.

1.4 Tipo de investigación

El presente proyecto se desplegó en los lineamientos de la investigación–acción ya que busca implementar en el proceso de aprendizaje de los estudiantes, una plataforma de entrenamiento Arduino para complementar lo aprendido teóricamente en las aplicaciones de Microcontroladores.

1.5 Metodología

Analítico porque se realizó un razonamiento minucioso de la realidad que viven los estudiantes en su aprendizaje. Además del método bibliográfico para la recopilación de información.

CAPITULO 2: ARDUINO

2.1 Introducción de Arduino.

De acuerdo a Candel N., R. (2013) Arduino es una herramienta tecnológica o plataforma de código abierto (open-source) que se basa en hardware y software flexibles y relativamente fáciles de usar. Mientras que para Toledano M., F. J. (2012) muchos fabricantes generan campañas a favor del uso de Arduino, siendo una valiosa plataforma diseñada para el aprendizaje a nivel secundario y pregrado, inclusive lo puede manejar cualquier novato que podrán implantar objetos interactivos.

En su forma más simple, un Arduino es una pequeña computadora que se puede programar para procesar las entradas y salidas que van desde y hacia el chip. Arduino es lo que se conoce como una plataforma de computación física o incorporada, lo que significa que se trata de un sistema interactivo, que mediante el uso de hardware y software pueden interactuar en um mismo ambiente.

Según Toledano M., F. J. (2012) y Nolivos T., N. T. (2013) arduino consta de un microcontrolador ATmega de Armel en la cual el hardware sigue siendo la herramienta o plataforma microcontroladora para computación física como otras muchas disponibles en el mercado. A diferencia de otras tarjetas de programación o sistemas de entrenamiento de microcontroladores, la tarjeta Arduino son relativamente económicas, y el software es un sistema de código abierto (open source) compatible con los SO’s Windows, Linux y Macintosh.

Vargas M., F. J. (2012) manifiesta que existen algunas versiones de los sistemas de entrenamiento “Arduino”, por ejemplo, tal como la tarjeta muy interesante “Arduino BT” que dispone de un módulo de comunicación bluetooth y que puede programarse sin necesidad de cables. Otro ejemplo, sería la tarjeta “Arduino Mini” que sería la más pequeña de los dispositivos electrónicos, y finalmente la tarjeta “Arduino Serial” que utiliza a la comunicación serial RS232.

Como se mencionó anteriormente, Arduino es una herramienta a tener en cuenta por su versatilidad y bajo coste para ser utilizado en la industria. Es muy útil en aquellas situaciones que requieren controlar sistemas que permiten fabricar un pequeño número de unidades. En esta situación el ingeniero no necesita diseñar electrónicamente la tarjeta de entrenamiento de microcontroladores, pues ya viene diseñada y lista para ejecutar diferentes aplicaciones.

Por ejemplo, un simple uso de Arduino sería convertir una luz encendida durante un período determinado de tiempo, digamos 30s, después de que un botón (pulsador) se ha pulsado. Para dicho ejemplo, suponemos que Arduino tendría una lámpara conectada a ella, así como un pulsador. Arduino espera pacientemente que el pulsador sea presionado, para el encendido de focos e iniciar el conteo. Una vez que cuenta 30s, entonces se apaga la lámpara y el proceso se repite nuevamente esperando a que otro pulsador lo detenga. Se podría utilizar esta configuración por ejemplo, para controlar una lámpara en un armario o debajo de las escaleras.

2.2 Características Básicas de Arduino.

En la figura 2.2 se muestra la tarjeta o placa de Arduino Uno, que se utilizará en el presente proyecto de titulación. Las dimensiones de esta pequeña placa son 74mmx53mm. Para poder programar se necesita de conexión USB en que también se utiliza como fuente de alimentación de 5 V. Si se requiere pueden utilizar la fuente de alimentación externa que es de 9 V.

Para cualquiera de los dos casos, el usuario no tendrá que seleccionar el tipo de alimentación, ya que esta se hace de manera automática. Arduino Uno consta de 14 entradas digitales y 6 analógicas que pueden usarse también como si fueran digitales. Además se puede utilizar para alimentar cualquier circuito montado por ejemplo en protoboard ya sea 5 V o 3.3 V.

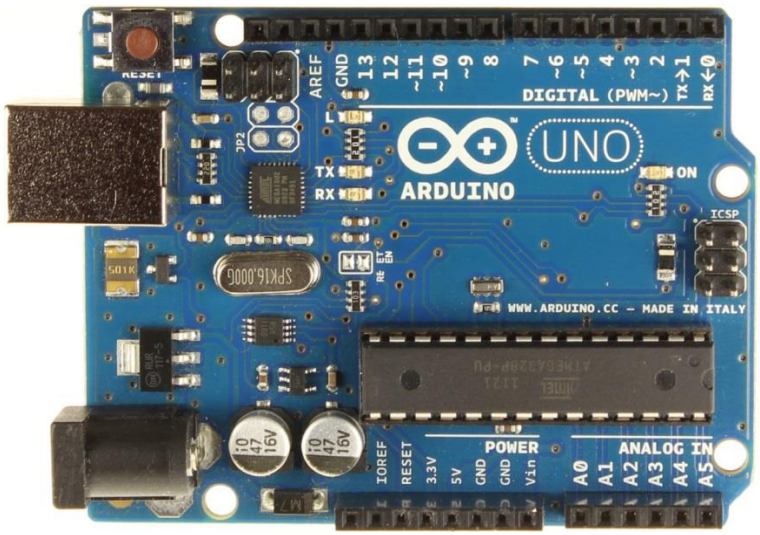

Figura 2. 1: Placa Arduino Uno.

En la tabla 2.1 se muestran las características básicas de las tarjetas controladoras Arduino Uno. En la figura 2.2 se muestran las especificaciones de Arduino Uno.

Tabla 2. 1: Características básicas de Arduino Uno.

Microcontrolador:	ATmega328
Voltaje de funcionamiento:	5V
Voltaje de entrada (recomendado):	7-12V
Voltaje de entrada (límites):	6-20V
Pines Digitales de I/O:	14 (de los cuales 6 proporcionan PWM)
Pines de entrada analógica:	6
Corriente DC por Pines I/O:	40 mA
Corriente DC para pines de 3.3V:	50 mA
Memoria Flash:	32 KB (ATmega328)
SRAM:	2 KB (ATmega328)
EEPROM:	1 KB (ATmega328)
Velocidad de reloj:	16 MHz

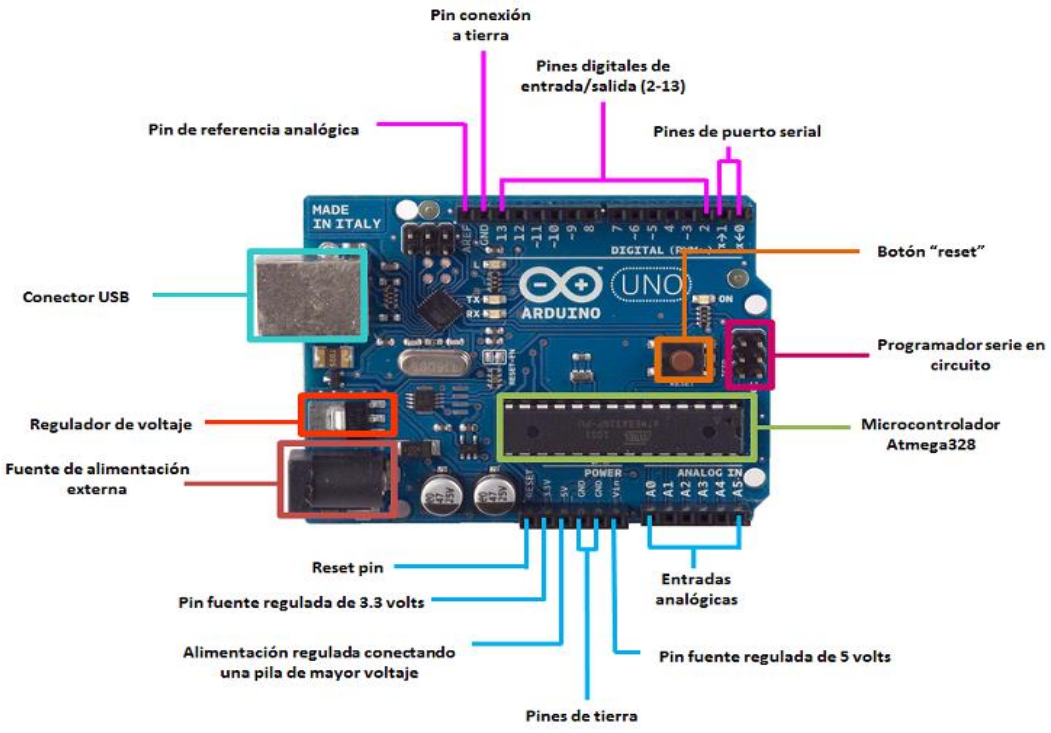

Figura 2. 2: Especificación de los elementos de la placa Arduino Uno

A continuación se describen cada una de las especificaciones mostradas en la figura 2.2 de la tarjeta Arduino Uno:

1. **Microcontrolador (uC) ATmega328**

   Este uC perteneciente a la compañía Atmel disponen de: una memoria flash de 32 kB, una memoria RAM de 2 kB y una memoria EEPROM de 1 kB. En la figura 2.3 se muestra el uC ATmega 328, el mismo que se utiliza para sustituir a los microcontroladores freeduino, arduino duemilanove (véase la figura 2.4) y diecimila; aunque se puede montar sobre un protoboard.

