# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

# DESARROLLO DE MÓDULOS DIDÁCTICOS BASADOS EN TARJETAS ARDUINO UNO, UTILIZANDO SHIELD ETHERNET, WIFI Y GSM PARA EL LABORATORIO DE MICROPROCESADORES DE LA ESFOT

**TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES**


**BANDA RAMIREZ BYRON MIGUEL**

byron.banda@epn.edu.ec

**REINOSO TARAPUÉS PAMELA FERNANDA**

pamela.reinoso01@epn.edu.ec

**DIRECTOR: ING. FANNY FLORES, MSC**

fanny.flores@epn.edu.ec

**CODIRECTOR: ING. MÓNICA VINUEZA, MSC**

monica.vinueza@epn.edu.ec



**Quito, Noviembre 2018**

# DECLARACIÓN

Nosotros, Byron Miguel Banda Ramirez y Pamela Fernanda Reinoso Tarapués, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

**Byron Miguel Banda Ramirez** **Pamela Fernanda Reinoso Tarapués**

# CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por Byron Miguel Banda Ramirez y Pamela Fernanda Reinoso Tarapués, bajo nuestra supervisión.

| | |
|---|---|
| **Ing. Fanny Flores, MSc** | **Ing. Mónica Vinueza, MSc** |
| **DIRECTOR DE PROYECTO** | **CODIRECTOR DE PROYECTO** |

# DEDICATORIA

*Dedico este trabajo a mi abuelito Jaime Banda el cual me ha ayudado incondicionalmente a terminar mis estudios y me ha impulsado a ser siempre mejor, además de tenerme mucha paciencia en conseguir esta meta, la cual será el principio de muchas más.*

*A mi madre Lida Ramirez y mi hermana Jenny Banda las cuales han sido importantes para mi desarrollo profesional.*

*A todas las personas que directa o indirectamente me ayudaron a superarme, en especial a todos mis compañeros de clases con quienes se vivió gratos momentos.*

Miguel Banda

# DEDICATORIA

*A  mi Padre celestial:*

*Tú, fortaleza mía, mi roca y baluarte.*

*Tu amor y bondad me han permitido alcanzar mis metas y alabarte con ellas, tú qué me escogiste desde el vientre de mi madre, permíteme dedicarte el presente proyecto y el título que obtendré con él, para que todo lo que viene tenga tu bendición.*

*Esto es por ti, mi Señor.*

*Pamela Reinoso*

# AGRADECIMIENTO

*Agradezco primeramente a Dios el cual me ha dado la oportunidad de crecer profesionalmente.*

*A mi familia la cual de una u otra manera ha sido pilar constante en este camino.*

*A todos mis profesores que en diferente medida han puesto en mí su conocimiento y vivencias para ser mejor cada día.*

Miguel Banda

# AGRADECIMIENTO

*Agradezco a mi Dios todopoderoso, que me ha dado el amor de mi familia y mis amigos, la sabiduría, el discernimiento, la perseverancia y todas sus bendiciones que me ayudaron en la culminación de mi carrera.*

*A mis padres, mis abuelitos y mi ñaña, sin su apoyo y ayuda no hubiese alcanzado mis metas y sueños, ustedes me dieron la fuerza y valentía para seguir adelante, como no dedicarles mi profesión, es una bendición tenerlos.*

*A J, tú me acompañaste todas las noches de desvelo, nunca dejaste de recibirme con emoción cuando llegaba a casa y hacías que me olvide de mis problemas, somos los mejores amigos desde el día en que te conocí.*

*A la Escuela de Formación de Tecnólogos de la EPN, a mis profesores quienes con sus conocimientos y sabiduría me guiaron en mi carrera; agradezco a mis amigos, ustedes hicieron de la universidad la mejor experiencia.*

*Y a ti Flaco, tus locuras alivian mi cansancio, tú me acompañas en los momentos más difíciles y nunca me dejas declinar, contigo entendí las palabras: "quien tiene un amigo, tiene un tesoro", a ti gracias infinitas.*

*No ceso de dar gracias por vosotros, haciendo memoria de vosotros en mis oraciones.*

*Efesios 1:16*

Pamela Reinoso

# ÍNDICE

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# RESUMEN

El presente proyecto tiene como finalidad modernizar el laboratorio de Microprocesadores, proporcionando a los estudiantes de la ESFOT placas didácticas avanzadas que incorporen un módulo Arduino UNO como elemento central, además de módulos complementarios como son: *Ethernet*, *WiFi* y GSM.

Esta placa incorpora también elementos básicos, que son objeto de estudio en los primeros niveles, como son: resistencias, diodos LED, *display* de 7 segmentos, pulsadores, entre otros. Estos elementos permiten al estudiante familiarizarse paulatinamente con el uso de la placa didáctica, empezando con prácticas sencillas como son encender un diodo LED o un *display* de 7 segmentos; y posteriormente realizar prácticas que conlleven el uso de módulos más avanzados sin mayor dificultad.

Complementando a la construcción de las placas, se desarrollan cuatro prácticas que constan de dos partes. Una parte orientada al docente, con todo el programa completo y comentado con ejemplos de conexiones. La otra parte está orientada al estudiante, cuyo propósito es prepararlo por medio de consultas sencillas sobre el tema a desarrollar durante la práctica.

Finalmente, todo el proceso de construcción, así como las pruebas de funcionamiento están explicadas detalladamente.

Palabras clave: placa didáctica, Arduino UNO, módulos *shield*, módulo GSM, módulo *Ethernet*, módulo *WiFi*.

# ABSTRACT

*The following project aims to modernize the Microprocessors laboratory, providing ESFOT students with advanced didactic boards that incorporate an Arduino UNO module as the main element, in addition to complementary modules such as: Ethernet, WiFi and GSM.*

*This board also incorporates basic elements, which are at the focus of studies during the first semestres, such as: resistors, LEDs, 7-segment display, pushbuttons, among others. These elements allow students to gradually become familiar with the use of the didactic board, beginning with simple practices such as turning on a LED or a 7-segment display and then carrying out practices that involve the use of more advanced modules without much difficulty.*

*Complementing the construction of the boards, four practices are designed, which consist of two parts. One part is addressed to teachers, including the whole program and commented with examples of connections. The other part is addressed to students; its purpose is to prepare them through simple research on the subject to develop during the practice.*

*Finally, the whole construction process as well as the performance tests are explained in.*



*Keywords: didactic board, Arduino UNO, shield modules, GSM module, Ethernet module, WiFi module.*

# 1. INTRODUCCIÓN

En la actualidad, en las carreras de Electrónica y Telecomunicaciones, al igual que Electromecánica de la Escuela de Formación de Tecnólogos, se imparten asignaturas tales como: Microprocesadores, Instrumentación Electrónica y Control con Microprocesadores [1]. Para el desarrollo de las prácticas de laboratorio de dichas asignaturas, se utiliza el laboratorio de Microprocesadores, el cual no cuenta con recursos pedagógicos modernos para el estudiante, y los escasos materiales que dispone como: grabador de PICs, cables, *protoboards*, entre otros, se encuentran en mal estado o no son suficientes para abastecer la cantidad de alumnos. Además, de acuerdo con el PEA (Plan de Estudios por Asignatura) de las materias de Microprocesadores y Control con Microprocesadores, se usa como microcontrolador base al PIC, particularmente al PIC 16F870 [2] [3]. Dicho microcontrolador es básico y conveniente para empezar; sin embargo, este no ha presentado un desarrollo significativo con el pasar del tiempo.

Los PICs son microcontroladores, que nativamente fueron desarrollados para ser programados con lenguaje Ensamblador; además, en las asignaturas correspondientes, aún se continúa impartiendo el manejo de este lenguaje. *Assembler* resulta limitado y complejo, con respecto a nuevos lenguajes de programación. *Assembler* presenta una estructura de programación orientada al manejo de bits, que resulta un problema en programas de mediana y alta complejidad debido a la exagerada cantidad de líneas de código, a comparación con lenguajes de alto nivel. Esto, en ocasiones puede provocar en el estudiante horas de trabajo injustificado debido a que en lenguajes de alto nivel se puede obtener la misma solución en un tiempo menor e incluso un mejor rendimiento debido a la sencillez que brindan los entornos actuales de programación, dando como resultado mayor motivación a seguir explorando las infinitas posibilidades de aplicaciones que se pueden crear.

Con los avances de la tecnología en el campo de la electrónica, en el año 2005 se lanzan plataformas de código abierto basadas tanto en *hardware* como en *software*

de fácil uso, conocidas como “módulos o placas Arduino” [4]. El módulo Arduino Uno es el más utilizado por estudiantes que dan sus primeros pasos en la programación de microcontroladores [5]. Arduino Uno, posee 16 veces más memoria flash que el PIC16f870, lo que implica almacenar más líneas de código para programas más pesados. Tiene 6 veces más salidas PWM *(Pulse-width modulation*) para usos de regulación de potencia de una señal digital; 15mA más de corriente a la salida de cada pin para cargas que exigen más potencia. Además, 16 veces más memoria RAM, la cual permite mayor carga de tareas, agilizando de mayor manera los procesos a realizar. Cuenta con 16 veces más memoria EEPROM, la cual permite almacenar mayor cantidad de datos, los cuales no se borrarán cuando se retire la alimentación. Finalmente, comunicaciones UART TTL, usadas en muchos dispositivos como módulos *WiFi*, *Bluetooth*, GSM, entre otros.

Mediante el bus I2C, es posible comunicar dos módulos Arduino o dispositivos que soporten este tipo de comunicación. SPI es utilizado en comunicaciones *full-duplex* con dispositivos que lo soporten. Por lo presentado, se observa que el PIC16f870 se ha quedado desactualizado frente a nuevas tecnologías que sobrepasan el límite tecnológico para la creación de *chips* [6]. (Para más detalles del módulo Arduino Uno ver, Anexo A1).

Las tarjetas Arduino utilizan un lenguaje amigable con el programador, evitan líneas de código que tienen nombres de posiciones de memoria que contienen instrucciones o datos, como *Assembler*, lo cual hace que programar sea más productivo y más corto en cuanto a líneas de código. Brinda al usuario, facilidad y accesibilidad al momento de utilizarlo [7]. Estas herramientas están dirigidas a los estudiantes que no tienen experiencia en electrónica o en programación de microcontroladores; pero también brinda facilidades de programación a usuarios que ya poseen conocimientos avanzados en este campo [8]. Con el pasar de los años, estas tarjetas se han implementado en miles de proyectos, que van desde simples prácticas de laboratorio, hasta en grandes empresas que han optado por esta nueva tecnología, ya que tan pronto como llegó al mercado se implementó

rápidamente y se adaptó a nuevos desafíos y necesidades, abarcando el comercio de los microcontroladores [8].

Con la carencia innovativa del PIC y con nuevas tecnologías en auge, se pretende actualizar el laboratorio de Microprocesadores, con módulos didácticos cuyo núcleo es la tarjeta Arduino Uno, acompañado de elementos electrónicos pasivos como: diodos LED, *dip-switch*, pulsadores, *displays* de 7 segmentos y LCD. También se añadirán módulos como: módulo *Ethernet*, módulo GSM y módulo *WiFi*. Estos módulos didácticos ayudarán al estudiante a adquirir nuevos conocimientos y destrezas, en base a recursos modernos que se pueden encontrar en el campo laboral y comercial.

## 1.1 MARCO TEÓRICO

Cada placa didáctica está conformada por elementos básicos y avanzados, centralizados con un módulo Arduino Uno. Dicho módulo, comanda todo, excepto el módulo *WiFi*, el cual es un módulo independiente cuya relación con el Arduino es el de puente para concretar la comunicación serial con la PC. El módulo *WiFi* posee su propio microcontrolador, capaz de interpretar el código realizado en el IDE de Arduino.

- **MÓDULO ARDUINO UNO**

Originalmente constituye una unidad autónoma, que posee elementos ya instalados en la placa como: diodos LED, microcontrolador Atmega 328, cristal de cuarzo, puerto USB, entre otros. Ver Figura 1.1 Partes del Arduino UNO. Por sí solo, se podrían desarrollar prácticas básicas, pero en la placa didáctica se trata de usar todo el potencial que ofrece con el uso de la mayoría de pines. Las características más importantes son: 14 pines digitales I/O, 6 salidas PWM, 6 entradas analógicas, 32kbytes de memoria flash [9]. Para más detalles, ver el Anexo A1.

Un factor importante del módulo Arduino Uno es que el microcontrolador es desmontable en caso de requerir solo el microcontrolador o si se desea reemplazarlo por daño. Además, el diseño de los pines del Arduino permite que su uso no sea solo mediante cables, sino también a través de módulos *shield*, los cuales son similares al Arduino en dimensiones para que calcen en todos sus pines. Estos módulos *shield* proveen aplicaciones adicionales al Arduino, y entre algunos de ellos se puede mencionar: *Ethernet*, *WiFi*, GSM, etc. La disposición de los pines entre el Arduino y el módulo *shield*, permite optimizar el tiempo necesario en realizar conexiones; priorizando así la programación. En algunos casos, se requiere de pocas conexiones.


*Figura 1. 1 Partes del módulo Arduino Uno* [5]

- **ATMEGA 328**

Es un microcontrolador de la familia ATmega que sirve como núcleo del Arduino. Tiene una arquitectura de 8bits y bajo consumo de energía (1.8V a 5.5V). Además,

es muy utilizado para aplicaciones pequeñas; posee 28 pines, de los cuales 23 son I/O y 6 canales PWM [10]. Para más información ver el Anexo A2.

- **MÓDULO *SHIELD***

Un módulo *shield* posee las mismas dimensiones que el Arduino con el único propósito de unir los pines del Arduino con el del módulo para formar un solo bloque o elemento [11].

Por lo general, los módulos poseen espadines hembra con patas más largas de lo normal para que encajen en los espadines hembra del Arduino, como se muestra en la Figura 1.2 Módulo *Shield.*

La razón principal de escoger este elemento, es predisponer al usuario en el uso y manipulación de módulos *Shield*, los cuales se encuentran abundantes en el mercado local y la mayoría son de bajo costo con la ventaja de no ser necesario interconectarlos con cables, facilitando en gran medida su uso. La segunda razón es que facilita de gran manera con la reducción de pistas que interconectarían estos módulos dando como resultado una placa didáctica más compacta y acorde con la realidad de funcionamiento de los módulos.