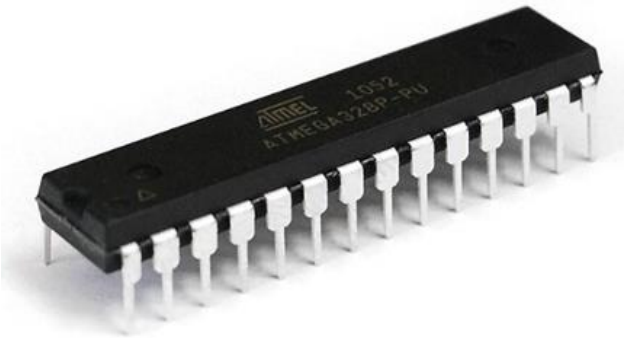

Figura 2. 3: Microcontrolador ATmega328 de Atmel.

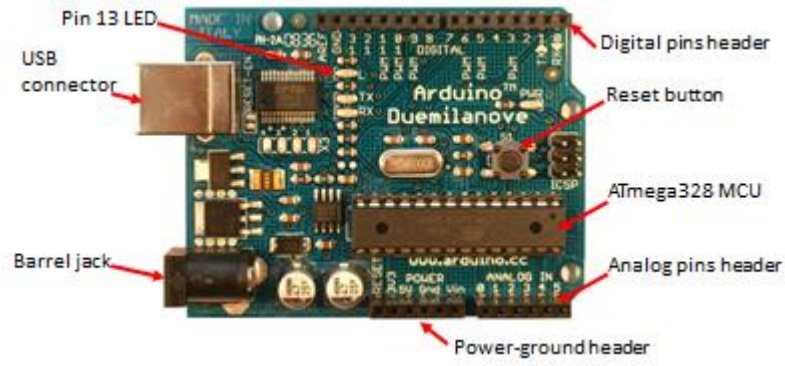

Figura 2. 4: Tarjeta Arduino Duemilanove con uC ATmega328

En la tabla 2.1 se muestran ciertas características importantes en los microcontroladores ATmega328.

Tabla 2. 2: Características del uC ATmega328.

Voltaje de funcionamiento:	5V
Memoria Flash:	32 KB de los cuales 512 bytes son utilizados por el gestor de arranque (bootloader)
SRAM:	2 KB
EEPROM:	1 KB
Velocidad de reloj:	16 MHz
Gestor de arranque:	Preinstalado

En la figura 2.5 se muestra la arquitectura interna de los microcontroladores ATmega328.

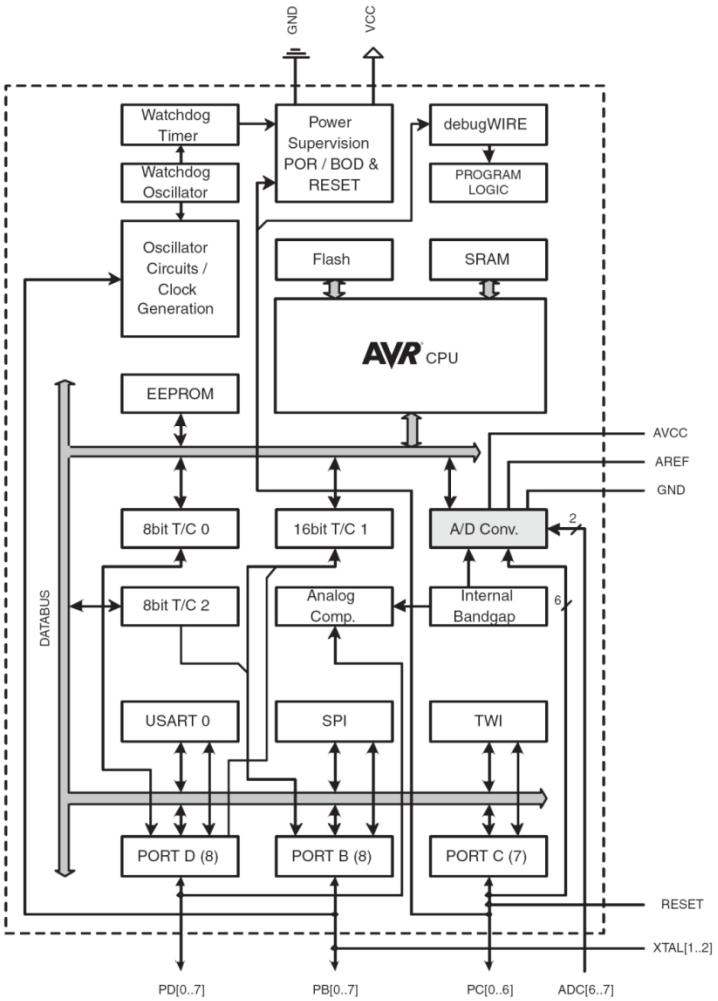

Figura 2. 5: Arquitectura interna del uC ATmega328 de Atmel.

2. Botón "Reset"

Este botón o pulsador de Reset permite reiniciar el uC ATmega328, cuyo valor es 0 V (LOW), generalmente utilizado en sistemas tipo escudo (shields) en la cual no se permite acceder a este pulsador en la placa.

3. Programación serial en circuitos.

Este terminal o punto de conexión permite programar el uC ATmega328, también conocido como programación ICSP (In-Circuit Serial Programming). En la figura 2.6 se muestra el diagrama esquemático de ICSP, que permite la comunicación entre el software y hardware (uC). También es muy utilizado en la mayoría de uC's PIC, en la cual son programados sin que se retire el integrado (chip) dentro del mismo circuito en el cual forma parte.

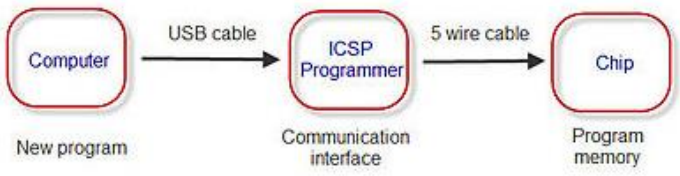

Figura 2. 6: Comunicación típica para programación.

4. Pines con modulación por ancho de pulso, PWM

En los pines digitales 3, 5, 6, 9, 10 y 11 se establece el PWM como salidas de 8 bits, a través de la función analogWrite ( ). La PWM es en sí una técnica que permite modificar los ciclos de las señales periódicas, también es muy utilizada para transferir datos a través de algún canal de comunicación o empleada para el control de energía enviada a una carga específica.

5. Pines de puerto serial.

Los pines de puerto serial, son utilizados para recibir (RX) y transmitir (TX) datos. Los pines RX (0) y TX (1) tienen conexión directa con pines correspondientes del uC ATmega328.

6. Interrupciones externas

Las interrupciones externas están ubicadas en los pines 2 y 3, éstos son utilizados para interrupciones mediante valores bajos, ya sea por flancos ascendentes o flancos descendentes, o por cambios de valores.

7. Bus de la Interfaz periférica serial, SPI

Estos SPI se encuentran en los pines 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) y es un protocolo de datos en serie síncrono utilizado por los microcontroladores para comunicarse con uno o más dispositivos periféricos rápidamente en distancias cortas. También utilizado para la comunicación entre dos microcontroladores.

8. Conexión a tierra (GND).

Los pines de tierra permiten el común del circuito, con un valor de 0 voltios.

9. Pines de tensión de referencia (AREF)

En las figuras 2.1 y 2.2 se muestra el pin AREF, se encarga de generar la tensión de referencia para entradas análogas. La instrucción a utilizar es analogReference ().

10. Conector USB

En la figura 2.2 se muestra el conector USB que tiene Arduino Uno, este estándar permite la comunicación o transmisión de datos a dispositivos electrónicos o para comunicarse a través de una computadora. Este estándar en sí, es un controlador de USB COM, es decir, que no requiere de controladores externos. Aunque para el sistema operativo Windows, se requiere de archivos .inf. Para saber si existe comunicación los LEDs de RX y TX de la tarjeta deben parpadear mientras se transmiten datos a través del USB.

11. Conector de alimentación externa.

En la figura 2.2 se muestra dicho conector tipo hembra (2.1mm) permite la alimentación de la tarjeta Arduino Uno.

12. Pines para alimentación regulada a 3.3 V.

Una fuente de voltaje a 3.3 voltios generada en el chip FTDI integrado en la placa. La corriente máxima soportada 50mA.

13. Alimentación DC de 5V

La fuente de voltaje estabilizado usado para alimentar el microcontrolador y otros componentes de la placa. Esta puede provenir de VIN a través de un regulador integrado en la placa, o proporcionada directamente por el USB o otra fuente estabilizada de 5V.

14. Alimentación DC de entrada, V_{IN}.

También la tarjeta de entrenamiento Arduino Uno se puede alimentar de una fuente de voltaje externa (a parte de la alimentación por USB) a través del pin V_{IN}.

15. Entrada analógica (Analog In).

La tarjeta de entrenamiento Arduino Uno dispone de 6 pines de entrada analógicas, donde cada pin llega a suministrar 10 bits (2^{10}=1024 valores).

16. Oscilador de cristal.

Es utilizado para como la señal de reloj necesaria para el funcionamiento del microcontrolador ATmega 328.

2.3 Entorno de Programación.

Para las tarjetas de entrenamiento Arduino Uno son utilizadas las plataformas de programación (software) versión 0021 y posteriores. En la misma página web de arduino se puede descargar e instalar dicha plataforma. Asimismo, el compilador que emplea Arduino Uno es sencillo y a la vez muy intuitivo de usar. Posteriormente, lo trascendental es escoger correctamente la tarjeta de entrenamiento y los puertos de conexión.

En la figura 2.7 se ilustra la interface básica del compilador utilizado en las tarjetas de entrenamiento Arduino Uno. El mismo se encarga de la edición de códigos fuentes, manejo de consolas de texto, y mediante pulsadores podemos seleccionar

ciertas funciones comunes y también se puede acceder al menu principal. Finalmente este programa se conecta a la tarjeta de entrenamiento Arduino Uno con la finalidad de cargar los códigos fuentes.