*Figura 1. 2 Módulo Shield* [12]

- **MÓDULO ETHERNET SHIELD**

Este módulo permite utilizar funciones de red cableada 10/100 *Ethernet*, respetando los estándares IEEE 802.3 10-BASE-T y 802.3u 100BASE-T compliant. Soporta protocolos TCP, UDP, IPv4, ICMP, ARP, IGMP y PPPoE. Posee 16Kbytes de *buffer*. Utiliza conectores RJ45; adicionalmente, en algunos de los módulos *Ethernet* se puede colocar cualquier dirección MAC [13], pero en otros viene especificada la MAC en la placa. Para más detalles ver el Anexo A3.

Los pines que utiliza este módulo son los digitales: 10-11-12-13 [14]. Estos pines están marcados en el módulo con una línea debajo del número; el resto de pines son solo una extensión del pin original del Arduino, como se aprecia en la Figura 1.3 Módulo *Ethernet Shield*.


*Figura 1. 3 Módulo Ethernet Shield* [14]

- **MÓDULO GSM SIM800C *SHIELD***

Este módulo permite expandir a operaciones de comunicación móvil como son: SMS (*Short Message Service*) y llamadas. Entrega señales GSM / GPRS850 / 900

/ 1800 / 1900MHz para servicio de Audio, SMS y GPRS; y en versiones más actuales, navegación en Internet por paquete de datos. Este *Shield* tiene bajo consumo de energía de alrededor de 0.6mA en modo de reposo; además, cumple con las normas GSM fase 2/2+: clase 4 (2 W @850/900 MHz), Clase 1 (1 W @ 1800 / 1900MHz). Opera con una tensión de 5-12V proporcionada por el Arduino o por una fuente externa y necesariamente usa una tarjeta *microSIM* de cualquier operadora [15]. Para más información revisar el Anexo A4.

Posee una antena pequeña, usualmente pegada a los pines en forma de cable, que no debe ser extraída si se encuentra en funcionamiento. Posee un *switch* que selecciona si la alimentación se la realiza por medio del Arduino o una fuente externa [15]. Ver Figura 1.4 Módulo GSM SIM800c *Shield.*

Para el funcionamiento posee un pulsador, el cual es la inicialización del módulo para engancharse a la red móvil. Posteriormente, unos LEDs informarán el estado de la conexión; si es mayor a 3 pulsos por segundo no se puede enganchar a la red, si es menor a 1 pulso por segundo la conexión es exitosa. Este módulo utiliza 2 pines a elección del usuario, los cuales son pines seriales, uno de transmisión y otro de recepción; el resto son extensiones de los pines originales del Arduino.

*Figura 1. 4 Módulo GSM SIM800c Shield* [15]

- **MÓDULO *WIFI* ESP8266**

Es un módulo muy parecido al módulo *Bluetooth*. Incluye toda la electrónica necesaria para la comunicación en Radio Frecuencia en la banda de *WiFi*, así como también la pila de TCP/IP que se comunica con el usuario a través de un puerto serie. Básicamente, es un microcontrolador que tiene como objetivo principal brindar acceso a una red *Ethernet* que opera en el protocolo 802.11 b/g/n y con un voltaje de 2.5V a 3.6V [16]. Para más información revisar el Anexo A5.

Este módulo funciona con máximo 3.6V, por lo que se debe acoplar la tensión de alimentación por *USB* de la PC de 5V. Para ello, se utiliza un regulador de 3.3V para garantizar el correcto funcionamiento. Este no es un módulo *Shield*, por lo cual en la placa didáctica tiene un comportamiento especial; es decir, trabaja autónomamente con sus propios pines de I/O (2 pines), cuya relación con el Arduino es solo de puente para comunicación serial entre el ordenador y el ESP8266 [16] Ver Figura 1.5 Módulo ESP8266.


*Figura 1. 5 Módulo ESP8266* [16]

# 2. METODOLOGÍA

En el presente proyecto se utiliza metodología aplicada, debido a que se usan conocimientos adquiridos durante la carrera y en el proceso de investigación para el desarrollo de los módulos. Además, el interés del proyecto apunta a que los estudiantes del laboratorio de Microprocesadores apliquen, en la práctica, los conocimientos adquiridos en las clases teóricas.

Para el desarrollo de los módulos didácticos se usa metodología experimental, debido a que se realizan prototipos y pruebas previas que, permiten obtener el módulo deseado, con las prestaciones, tamaño, diseño y funcionalidad requeridos.

El desarrollo de los módulos sigue varios pasos que se describen a continuación:

Se determinan los componentes que irán en el módulo didáctico tomando en cuenta que se utilizará el módulo Arduino Uno, por ser el módulo más comercial [6], se considera apropiado utilizar módulos complementarios para mejorar la experiencia y aprendizaje del estudiante. Estos módulos desmontables son: GSM y *Ethernet* que tengan características desmontables con el Arduino Uno porque sus pines encajan con los del mismo, es decir *"Módulos Shield"*. En cuanto al módulo *WiFi*, no se considera un módulo *Shield* debido a su elevado costo; sin embargo, existe una alternativa de bajo costo con el módulo ESP8266. Dicho módulo ofrece el mismo funcionamiento, con la característica de ser más pequeño y autónomo; es decir, no necesita al Arduino para funcionar, pero necesita un convertidor serial que este posee para cargar el programa directamente en el ESP8266, por lo que necesita conectarse al Arduino, más no para controlarlo. Aparte de los módulos mencionados, se consideraron otros elementos como: diodos LEDs, pantalla LCD, pulsadores y *dip-switch*.

Se procede a realizar el esquema del circuito del módulo, tomando en cuenta las características físicas del Arduino y los otros módulos *shield*. Programas como *Ares* y *Proteus* ayudan en el diseño del circuito que se desea implementar.

Se crea el módulo didáctico físicamente; se realiza la *"quema de la baquelita"*, el cual consiste en pasar el circuito diseñado en papel a una lámina de cobre, dejando únicamente las pistas que unen todos los elementos al Arduino. Con la placa de cobre lista, se procede a soldar los elementos.

Con ayuda del programa *Arduino Sketch,* se realizan programas para comprobar el funcionamiento de los módulos, además se crean las cuatro prácticas de laboratorio, las cuales serán guiadas por el maestro de la asignatura. Para finalizar, se realiza una inspección de *hardware* y se comprueba su funcionamiento, implementando las prácticas desarrolladas anteriormente.

# 3. RESULTADOS Y DISCUSIÓN

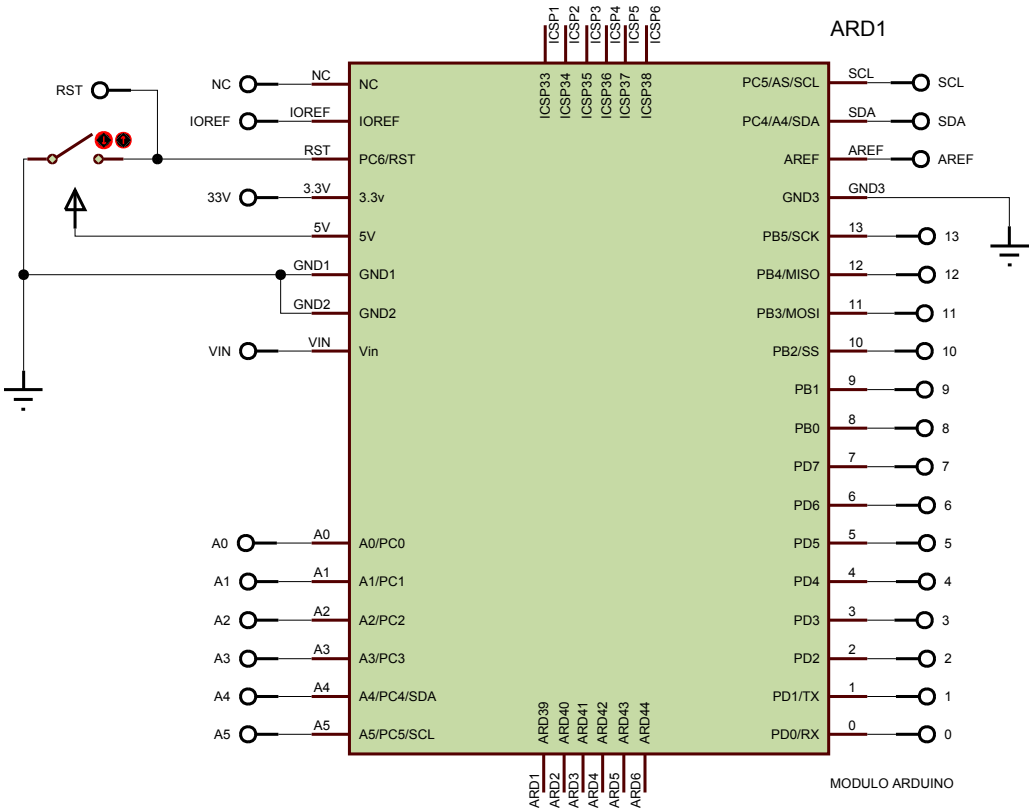
## 3.1 DISEÑO DE LOS MÓDULOS DIDÁCTICOS

Con los elementos que conformarán la placa didáctica, se procede a realizar un esquema lógico. Dicho esquema permite interconectar todos los componentes entre sí, sin importar las dimensiones físicas sino el funcionamiento lógico de la placa. Debido a que posee varios elementos, es necesario organizarlos de tal manera que sea fácilmente diferenciable para el usuario de la placa. Para este caso en particular, se puede agrupar todos los elementos en 8 subcircuitos, los cuales en el momento del funcionamiento interactuarán alternadamente con el módulo Arduino como el elemento central del funcionamiento, lo que da un total de 9 subcircuitos. Para crear este esquema se utilizará el *software* ISIS *7 Profesional*. Ver Figura 3.1.

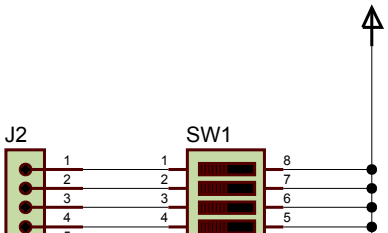
Después de crear un esquema lógico, las dimensiones físicas toman relevancia debido a que cada elemento electrónico posee características únicas, las cuales impactarán en el tamaño total de la placa.

DISPLAY 7SEG

# MÓDULO ARDUINO




# DIPSWITCHS



El *software* ISIS *Ares 7 Profesional* posee una gran cantidad de encapsulados y dimensiones reales de cada elemento; no obstante, no existen todos los elementos debido a que los fabricantes siempre tratan de mejorar o cambiar de tecnologías.

En algunos casos, las dimensiones cambian drásticamente del encapsulado; y en otros casos, crean nuevos elementos que en el momento que se lanzó el *software* a la venta, no existían y por lo tanto se ve la necesidad de crear los encapsulados de algunos elementos tales como: *buzzer*, *switches*, *displays* LCD 16x2, *display* 7 segmentos, módulos *WiFi*, GSM, *Ethernet* y Arduino; entre otros. Ver Figura 3.2.

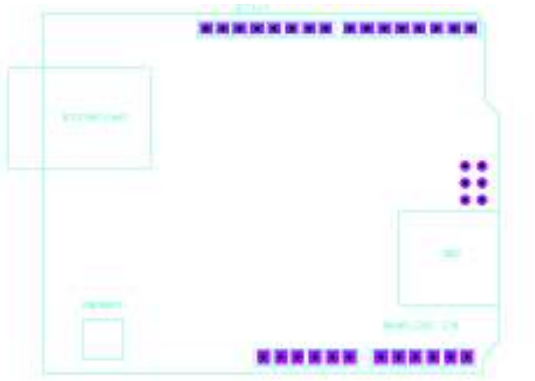


*Figura 3. 2 Creación del PCB del Módulo Arduino Uno*

Una de las ventajas de agruparlos en subcircuitos, es que al crear el esquema físico, están bien organizados para poder ubicarlos lo más cerca posible para optimizar el tamaño y materiales. Esto es posible en el *software* ISIS *Ares Professional* 7, ubicándolos de manera que el módulo Arduino esté en el lugar más céntrico para que el resto de subcircuitos se interconecten con facilidad. Ver Figura 3.3

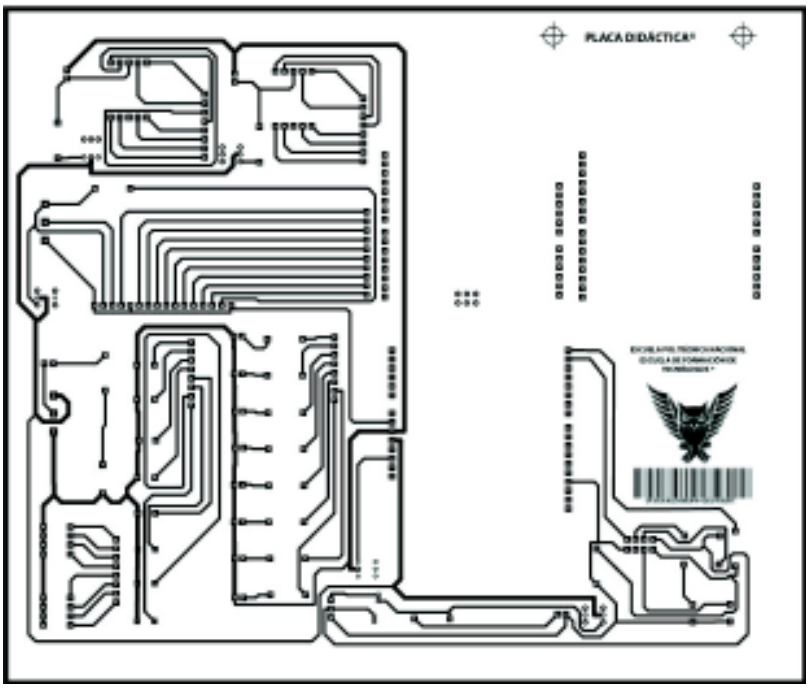



*Figura 3. 3 Diseño final de las placas didácticas en Ares*

Adicionalmente se creó una pegatina, la cual se puede observar en la Figura 3.4, que ayudará a los usuarios en la identificación de los elementos.