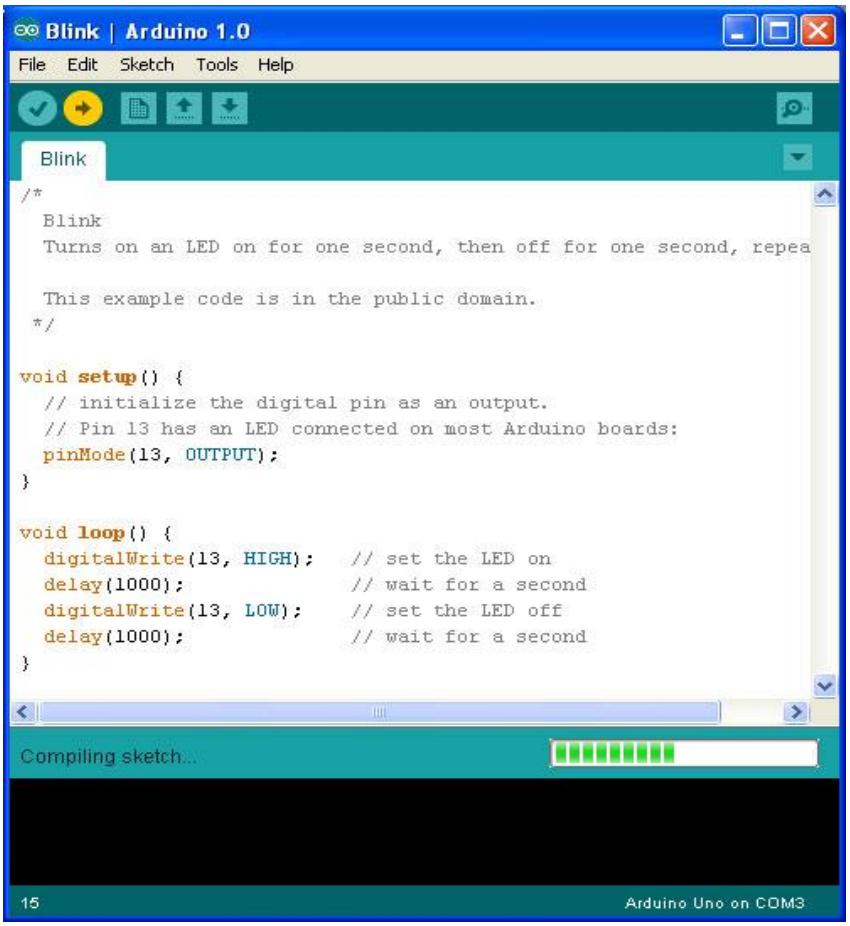

Figura 2. 7: Interface del compilador Arduino Uno.

Esta plataforma de programación es conocida como Blink, desarrollada por Arduino. Los códigos fuentes o también sketch (nominado por Arduino) son compilados directamente desde el editor de texto. Como en todo software, Blink permite copiar, cortar, pegar, buscar y sustituir texto. También se puede visualizar los mensajes tanto de la compilación y carga de cada sketch, así mismo se muestran los errores (véase la figura 2.7). Mientras que con la barra de herramientas se puede comprobar cada uno de los procesos explicados tal como se describe en la tabla 2.3.

Tabla 2. 3: Funciones de la barra de herramientas de Blink de Arduino.

 Verificar compilación: Chequea el código en busca de errores.
 Detener: Finaliza la monitorización serie y oculta otros botones
 Nuevo: Crea un nuevo sketch.
 Abrir: Presenta un menú de todos los programas sketch de su "sketchbook", (librería de sketch). Un click sobre uno de ellos lo abrirá en la ventana actual.
 Grabar: Salva el programa sketch.
 Cargar a tarjeta I/O: Compila el código y lo vuelca en la placa E/S de Arduino. Ver volcado más detalles abajo.
 Monitoreo: Inicia la monitorización serie.

Adicionalmente existen otras funciones o comandos en la barra de herramientas principal, tales como File (Archivo), Edit (Edición), Sketch (código fuente), Tools (Herramientas) y Help (ayuda). Dichas funciones son descritas en las tablas 2.4, 2.5, 2.6 y 2.7.

Tabla 2. 4: Descripción de la función File.

Add File:	Permite añadir ficheros de código fuente (se incluirá desde su ubicación actual). El fichero aparece en una nueva pestaña en la ventana del programa. Los ficheros pueden ser quitados del sketch utilizando el menú "tab".

Tabla 2. 5: Descripción de la función Edit.

Copy for Discourse:	Copia el código sketch en el portapapeles con el formato adecuado para publicarlo en un foro, incluyendo la sintaxis coloreada.
Copy as HTML:	Copia el código de un programa (sketch) al portapapeles en formato HTML, adecuándolo para incrustarlo en una página web.

Tabla 2. 6: Descripción de la función Sketch.

Verify/Compile:	Verifica los errores de su programa (sketch).
Import Library:	Añade una librería a su programa (sketch) insertando la sentencia include en el código.
Show Sketch Folder:	Abre la carpeta de programas (sketch) en el escritorio.

Tabla 2. 7: Descripción de la función Tools.

Auto Format:	Proporciona la estética del formato de código fuente, por ejemplo, realiza tabulaciones entre la apertura y cierre de llaves, y las sentencias que tengan que ser tabuladas lo estarán.
Board	Selecciona el tipo de tarjeta de Arduino que se está utilizando.
Serial Port	Este menú contiene todos los dispositivos de serie (reales o virtuales) de su equipo. Se refrescará automáticamente cada vez que se abra tools.
Burn Bootloader	Permite grabar un gestor de arranque (bootloader) dentro del micro-controlador de Arduino. Aunque no es un requisito para el funcionamiento de la placa, le será útil si compra un nuevo ATmega (el cual viene normalmente sin gestor de arranque). Asegúrese que ha seleccionado la placa correcta en el menú Boards antes de grabar el bootloader. Cuando use AVR ISP, tendrá que seleccionar en el menú Serial Port el puerto correspondiente.

2.4 Programación y funciones específicas.

Si comparamos Arduino con otras tarjetas de entrenamiento como los microcontroladores PIC de Microchip, se puede verificar que la única desventaja es la programación que se desarrolla bajo el lenguaje de alto nivel C++. Por este motivo tiene a ser más complicado, aunque es más versátil.

La ventaja que tiene Arduino, es que mediante C++ se pueden desarrollar diferentes aplicaciones que dependerán mucho del usuario. Por ejemplo, se puede diseñar la linealidad de un sensor sin requerir de circuitos externos, que permita la adquisición de datos de los sensores y poder guardarlos en la memoria, esto siempre que se le asigne un valor específico en un rango definido a través de convertidores A/D que viene incorporado en el microcontrolador ATmega 328.

Asimismo, la tarjeta de entrenamiento Arduino Uno, permite el uso de la mayoría de librerías que dispone comúnmente C++ y que incluyen las estructuras básicas. Mientras que la tarjeta controladora nos facilita ciertas funciones que benefician mucho a los programadores. A continuación se muestra la tabla 2.8 en la cual se describen las estructuras y funciones principales de la programación.

Tabla 2. 8: Descripción de las estructuras y funciones de Blink de Arduino.

setup () (Configuración):	Setup () se inicia al arrancar el programa. Inicializa las variables, modos de contactos, uso de bibliotecas, etc. La función de configuración sólo se ejecutará una vez, después del encendido o reinicio de la placa Arduino.
loop () (Bucle):	Después de la función setup (), se inicializa y establecen valores iniciales. Se utiliza para controlar activamente la placa Arduino.
pinMode():	Configura pines E/S, para utilizarla, se pasa el número del pin que vas a configurar y las constantes INPUT u OUTPUT.
digitalWrite():	Escribe o envía un valor HIGH o LOW hacia un pin digital. Por ejemplo, la línea: digitalWrite (ledPin, HIGH);
digitalRead ():	Lee el valor de un pin digital especificado, HIGH o LOW. Sintaxis-> digitalRead (pin) Parámetros->pin: el número de pin digital que quieres leer (int) Devuelve HIGH o LOW.

attachInterrupt (interrupción, función, modo):	Especifica la función a la que invocar cuando se produce una interrupción externa. Reemplaza cualquier función previa que estuviera enlazada a la interrupción. La mayoría de las placas Arduino tienen dos interrupciones externas: Las número 0 (pin digital 2) y 1 (pin digital 3). Parámetros: • interrupción: el número de la interrupción (int) • función: la función a la que invocar cuando la interrupción tiene lugar; esta función no debe tener parámetros ni devolver nada. Esta función es a veces referenciada como rutina de interrupción de servicio • modo define cuando la interrupción debe ser disparada. Hay cuatro constantes predefinidas como valores válidos:
detachInterrupt (interrupt)	Apaga la interrupción dada. Interrupt: el número de interrupción a invalidar (0 o 1).
noInterrupts ()	Desactiva las interrupciones (pueden reactivarse usando interrupts()). Las interrupciones permiten que las operaciones importantes se realicen de forma transparente y están activadas por defecto. Algunas funciones no funcionarán y los datos que se reciban serán ignorados mientras que las interrupciones estén desactivadas. Las interrupciones pueden perturbar ligeramente el tiempo de temporizado, sin embargo puede que sea necesario desactivarlas para alguna parte crítica del código.
Interrupts ()	Activa las interrupciones (después de haberlas desactivado con noInterrupts (). Las interrupciones permiten que se ejecuten ciertas tareas en segundo plano que están activadas por defecto. Algunas funciones no funcionarán correctamente mientras las interrupciones estén desactivadas y la comunicación entrante puede ser ignorada. Las interrupciones pueden perturbar ligeramente la temporización en el código y deben ser desactivadas sólo para partes particularmente críticas del código.

2.5 Características técnicas de los pines de E/S.

En los microcontroladores ATmega de Arduino la mayoría de los pines son de entrada, para lo cual debe configurarse de manera explícita a través de pinMode ().

En otras palabras los pines de entrada son establecidos como de alta impedancia, con esto se lograría que cada uno de los pines de entrada circular una cantidad pequeña de corriente (en microamperios, uA). Esta impedancia equivale a una resistencia conectada en serie frente al pin y cuyo valor puede aproximarse a $100\,M\Omega$. La pequeña corriente pasa en cada pin de entrada de un estado a otro, y permite desarrollar aplicaciones que utilizan dispositivos tales como sensores de tacto, lectura a través de fotodiodos, etc.

Esto también significa sin embargo, que los terminales de entrada sin conectar nada a ellos, o con los cables conectados a ellos sin estar conectados a otros circuitos, reflejarán cambios aparentemente aleatorios en el estado de pin, recogiendo el ruido eléctrico del entorno, o el acoplamiento capacitivo del estado de un pin próximo.

- **Resistencias Pull-up y Pull-down:** por lo general se recomienda conocer el estado de los pines de entrada. Por ejemplo, para las resistencias Pull-up y Pull-down de entrada con conectadas a +5 V y tierra (GND) respectivamente. Existen otros valores en las resistencias pull-up de aproximadamente $20\,k\Omega$ que viene incorporado en los microcontroladores ATmega, las mismas son accesibles a través de Blink. La manera de acceder a las resistencias pull-up es como sigue:

```
pinMode (pin, INPUT);        // pone el pin como entrada
digitalWrite (pin, HIGH);    // activa la resistencia pull-up
```

Debemos considerar que las resistencias pull-up suministran la corriente necesaria para encender tenuemente LEDs que son conectados en los pines

que han sido designados como entradas. Tal vez puede suceder que los LEDs tengan un funcionamiento muy tenuemente, lo cual significa que el programador (usuario) no considero la instrucción pinMode () que permite configurar los pines como salidas.

Es de suma importancia recordar que las resistencias pull-up son registradas en las posiciones de memoria (registros) interna de ATmega, las mismas son configuradas como HIGH (si pin está en alto, 5 V o 1 lógico) o LOW (si pin están en bajo, GND o 0 lógico). En otras palabras, los pines pueden configurarse para activar resistencias pull-up como entrada (HIGH) y como salida (OUTPUT) con pinMode (). Puede ocurrir lo contrario si tenemos pines de salida, las resistencias pull-up quedarán en estado alto activa y cambia la entrada (INPUT) con pinMode ().

Ahora describimos cuando los pines sean configurados como salidas (OUTPUT) mediante la instrucción pinMode (), lo que indica un estado de baja impedancia, permitiendo el suministro de corriente a otros elementos del circuito. Para esta configuración, los pines abastecen corriente tanto positiva (+) como negativa (-) cuyo valor máximo es 40 mA.

El suministro de corriente hace que los LEDs emitan una luz brillante y también conectar a la mayoría de sensores (movimiento, tacto, etc.), pero no muy robusta para elementos tales como relés, motores y solenoides. Hay que tener mucho cuidado con

algún cortocircuito que se generen en los pines de Arduino, de llegar a ocurrir esto, podría dañar tanto transistores (pines de salida) como el microcontrolador ATmega.

De acuerdo a lo explicado, es importante que se conecten pines de salida con ciertos dispositivos a través de resistencias tanto de $470\ \Omega$ como de $1\ k\Omega$, lo que permite definir una máxima corriente a los pines requeridos para una aplicación particular.

2.6 Versiones de las tarjetas Arduino.

Hay diversos tipos de versiones de las tarjetas de entrenamiento Arduino. La que se utiliza en el presente trabajo de titulación es la Arduino Uno, y las demás tarjetas Arduino son: Duemilanove (incorpora el uC Atmega328), Diecimila, Mega (incorpora el uC Atmega1280). A continuación se describen brevemente las diversas tarjetas de entrenamiento de Arduino y se muestran sus características.

2.6.1. Arduino Duemilanove

La tarjeta Arduino Duemilanove (véase la figura 2.8) tiene incorporado al microcontrolador Atmega168 o Atmega328, dispone de 14 pines digitales para ser configurados como E/S, 6 entradas analógicas, un oscilador de cristal con una frecuencia de 16Mhz, conexión USB, entrada de alimentación, una cabecera ISCP, y un pulsador para reinicio (Reset). Contiene todo lo necesario para utilizar el microcontrolador; simplemente conectándolo a un ordenador mediante un cable USB o alimentándolo a través de un batería (cargador) para su funcionamiento. En la tabla 2.9 se muestran en resumen las características de las tarjetas duemilanove.

Tabla 2. 9: Características principales de tarjeta Arduino Duemilanove.

Microcontrolador	ATmega368 (ATmega168 en versiones anteriores)
Voltaje de funcionamiento	5 V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	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	16 KB (ATmega168) o 32 KB (ATmega328) de las cuales 2 KB las usa el gestor de arranque(bootloader)
SRAM	1 KB (ATmega 168) o 2 KB (ATmega 328)
EEPROM	512 bytes (ATmega 168) o 1 KB (ATmega 328)
Velocidad de reloj	16 MHz

Figura 2. 8: Tarjeta Arduino Duemilanove.

2.6.2. Arduino Diecimila

La tarjeta Arduino Diecimila (véase la figura 2.9) tiene incorporado al microcontrolador ATmega168, dispone de 14 pines digitales de E/S, 6 entradas analógicas, un oscilador de cristal con una frecuencia de 16MHz, conexión USB y un

pulsador para reinicio (Reset). Su funcionamiento es similar a la de la tarjeta duemilanove. En la tabla 2.10 se muestran las características de Arduino Diecimila.

Tabla 2. 10: Características principales de tarjeta Arduino Diecimila.

Microcontrolador	ATmega168
Voltaje de funcionamiento	5 V
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	16 KB (2 KB reservados para gestores de arranque)
SRAM	1 KB
EEPROM	512 bytes
Velocidad de reloj	16 MHz

Figura 2. 9: Tarjeta Arduino Diecimila.

2.6.3. Arduino Nano

El Arduino Nano es una pequeña y completa placa basada en el ATmega328 (Arduino Nano 3.0) o ATmega168 (Arduino Nano 2.x) que se usa conectándola a una protoboard. Tiene más o menos la misma funcionalidad que el Arduino Duemilanove, pero con una presentación diferente. No posee conector para alimentación externa, y funciona con un cable USB Mini-B en vez del cable estándar. El nano fue diseñado y está siendo producido por Gravitech.

Tabla 2. 11: Características principales de tarjeta Arduino Nano.

Microcontrolador	Atmel – Atmega168 o Atmega328
Voltaje de funcionamiento	5 V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	6-20V
Pines E/S digitales	14 (6 proporcionan salida PWM)
Pines de entrada analógica	8
Corriente máxima por cada pin de E/S	40 mA
Memoria Flash	16 KB (ATmega168) o 32 KB (ATmega328) de los cuales 2KB son usados por el bootloader
SRAM	1 KB (Atmega168) o 2 KB (ATmega328)
EEPROM	512 bytes (Atmega168) o 1 KB (Atmega328)
Velocidad de reloj	16 MHz
Dimensiones	18,5mm x 43.2mm

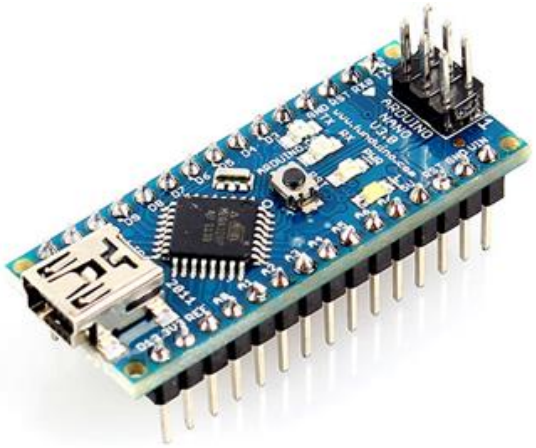

Figura 2. 10: Tarjeta Arduino Nano.

2.6.4. Arduino Mega

La tarjeta Arduino Mega (véase la figura 2.11) incorpora el microcontrolador Atmega1280, dispone de 54 pines digitales de E/S, 16 entradas digitales, 4 UARTS (puertos serie por hardware), un oscilador de cristal con frecuencia de 16MHz, conexión USB, entrada de corriente, conector ICSP y un pulsador para reinicio (Reset). Su funcionamiento es similar a las otras versiones de Arduino ya descritas con anterioridad. Arduino Mega es compatible con las tarjetas Arduino Duemilanove o Diecimila. En la tabla 2.12 se muestran las características de la tarjeta Arduino Mega.

Figura 2. 11: Tarjeta Arduino Mega.

Tabla 2. 12: Características principales de tarjeta Arduino Mega.

Microcontrolador	Atmega1280
Voltaje de funcionamiento	5 V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines E/S digitales	54 (14 proporcionan salida PWM)
Pines de entrada analógica	16
Corriente máxima por cada pin de E/S	40 mA
Intensidad en pin 3.3V	50 mA
Memoria Flash	128 KB de las cuales 4 KB las usa el gestor de arranque (bootloader)
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz

2.6.5. Arduino BT

La tarjeta Arduino BT (véase la figura 2.12) dispone internamente de un módulo de comunicación inalámbrico Bluetooth. Este módulo utilizado es del tipo Bluegiga WT11 cuya versión es iWrap. El módulo es configurado mediante él envió de comandos por el puerto serie del microcontrolador Atmega168. Para poder acceder al dispositivo o emparejar se debe visualizar ARDUINOBT y el código de emparejamiento es 12345. En la tabla 2.12 se muestran las características de la tarjeta Arduino BT.

Tabla 2. 13: Características principales de tarjeta Arduino BT.

Microcontrolador	Atmega3280
Voltaje de funcionamiento	5 V
Voltaje de entrada	2.5-12V
Pines E/S digitales	14 (6 proporcionan salida PWM)
Pines de entrada analógica	6
Intensidad por cada pin de E/S	40 mA
Intensidad en pines para 3.3V	500 mA
Intensidad en pines para 5V	1000 mA
Memoria Flash	32 KB de las cuales 2 KB las usa el gestor de arranque (bootloader)
SRAM	2 KB
EEPROM	2 KB
Velocidad de reloj	16 MHz
Módulo BT	2.1 WT11i-A-AI4

Figura 2. 12: Tarjeta Arduino Bluetooth.