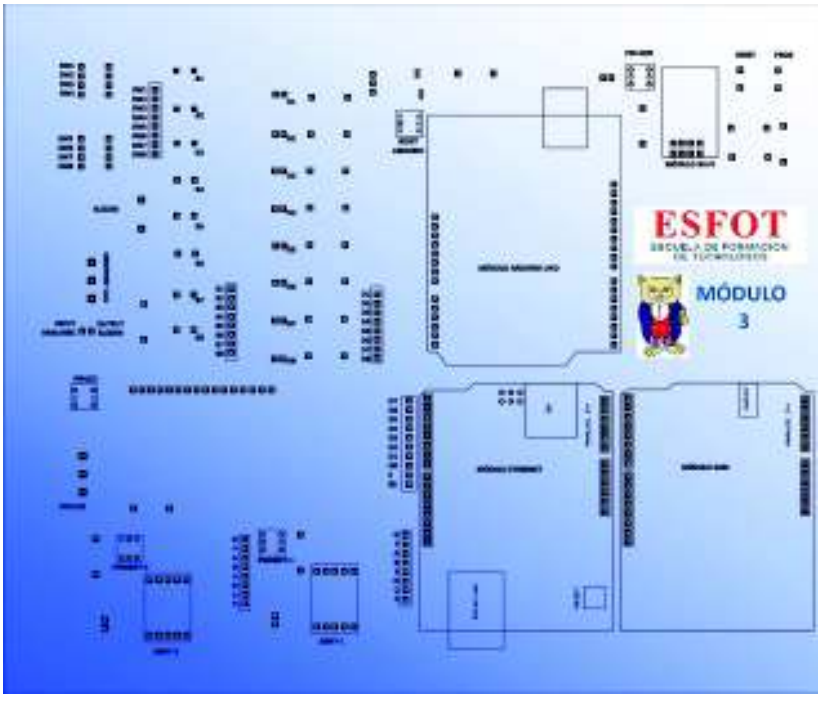



*Figura 3. 4 Pegatina para módulo didáctico*

## 3.2 IMPLEMENTACIÓN DE LOS MÓDULOS DIDÁCTICOS

Una vez realizado el esquema de circuito del módulo, se procede con la implementación del circuito, esto hace referencia a la creación física del módulo, para lo cual se procede de la siguiente manera:

- Se imprime el diseño final del circuito (Figura 3.3) en papel termotransferible (ver Figura 3.5), este papel permite transferir su contenido a cualquier superficie mediante la aplicación de calor [17].

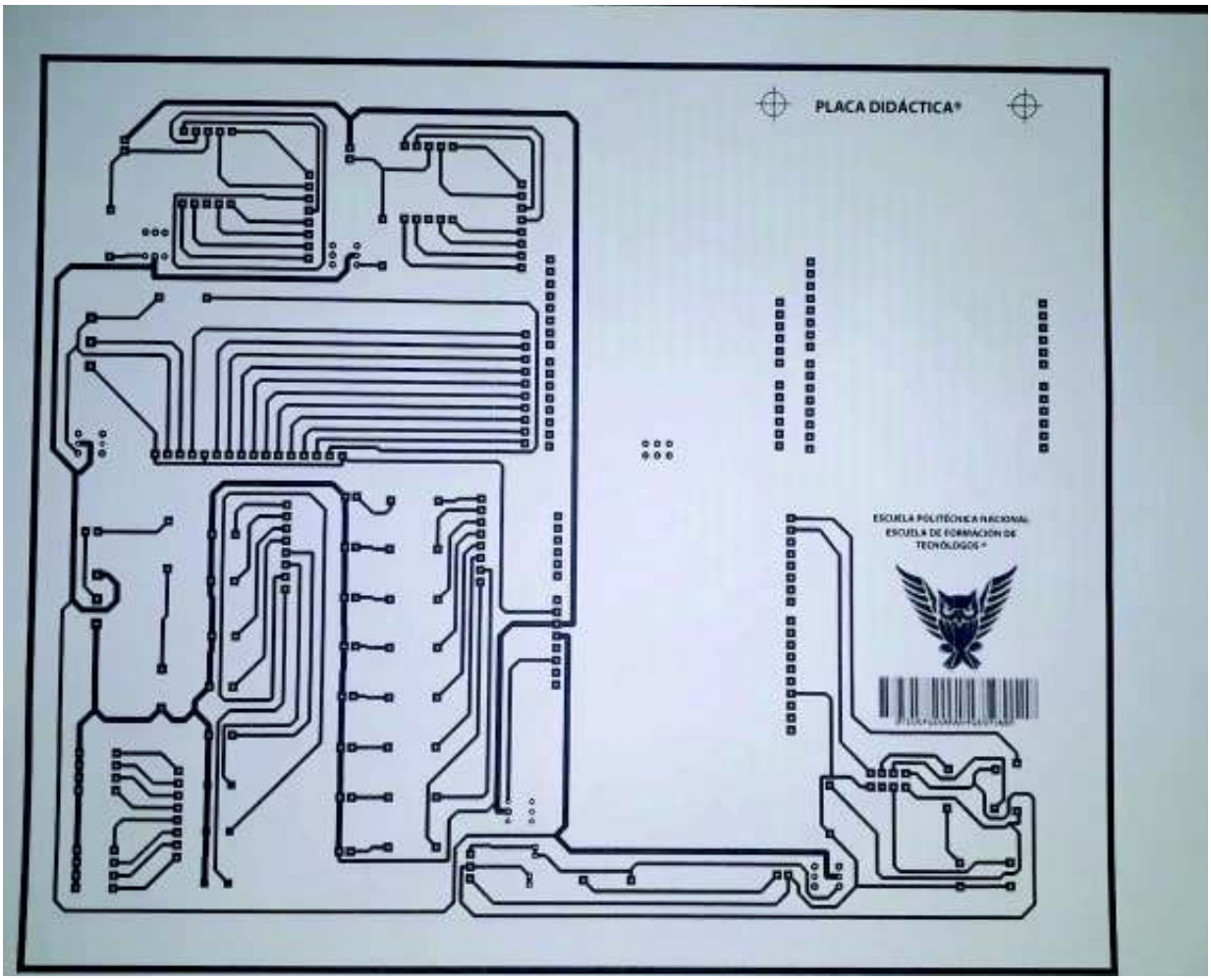



*Figura 3. 5 Impresión del circuito en el papel termotransferible*

- Una vez que se tenga la impresión del circuito, en el papel termotransferible, se coloca sobre el lado de cobre de la baquelita y se procede a planchar. Es importante que la plancha se encuentre caliente, eso ayudará a transferir con facilidad la impresión de las pistas en la baquelita.

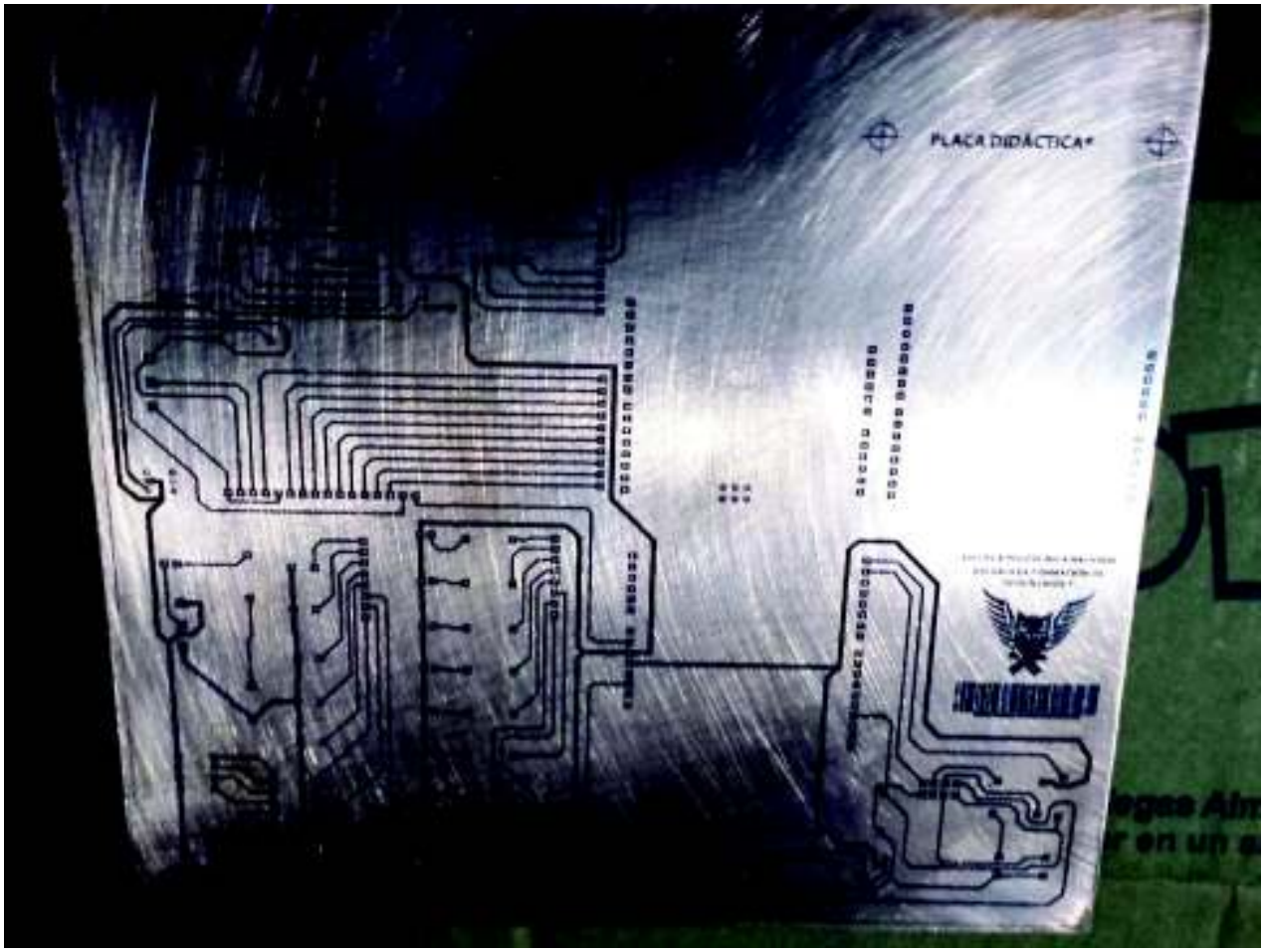



*Figura 3. 6 Pistas del circuito impresas en la baquelita*

- Cuando todas las pistas se encuentren impregnadas en la baquelita (ver Figura 3.6), se procede a *"quemar la baquelita"*, esto consiste en: colocar la baquelita en agua hirviendo y poco a poco añadir percloruro férrico [17] (ver Figura 3.7), ésta solución ácida elimina el exceso de cobre, dejando únicamente las pistas como se muestra en la Figura 3.8.

*Figura 3.7 Baquelita en solución ácida*

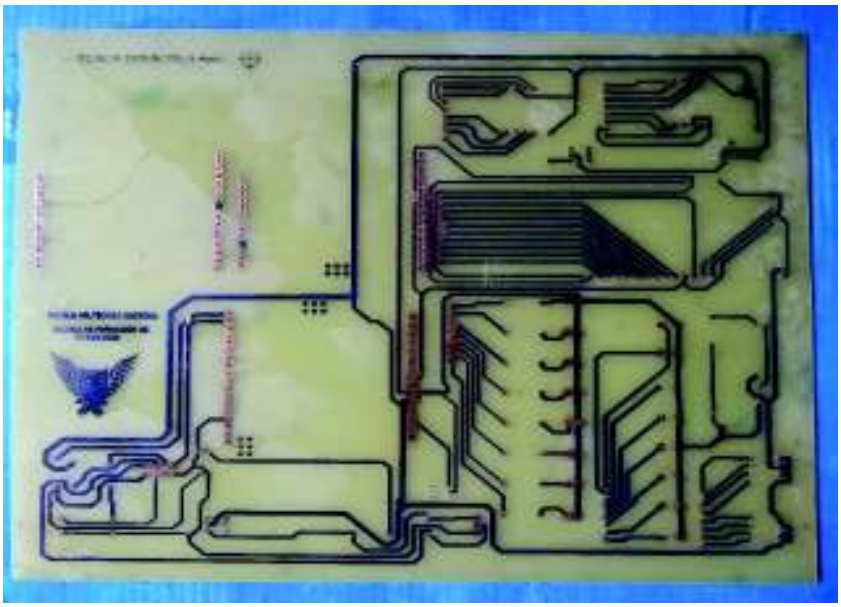


*Figura 3.8 Pistas de cobre después de quemar la baquelita*

- Se retira la tinta negra con una lija, dejando al descubierto el cobre y se procede a taladrar los agujeros donde, se insertarán los elementos. Ver Figura 3.9.

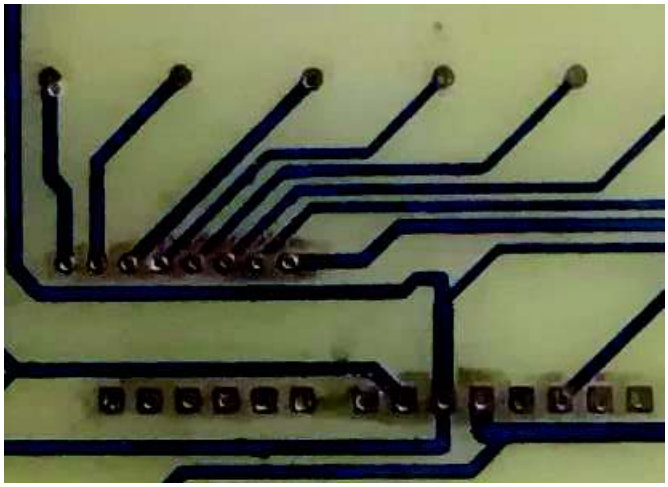


*Figura 3. 9 Perforación de la placa*

- Una vez que se tenga todas las perforaciones, se coloca la pegatina y se comienza a soldar los elementos (diodos LED, *dip-switches*, pulsadores, *displays* de 7 segmentos, potenciómetros, *buzzer*, *display* LCD 16x2 y espadines para módulos), como se muestra en las Figura 3.10 y 3.11.