2.6.6. Arduino LilyPad.

La tarjeta Arduino LilyPad (véase la figura 2.13) ha sido diseñado para llevar en prendas de vestir. Arduino LilyPad tiene incorporado el microcontrolador ARmega168V o ATmega328V. Puede utilizar con complementos similares como fuentes de alimentación, sensores actuadores unidos por hilo conductor. En la tabla 2.14 se muestran las características de la tarjeta Arduino LilyPad.

Tabla 2. 14: Características principales de tarjeta Arduino LilyPad.

Microcontrolador	Atmega168V o Atmega328V
Voltaje de funcionamiento	2.7-5.5 V
Voltaje de entrada	2.7-5.5 V
Pines digitales de E/S	14 (6 proporcionan salida PWM)
Pines de entrada analógica	6
Intensidad por cada pin de E/S	40 mA
Memoria Flash	16 KB de las cuales 2 KB las usa el gestor de arranque (bootloader)
SRAM	1 KB
EEPROM	512 Bytes
Velocidad de reloj	8 MHz
Módulo BT	2.1 WT11i-A-AI4

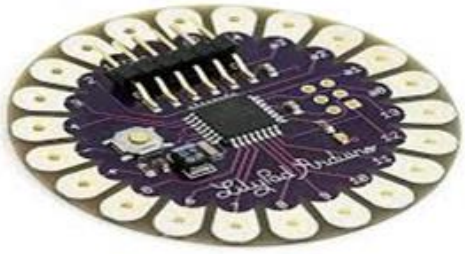

Figura 2. 13: Tarjeta Arduino LilyPad.

2.6.7. Arduino Fio

La tarjeta Arduino Fio (véase la figura 2.14) tiene incorporado el microcontrolador ATmega328P, que funciona con una alimentación de 3.3V y opera a una frecuencia de oscilación de 8MHz. La Arduino Fio dispone de 14 pines digitales de E/S, 8 entradas analógicas, un oscilador resonante, pulsador para reinicio (Reset), y agujeros para montar conectores de pines. Tiene conexiones para una batería de polímero de Litio e incluye un circuito de carga a través de USB. En el reverso de la placa tiene disponible un zócalo para módulos XBee.

Figura 2. 14: Tarjeta Arduino Fio.

La Arduino FIO ha sido diseñada para desarrollar aplicaciones mediante comunicación inalámbrica. Además, si utiliza un adaptador de USB a XBee modificado, como el USB Explorador de XBee, el usuario puede subir sketches de forma inalámbrica. En la tabla 2.15 se muestra las características de las tarjetas de entrenamiento Arduino Fio.

Tabla 2. 15: Características principales de tarjeta Arduino Fio.

Microcontrolador	Atmega328P
Voltaje de funcionamiento	3.3 V
Voltaje de entrada	3.3-12 V
Voltaje de Entrada en Carga	3.7 - 7 V
Pines digitales de E/S	14 (6 proporcionan salida PWM)
Pines de entrada analógica	8
Intensidad por cada pin de E/S	40 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	8 MHz

2.6.8. Arduino Mini

La tarjeta de entrenamiento Arduino Mini (véase la figura 2.15) es una placa con un pequeño microcontrolador basada en el ATmega168, pensada para ser usada en placas de prototipado y donde el espacio es reducido. Se puede programar con Mini USB u otros adaptadores USB o RS232 a TTL serial.

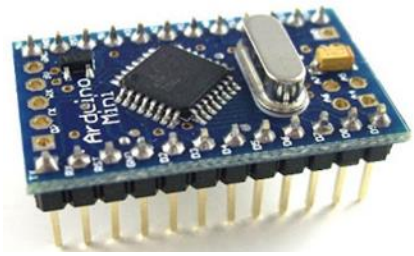

Figura 2. 15: Tarjeta Arduino Mini.

Tabla 2. 16: Características principales de tarjeta Arduino Mini.

Microcontrolador	Atmega168
Voltaje de funcionamiento	5 V
Voltaje de entrada	7 - 9 V
Pines digitales de E/S	14 (6 proporcionan salida PWM)
Pines de entrada analógica	8 (de las cuales 4 se extienden en pines)
Intensidad por cada pin de E/S	40 mA
Memoria Flash	16 KB de las cuales 2 KB las usa el gestor de arranque (bootloader)
SRAM	1 KB
EEPROM	512 Bytes
Velocidad de reloj	16 MHz

2.6.9. Arduino Pro

La tarjeta Arduino Pro (véase la figura 2.16) viene incorporado ya sea el microcontrolador ATmega168 o el ATmega328. Arduino Pro dispone de dos versiones, la primera con 3.3 V/8 MHz y la segunda con 5 V/16 MHz. Tiene 14 pines digitales de E/S, 6 entradas analógicas, un oscilador resonante interno, pulsador para reinicio (Reset) y agujeros para el montaje de tiras de pines. Tiene 6 pines para la conexión a un cable FTDI ya sea para ser usada como puerto USB o de alimentación.

La placa viene sin conectores montados, permitiendo el uso de varios tipos de conectores o soldado directo de cables según las necesidades de cada proyecto en particular. En la tabla 2.17 se muestran las características de la tarjeta Arduino Pro.

Tabla 2. 17: Características principales de tarjeta Arduino Mini.

Microcontrolador	ATmega168 o ATmega328
Voltaje de entrada	3.35 -12v (en el modelo de 3.3v) o 5 - 12v (en el modelo de 5v)
Pines digitales de E/S	14 (6 proporcionan salida PWM)
Intensidad por cada pin de E/S	40 mA
Memoria Flash	16KB en el ATmega168 y 32KB con el ATmega328 de las cuales 2 KB las usa el gestor de arranque (bootloader)
SRAM	1 KB
EEPROM	512 Bytes
Velocidad de reloj	8 MHz (modelo de 3.3V) o 16 MHz (modelo de 5V)

Figura 2. 16: Tarjeta Arduino Pro.

2.6.10. Arduino Pro Mini

La tarjeta de entrenamiento Arduino Pro Mini (véase la figura 2.17) tiene incorporado el microcontrolador ATmega168, dispone de 14 pines digitales de E/S, 8 entradas analógicas, un circuito resonador, un pulsador de reinicio (Reset), y agujeros para el montaje de cabezales de pin. Un cabezal de 6 pines se conecta a un

cable FTDI o tablero del desbloqueo Sparkfun para proporcionar alimentación USB y la comunicación a la junta.

Tabla 2. 18: Características principales de tarjeta Arduino Pro Mini.

Microcontroladores	ATmega168
Tensión de funcionamiento	3.3V o 5V (según el modelo)
Voltaje de entrada	3,35 -12 V (modelo de 3,3 V) o 5 - 12 V (modelo 5V)
Digital pines I / O	14 (de los cuales 6 proporcionan PWM)
Pines de entrada analógica	8
Corriente DC por Pin I / O	40 mA
Memoria Flash	16 KB (de los cuales 2 KB utilizado por gestor de arranque)
SRAM	1 KB
EEPROM	512 bytes
Velocidad de reloj	8 MHz (modelo de 3,3 V) o 16 MHz (5V modelo)

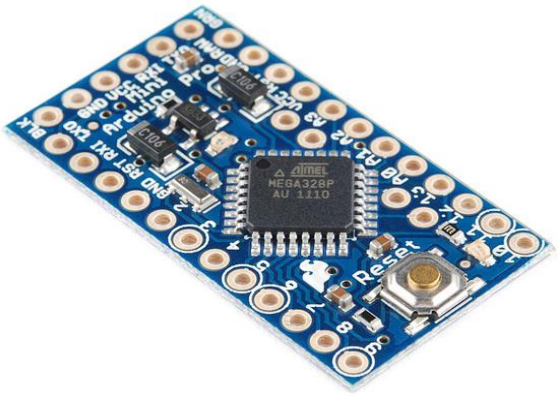

Figura 2. 17: Tarjeta Arduino Pro mini.

2.6.11. Arduino Serial

Es una placa básica que utiliza una interfaz RS232 para comunicarse con el ordenador o para la carga de sketches. Esta placa es fácil de montar, incluso como

ejercicio de aprendizaje. Se ha diseñado para utilizar los componentes más simples posibles, de manera que sea fácil de construir, incluso si buscas las componentes en la tienda de la esquina.

Tabla 2. 19: Características principales de tarjeta Arduino Serial.

Microcontrolador	ATmega8
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12 V
Voltaje de entrada (limites)	6-20 V
Pines E/S Digitales	14 (de ellos 6 son salidas PWM)
Pines de entrada Analógica	6
Intensidad por pin de E/S	40 mA
Intensidad por pin de 3.3V	50 mA
Velocidad del reloj	16 MHz

Figura 2. 18: Tarjeta Arduino conexión serie.

2.7 Los Shields de Arduino

Los Shields son placas que se colocan encima de la placa Arduino y que amplían una nueva función para que sea controlada desde Arduino, para controlar diferentes aparatos, adquirir datos, etc.

2.7.1. Módulo de comunicación Xbee.

La Xbee shield permite a una placa Arduino comunicarse de forma inalámbrica usando Zigbee. Está basada en el módulo Xbee de MaxStream. El módulo puede comunicarse hasta 100ft (30 metros) en interior o 300ft (90 metros) al aire libre (en visión directa). Puede ser usado como reemplazo del puerto serie/usb o puedes ponerlo en modo de comandos y configurarlo para una variedad de opciones de redes broadcast o malladas. La shield tiene pistas desde cada pin del Xbee hasta un orificio de soldar. También provee conectores hembra para usar los pines digitales desde 2 hasta 7 y las entradas analógicas, las cuales están cubiertas por la shield (los pines digitales de 8 a 13 no están cubiertos por la placa, así que puedes usar los conectores de la placa directamente). Podemos ver su aspecto en la figura 2.7.1.

Figura 2. 19: Módulo de comunicación Xbee de Arduino.

2.7.2. Módulo de comunicación Ethernet

La Arduino Ethernet Shield permite a una placa Arduino conectarse a internet. 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 a internet 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.

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
- FULLD: indica que la conexión de red es full duplex
- 100M: indica la presencia de una conexión de red de 100 Mb/s (de forma opuesta a una de 10Mb/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. En la figura 2.7.3 podemos ver el aspecto de esta placa.

Figura 2. 20: Módulo de comunicación Ethernet de Arduino.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DEL PROYECTO

3.1 Generalidades de diseño e implementación.

En el presente capitulo tendremos una visión global de nuestro proyecto el cual se trata de una práctica plataforma interactiva en la cual demostraremos las instrucciones del lenguaje ensamblador para los microcontroladores de la familia ATMEL y lo haremos por medio de la plataforma de entrenamiento Arduino, el cual contiene al microcontrolador ATmega328 y una serie de elementos (resistencias, LEDs, displays, etc.).

La demostración de las instrucciones se la hará por medio de sencillos pero prácticos ejercicios en los cuales hemos hecho lo posible para utilizar los diferentes tipos de instrucciones de dicho microcontrolador ya sean este tipo de instrucciones de tipo aritméticas, lógicas, transferencia de datos, control del programa y bits.

3.2 Implementación.

Uno de los principales problemas en la realización de este proyecto fue ¿Cómo Desarrollar una plataforma interactiva para demostrar las instrucciones en Lenguaje Ensamblador utilizando microcontroladores ATMEL? Nosotros pensamos que la mejor manera para la demostración de las instrucciones de los microcontroladores ATMEL es desarrollando un conjunto de ejercicios. Donde cada uno estos ejercicios

está destinado a resolver un problema específico, los cuales están integrados en una plataforma compacta, plataforma que probablemente para los expertos en este tipo de Microcontroladores podría ser cuestionado como una plataforma demasiado sencilla o básica. Pero es de gran utilidad para las personas que estén dando sus primeros pasos con este tipo de microcontroladores ya que por medio de este conjunto de ejercicios podrán observar y comprobar a través de software (AVR Studio 4, Proteus) y hardware (AVR Butterfly, Leds, pulsadores, displays, etc.) las diversas aplicaciones de las instrucciones en lenguaje Ensamblador de los microcontroladores de la familia ATMEL.

3.3 Plataforma Interactiva

En la presente plataforma la cual se la puede apreciar en la figura 3.1, como podemos observar que consta de un teclado matricial sencillo y sensores infrarrojos, sensor de temperatura, sensor de luz, juego de leds y un buzzer que hacen las veces de indicadores o alarma, y un display en el cual se muestra valores. En la plataforma interactiva se desarrollaran problemas como:

a. Detector de intensidad de la luz
b. Detector de temperatura
c. Generador de tonos
d. Detección de colores blanco y negro
e. Visualización de datos en display
f. Ingreso de datos en el teclado matricial

Las aplicaciones que se los han desarrollado de la forma más didáctica posible para ver de una manera clara el comportamiento del lenguaje Ensamblador. A continuación en la siguiente figura 3.1 se muestra la plataforma interactiva.

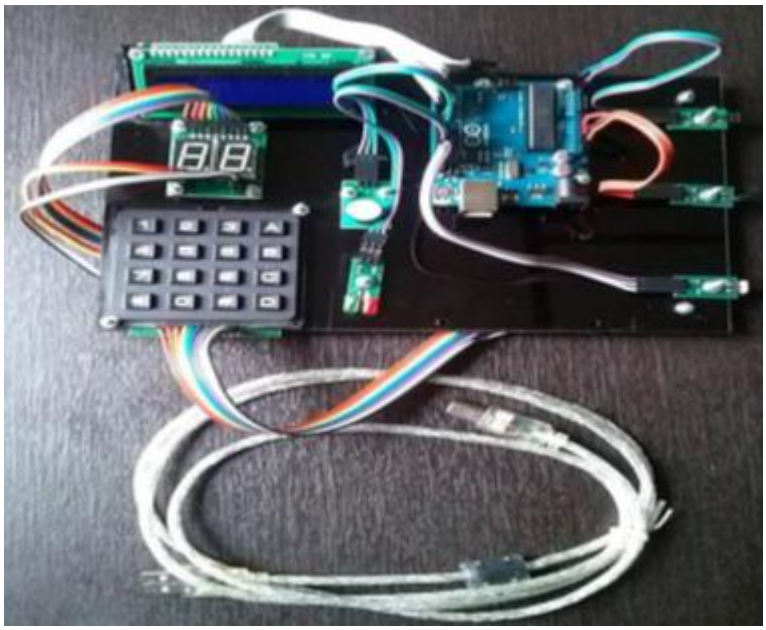

Figura 3. 1: Prototipo de plataforma interactiva.

3.4 Materiales Utilizados

Los materiales utilizados para montar esta plataforma interactiva son los siguientes:

a. Módulo Arduino Uno
b. Modulo infrarrojo QRD1114
c. Sensor de temperatura DS18B20
d. Sensor de luz I&T LDR
e. Modulo Buzzer I&T
f. Modulo de leds I&T
g. Display Cátodo-Común I&T
h. LCD 2X16 con Base

i. Keypad 4x4 NE

j. Chasis

k. Cables UTP

3.5 Breve bosquejo de los ejercicios prácticos.

Como ya lo sabíamos esta plataforma consta de cinco ejercicios orientados a resolver un problema específico los cuales serán descritos a continuación:

Ejercicio 1: Simulación de un sistema de iluminación

En este ejercicio de Sistema de iluminación en función de la luz recibida sobre el sensor LDR. El programa aumenta o disminuye el brillo de los leds y el sonido de un buzzer dependiendo de los valores sensados del LDR, en esta práctica se hace el uso de las salidas analógicas de los pines del arduino. Cuando se oscurece la LDR el sistema responde encendiendo con mayor intensidad los leds y agudiza el sonido del led. Los materiales utilizados en esta práctica son:

- Sensor LDR
- Buzzer
- Leds
- Placa Arduino

Ejercicio 2: Sistema de alerta por temperatura

En este ejercicio se va a desarrollar un Sistema de alerta por temperatura, el programa monitorea con un sensor de temperatura (DS18B20) el ambiente y si éste detecta una temperatura mayor a un umbral (30°c) emite un sonido de buzzer y enciende un led rojo, caso contrario se mantiene un led verde encendido indicando

que la temperatura sensada es la adecuada. Los materiales utilizados en esta práctica son:

- LCD
- Sensor de temperatura ds18b20
- Leds
- Buzzer

Ejercicio 3: Simulación de un sistema de acceso por clave

El programa recibe una clave la cual es oculta por asteriscos para su posterior verificación dentro del sketch, en caso de ser correcta se muestra un mensaje indicando “Bienvenido” y encendiendo un led Verde, si es incorrecta se mostrará un mensaje indicando “Incorrecto” y encendiendo un led rojo al presionar cada tecla debe sonar el buzzer. Para enviar la clave se presiona la tecla “A” y para borrar hacia atrás se presiona “C”.

Materiales utilizados

- Buzzer
- Teclado 4x4
- Pantalla lcd
- Leds

3.6 Desarrollo de los ejercicios prácticos.

En esta sección se describen los modos de operación de los elementos que conforman los ejercicios y su funcionamiento en conjunto para la aplicación implementada usando la plataforma de entrenamiento Arduino

3.6.1. Sistema de Iluminación.

En este ejercicio forjaremos iluminación en función de la luz recibida sobre el sensor LDR. La programación de este ejercicio está diseñada para aumentar o disminuye el brillo de los leds y el sonido de un buzzer dependiendo de los valores sensados del LDR, en esta práctica se hace el uso de las salidas analógicas de los pines del arduino. Cuando se oscurece el sensor LDR el sistema responde encendiendo con mayor intensidad los leds y agudiza el sonido del led

a. Diagrama de conexiones.

Para comprender esta aplicación se realizó el diagrama de conexiones para la transferencia de las señales a los diferentes dispositivos periféricos como se indica en la figura 3.2, donde vemos que se realiza una comunicación con la placa de entrenamiento Arduino la cual simula un sistema de iluminación conectadas a los LEDs, sensor LDR y buzzer, a fin de analizar el uso de las salidas analógicas.

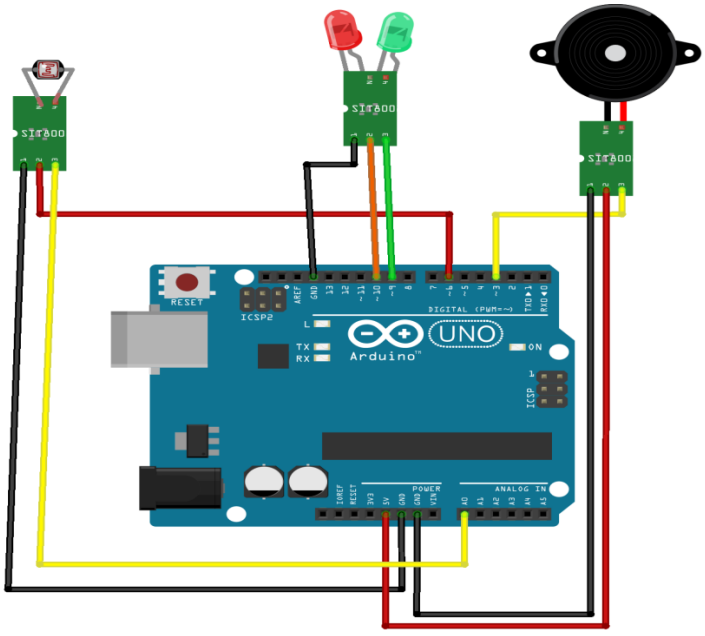

Figura 3. 2: Diagrama de conexiones del sistema de iluminación.

b. Conexiones de los dispositivos periféricos.