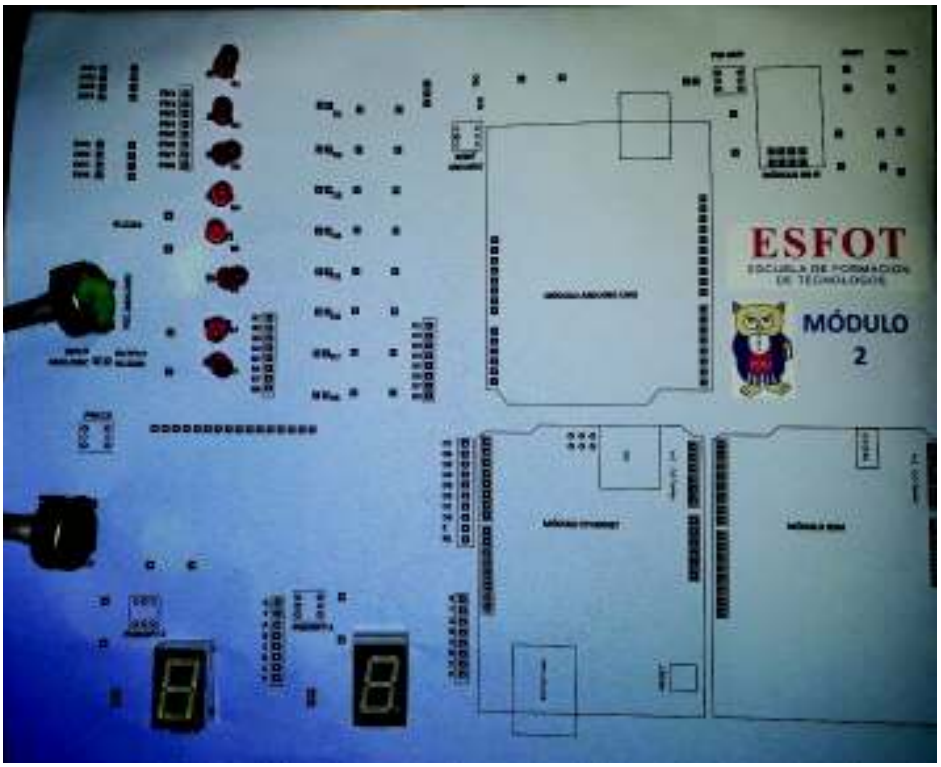



*Figura 3. 10 Placa con pegatina y elementos para soldar*

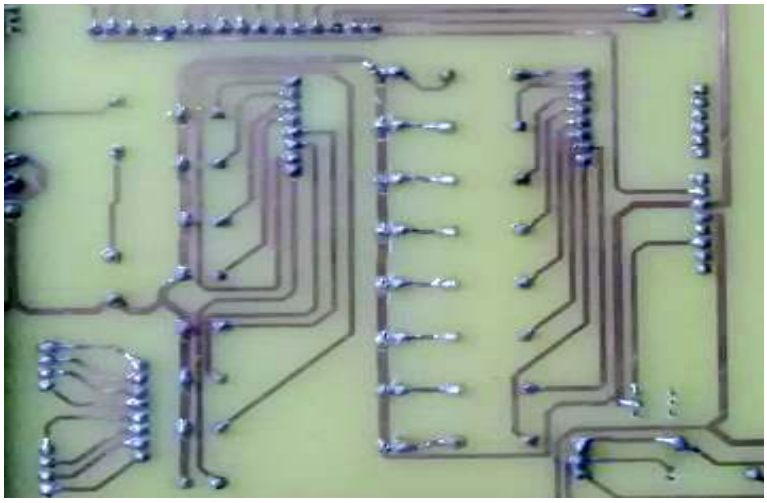


*Figura 3. 11 Elementos soldados*

- La Figura 3.12 presenta la distribución de los elementos en la placa didáctica

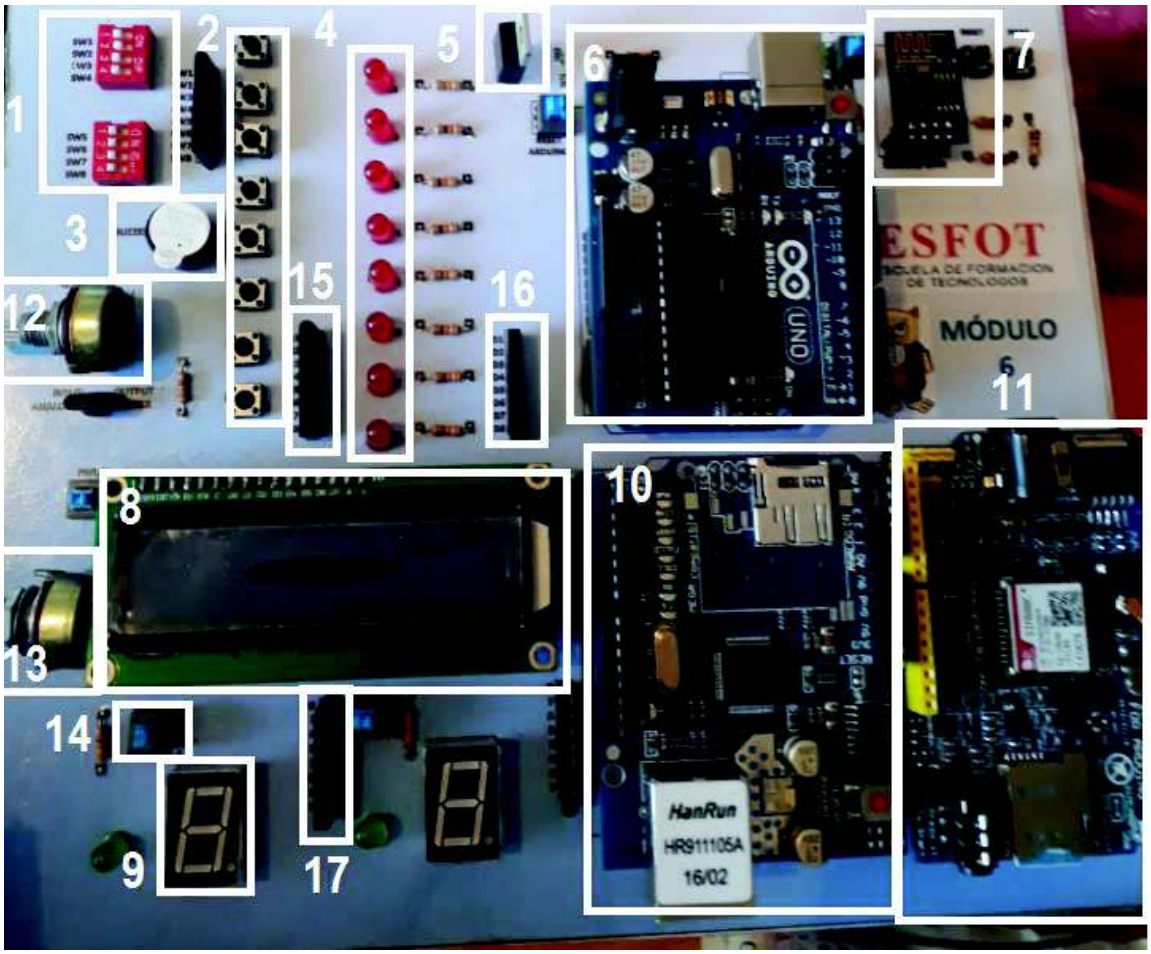



*Figura 3. 12 Elementos de la Placa didáctica*

1. *Dip-switchs* de 4 vías
2. Pulsadores
3. *Buzzer*
4. Diodos led
5. Regulador de voltaje de 3.3V
6. Arduino UNO
7. Módulo ESP8266
8. *Display* LCD 16x2
9. *Display* 8 segmentos
10. Módulo *Ethernet*
11. Módulo GSM Sim800c
12. Potenciómetro para propósito general
13. Potenciómetro para el contraste del LCD
14. *Switch* para encender el *display* de 7 segmentos
15. Espadines para habilitar los pulsadores
16. Espadines para habilitar los diodos LED
17. Espadines para habilitar *Display* 8 segmentos

**DIMENSIONES:**

20cm x 21cm x 2.5cm

**VOLTAJE DE OPERACIÓN DE LA PLACA DIDÁCTICA:**

4 .5v – 5.5V

**OBSERVACIONES:**

-Los elementos desmontables son: Módulos *Ethernet*, *WiFi*, Arduino Uno, GSM y el *display* LCD.

-Todos los elementos básicos como: diodos LED, pulsadores, *display* LCD, *displays* 7 segmentos, *dip-switchs, buzzer* y potenciómetro poseen espadines hembra los cuales permiten la fácil conexión con el Arduino.

-Los elementos: *display* LCD, *displays* de 7 Segmentos y el módulo *WiFi* poseen un *switch* de encendido para que funcionen.

-El Arduino posee un *switch* de *reset* que se usará únicamente cuando entre en operación el módulo *WiFi*. El cual se lo deja accionado hasta terminar de usar el modulo WiFi.

-La placa didáctica posee una estructura deslizante, como se muestra en la Figura 3.13 la cual, permite guardar los cables de *protoboard*, cable USB-AB para alimentación y transferencia de información del Arduino Uno y extensor de USB.

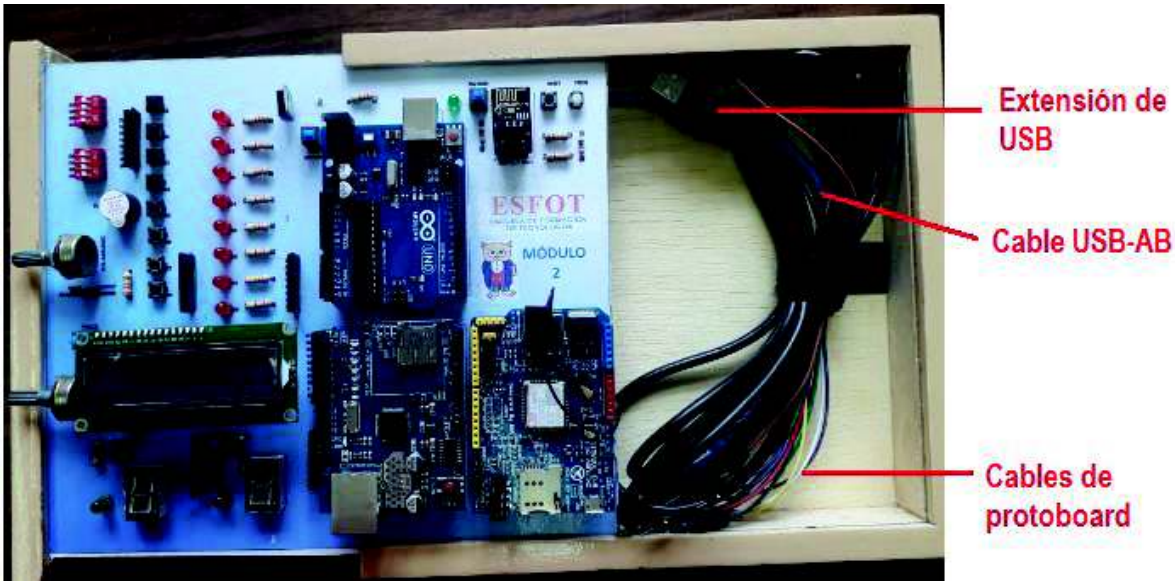



*Figura 3. 13 Estructura deslizante de la placa didáctica*

## 3.3 PRÁCTICAS DE LABORATORIO DESARROLLADAS

Con los módulos ya listos, se procede a la creación de las prácticas de laboratorio. Dichas prácticas abordarán temas relacionados con el PEA de la materia de Microprocesadores como: manejo de puertos de entrada/salida digitales y analógicas, PWM, temporizadores, interrupciones, uso de pantalla LCD; adicionalmente se impartirán prácticas con los módulos *WiFi*, *Ethernet*, serial y GSM. A continuación, se presentan las prácticas de laboratorio desarrolladas.