Conexión del buzzer Idetec

Buzzer a Arduino

S --> 43

VCC (+) --> 5V

GND (-) --> GND

Conexión de los leds (Rojo Verde) Idetec

Leds a Arduino

S --> 9 (Led Verde)

VCC (+) -->10 (Led Rojo)

GND (-) --> GND

Conexiones del Sensor LDR Idetec

Sensor LDR a Arduino

S --> A0 entrada analógica

VCC (+) --> 6

GND (-) --> GND

c. Programación del Sistema de Iluminación.

Declaraciones de variables

```
int pin_ldr = A0;                 // Pin para la fotoresistencia
int pin_led_verde = 9;            // Pin al que se conecta el LED Verde
int pin_led_rojo = 10;            // Pin al que se conecta el LED Rojo
int pin_buzzer = 3;               // Pin al que se conecta buzzer
```

```
int pin_vcc_ldr = 6;
                                        //Nota: los pines 9, 10 y 3 tienen salidas
Void setup ()                           analógicas lo cual permite variar los
                                        voltajes de los elementos
{
 Serial.begin (9600);                   // Inicia la comunicación serie para
                                        mostrar datos en el monitor serial
 pinMode (pin_led_verde, OUTPUT);       // Se establece los pines de salida y
                                        entrada
 pinMode (pin_led_rojo, OUTPUT);
 pinMode (pin_buzzer, OUTPUT);
 pinMode (pin_vcc_ldr, OUTPUT);
 pinMode (pin_ldr, INPUT);
 digitalWrite (pin_vcc_ldr, HIGH);      // Se simula 5v en un determinado pin a
                                        falta de otro pin de 5v en el arduino
}

Void loop ()
{
                                        //Se lee el valor de la fotoresistencia
                                        sensor ldr
 int valor_ldr = analogRead (pin_ldr);
 Serial.print ("Valor LDR: ");         // Se mapea el valor que esta entre un
                                        rango 1 [0 - 1023] a un rango 2 [0 - 255]
                                        //ya que las salidas analógicas (pines con
                                        ~) soportan valores entre el rango 2
                                         valor_ldr = map (valor_ldr, 0, 1023, 0,
                                        255);
                                         //Se lo muestra por el puerto serie
 Serial.println (valor_ldr);

                                         //Se escribe la salida en los pines de
                                        ambos leds y el buzzer
 analogWrite (pin_led_rojo, valor_ldr);
 analogWrite (pin_led_verde, valor_ldr);
 analogWrite (pin_buzzer, valor_ldr);
                                          //Se hace una pausa de un segundo
 delay (1000);
}
```

3.6.2. Sistema de Alerta por Temperatura.

En este ejercicio de sistema de alerta por temperatura, el programa monitorea con un sensor de temperatura (DS18B20) el ambiente y si este sensor detecta un incremento de la temperatura mayor a 30°c emita por medio del buzzer un sonido y a su vez encienda el led rojo, caso contrario se mantiene el led verde encendido indicador de que la temperatura sensada del DS18B20 sea la adecuada.

a. Diagrama de conexiones.

Para comprender esta aplicación se realizó el diagrama de conexiones para la transferencia de las señales a los diferentes dispositivos periféricos como se indica en la figura 4.2, donde vemos que se realiza una comunicación con la placa de entrenamiento Arduino la cual alerta de un cambio de temperatura del ambiente a los LEDs, sensor DS18B20 y buzzer.

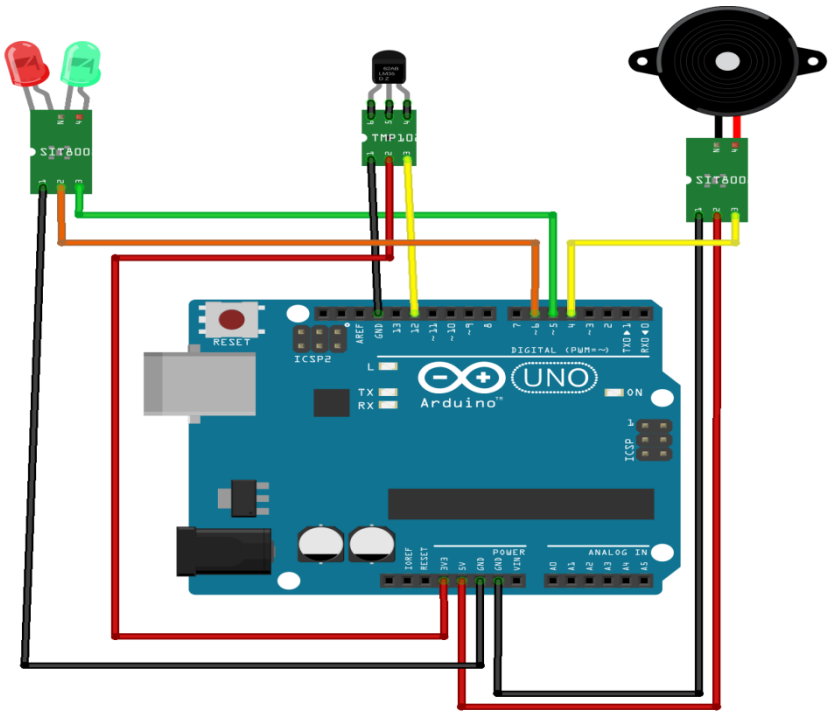

Figura 3. 3: Diagrama de conexiones del sistema de alerta por temperatura.

b. Conexiones de los dispositivos periféricos.

Conexión del buzzer Idetec

Buzzer a Arduino

S --> 4

VCC (+) --> 5V

GND (-) --> GND

Conexión de los leds (Rojo Verde) Idetec

Leds a Arduino

S --> 5 (Led Verde)

VCC (+) -->6 (Led Rojo)

GND (-) --> GND

Conexiones del Sensor DS18B20 Idetec

Sensor a Arduino

S --> 5

VCC (+) --> 3.3V

GND (-) --> GND

c. Programación del sistema de alerta por temperatura.

***Declaracion de librerias**

```
                                        //Se importan las librerías
#include <OneWire.h>
#include <DallasTemperature.h>
```

*Declaración de variables

```
float temp_celsius;
int pin_sensor = 12;                    //pin para el sensor de temperatura
int pin_buzzer = 4;                     //pin del buzzer
int led_verde = 5;                      //pin del led verde
int led_rojo = 6;                       //pin del led rojo
int index_sensor = 0;


OneWire ds(pin_sensor);                 //Se establece el pin declarado como bus
                                        para la comunicación OneWire
DallasTemperature sensors(&ds);         //Se instancia la librería
                                        DallasTemperature
void setup()
{
  Serial.begin(9600);                   //Se abre el puerto para mostrar los datos

  sensors.begin();                      //Se inician los sensores

// Se inicializa el estado de los pines de entrada y salida.
  pinMode(led_rojo, OUTPUT);
  pinMode(led_verde, OUTPUT);
  pinMode(pin_buzzer, OUTPUT);
  pinMode(pin_sensor, INPUT);

}

void loop()
{

sensors.requestTemperatures();          //Prepara el sensor para la lectura
temp_celsius = sensors.getTempCByIndex(index_sensor); // Se obtiene(lee) la
temperatura en Celsius
Serial.print(temp_celsius);             // se colocan los datos en el Puerto para
                                        mostrarlos en el monitor serial
Serial.print("C ");
                                        //Se valida si la temperatura sobrepasa
                                        los 30°C (umbral)
    if (temp_celsius > 30){

      Serial.println("Alerta!!!");
      digitalWrite(led_verde, LOW);
```

```
    alerta(led_rojo);                    //Se llama a la funcion que genera el
                                         sonido del buzzer
       }else{

    Serial.println(" ");
    digitalWrite(led_verde, HIGH);
   }

   delay(1000);                          //Pausa de 1 segundo

}
//Funcion que genera un sonido con el buzzer y enciende un determinado pin de led
//a fin de dar una alerta.

void alerta(int pin_led)
{
  digitalWrite(pin_led, HIGH);          //Enciende un led y el buzzer
   digitalWrite(pin_buzzer, HIGH);
    delay(1000); //pausa 1 segundo

  digitalWrite(pin_led, LOW);           //Apaga un led y el buzzer
  digitalWrite(pin_buzzer, LOW);
  delay(1000); //pausa 1 segundo

}
```

3.6.3. Sistema de Seguridad por Acceso mediante Clave

En este programa se recibe una clave la cual es oculta por asteriscos por seguridad para su posterior verificación dentro del sketch, en el caso de ser correcta se muestra un mensaje indicando "Bienvenido" y se tiene que encender un led Verde, si es incorrecta se mostrará un mensaje indicando "Incorrecto" y encendiendo un led rojo al presionar cada tecla debe sonar el buzzer. Para enviar la clave se presiona la tecla "A" y para borrar hacia atrás se presiona "C".

a. Diagrama de conexiones.

Para comprender esta aplicación se realizó el diagrama de conexiones para la transferencia de las señales a los diferentes dispositivos periféricos como se indica en la figura 3.4 y la figura 3.5, donde vemos que se realiza una comunicación con la placa de entrenamiento Arduino, la cual al ingresar una clave incorrecta por medio del teclado se encienda el led rojo y al presionar las teclas el buzzer emita un tono y al poder ingresar muestre un mensaje en el LCD.

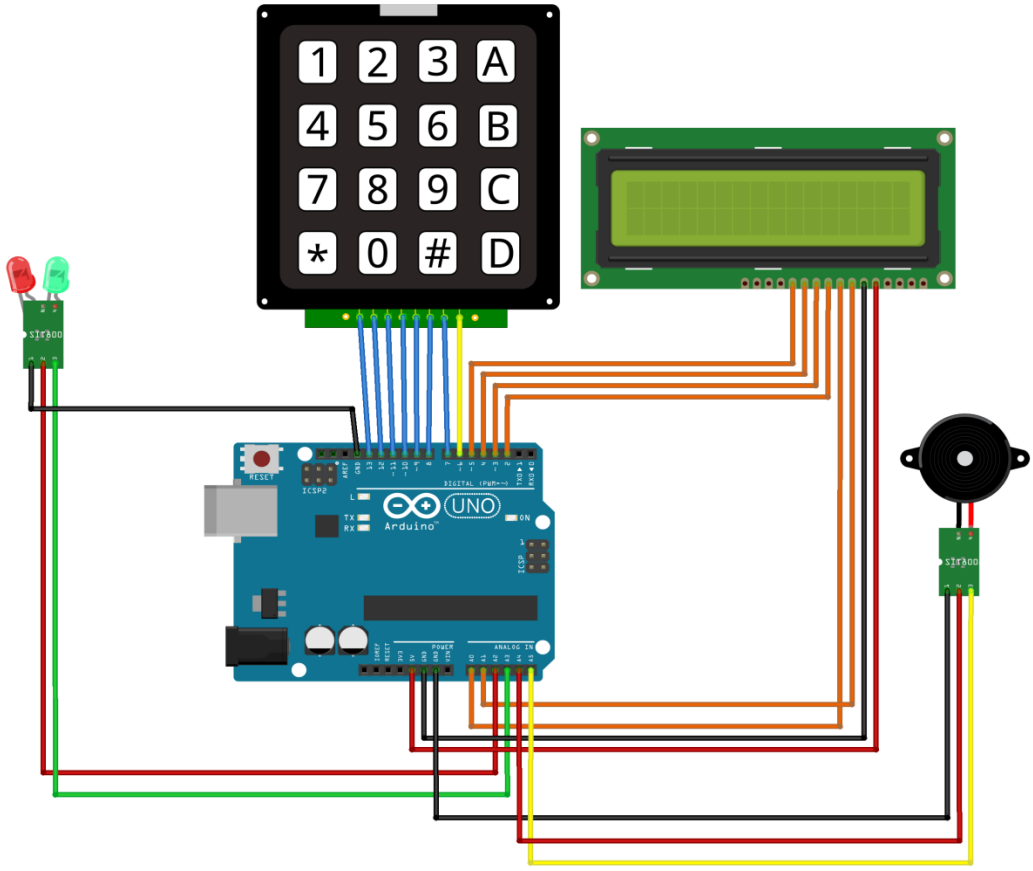

Figura 3. 4: Diagrama de conexiones del sistema de seguridad.

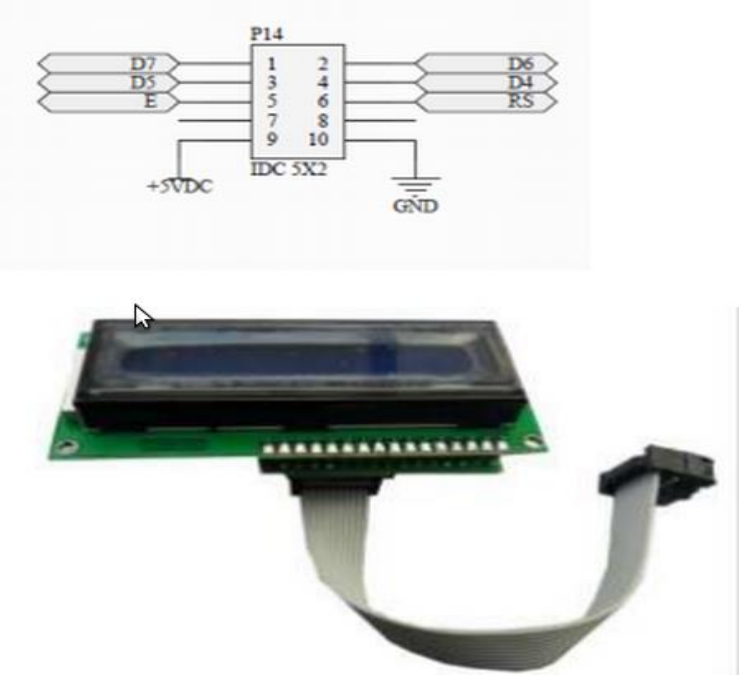

Figura 3. 5: Diagrama de conexiones del LCD.

b. Conexiones para el Sistema de Seguridad por clave.

Conexión del teclado de izquierda a derecha.

Pines del teclado a Pines de arduino

1 --> 13

2 --> 12

3 --> 11

4 --> 10

5 --> 9

6 --> 8

7 --> 7

8 --> 6

Conexiones del LCD Idetec

Pines del LCD a Pines de arduino

D7 --> 5

D6 --> 4

D5 --> 3

D4 --> 2

E -->14(A0)

RS --> 15(A1)

c. Programación para el Sistema de Seguridad por clave.

```
/*  Sistema de Seguridad Por Clave
/* Declaracion de variables
 */
#include <Keypad.h>
#include <LiquidCrystal.h>

const byte FILAS = 4;
const byte COLUMNAS = 4;

/*Conexiones del Teclado

                {'1','2','3','A'}
 Teclado--->{'4','5','6','B'}
                {'7','8','9','C'}
                {'*','0','#','D'}
 Pines Teclado-> 1  2  3  4  5  6  7  8
              | | | | | | | |
             C0 C1 C2 C3 R0 R1 R2 R3

// Se define la Matriz con la apariencia del teclado
char keys[FILAS][COLUMNAS] =

{
  {'1','2','3','A'},
```

```
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

// Se definen los pines que seran columnas
byte colPines[COLUMNAS] = { 13, 12, 11, 10 };

//Se definen los pines que seran filas
byte filPines[FILAS] = { 9, 8, 7, 6 };

//Se definen las constantes de las conexiones del LCD
int D7 = 5;
int D6 = 4;
int D5 = 3;
int D4 = 2;
int E = 14;
int RS = 15;



///// Leds y buzzer ///
int led_rojo = 17;
int led_verde = 16;
int buzzer = 19;
int vcc_buzzer = 18;

String pass_valor;

String pass_asteriscos; // enmascarara con asteriscos a la pass_valor

// Se crea la variable tipo teclado
Keypad teclado = Keypad( makeKeymap(keys), filPines, colPines, FILAS,
COLUMNAS );

// Se crea la variable tipo tipo LCD
LiquidCrystal lcd(RS, E, D4, D5, D6, D7);


void setup()
{
```

```
  Serial.begin(9600);
  lcd.begin(16, 2); //dimensiones del LCD 16 columnas, 2 filas

  pinMode(led_rojo, OUTPUT);
  pinMode(led_verde, OUTPUT);
  pinMode(vcc_buzzer, OUTPUT);
  digitalWrite(vcc_buzzer, HIGH);

}

void loop()
{
 char tecla = teclado.getKey(); // se lee el caracter enviado por teclado

 lcd.setCursor(0,0); // inicia el cursor del LCD en 0,0
 lcd.print("Ingrese Clave:"); //muestra el pass_asteriscos
 lcd.setCursor(0,1); // inicia el cursor del LCD en 0,1
 lcd.print(pass_asteriscos); //muestra los asteriscos

 if( tecla != NO_KEY )  //Se verifica si existe un caracter enviado desde el teclado
 {


  if(isDigit(tecla)) // Se verifica si es un digito
  {
     pass_valor += tecla;
     pass_asteriscos += '*';
  }

  sonidoTecla();

  if(tecla == 'A') // Si presiona A se valida la pass_valor
  {

    if (pass_valor == "1234")
    {

      lcd.setCursor(0,1);
      lcd.print("Bienvenido"); //Muestra el mensaje 'Bienvenido'
      digitalWrite(led_verde,HIGH);
      pass_valor = "";
```

```
      pass_asteriscos = "";
      delay(1500);
      lcd.clear(); // borra la pantala
    }
    else
    {

      lcd.setCursor(0,1);
      lcd.print("Incorrecto"); //Muestra el mensaje 'Incorrecto'
      digitalWrite(led_rojo,HIGH);
      pass_valor = "";
      pass_asteriscos = "";
      delay(1500);
      lcd.clear(); // borra la pantala

    }

     //apaga leds
    digitalWrite(led_verde,LOW);
    digitalWrite(led_rojo,LOW);

  }

  if(tecla == 'C') //Borra el ultimo caracter
  {
    int nuevo_tam = pass_valor.length() - 1; // -1 quita una unidad al tamaño
    pass_valor = pass_valor.substring(0, nuevo_tam); // crea un nuevo string
eliminando el ultimo caracter de la pass_valor
    pass_asteriscos = pass_asteriscos.substring(0,nuevo_tam);// crea un nuevo string
eliminando el ultimo caracter del pass_asteriscos
    lcd.clear(); // borra la pantala
  }

 }

}

void sonidoTecla()
{
 digitalWrite(buzzer,HIGH);
 delay(50);
```

```
  digitalWrite(buzzer,LOW);
  delay(50);
}
```

CONCLUSIONES

1. A través de la descripción de las tarjetas de entrenamiento de Arduino se pudo seleccionar la placa Arduino Uno debido a sus bondades y versatilidad para el desarrollo de diversas aplicaciones y que están disponibles en el mercado electrónico ecuatoriano.

2. Mediante el diseño de las aplicaciones prácticas se pudo comprobar que la tarjeta de entrenamiento de microcontroladores Atmel de Arduino es apropiada para la enseñanza y aprendizaje de otro microcontrolador diferente a los PICs.

REFERENCIAS BIBLIOGRÁFICAS

Candel N., R. (2013). *Plataforma docente para la enseñanza de las TIC basada en la maqueta de un puente colgante.* Proyecto Fin de Carrera. Repositorio Digital de la Universidad Politécnica de Cartagena, España.

Nolivos T., N. T. (2013). *Diseño de un sistema domótico embebido para gestionar la luminosidad*. Tesis de Ingeniería Mecatrónica. Repositorio Digital de la Universidad Tecnológica Equinoccial.

Toledano M., F. J. (2012). *Diseño y construcción de una maqueta para el control semafórico con Arduino.* Proyecto Fin de Carrera. Repositorio Digital de la Universidad Politécnica de Cartagena, España.

Vargas M., F. J. (2012). *Estado del Arte de la Domótica en el 2012 y sus avances en la iluminación LED y alarmas inteligentes.* Grado de Ingeniero Eléctrico. Facultad de Ingeniería de la Universidad de Costa Rica.