## PRÁCTICA 1: MANEJO DE PUERTOS DE ENTRADA Y SALIDA UTILIZANDO EL MÓDULO ARDUINO UNO.

Objetivo: Manejar puertos de entrada /salida digitales y analógicas, utilizando el IDE de Arduino para encender/apagar diodos LED y *display* de 7 segmentos a través de *dipswitch* y potenciómetro**.**

*Sección 1*

La primera sección consiste en el manejo de una entrada analógica y seis salidas PWM y digitales.

Mediante la manipulación del potenciómetro conectado al pin A3 del Arduino Uno, se visualiza la variación en el comportamiento de los LEDs conectados a los pines: 11, 10, 9, 6, 5, 3 del Arduino Uno. También se puede observar qué juego de luces se muestra en los LEDs en un *display* LCD conectado a los pines: 2, 4, 7, 8, 12, 13. Para el desarrollo de esta práctica, los elementos del módulo didáctico que se utilizan son: cables de *protoboard*, LEDs, potenciómetro, *display* LCD. En cuanto al *software*, se utiliza: *Arduino,* programa proporcionado por los creadores de las placas Arduino. La programación correspondiente a esta sección se la puede encontrar en hoja guía del instructor en el Anexo B 1 y la correspondiente hoja guía del estudiante de esta práctica se presenta en el Anexo C 1.

A continuación, en la Figura 3.14, se presenta el diagrama de flujo del programa utilizado para esta sección. Inicialmente se declaran las variables a usarse como: *cons* e *int*, las cuales serán usadas para las salidas PWM, para asignar un valor de 0 a 255 que serán observadas en la intensidad luminosa de los diodos LEDs, una variable guardará el valor de la entrada del potenciómetro para mostrarla en el LCD y se declara las variables que usará el LCD. En la función *void setup¸* se declara el modo de los pines a usarse ya sean de entrada o salida, también se escribe el mensaje de bienvenida en el LCD. En *void loop,* se configura la variable

del potenciómetro para que guarde su variación, si el potenciómetro muestra un valor inferior a 600, se activará el primer juego de luces, caso contrario, es decir si tiene un valor superior a 600, se activará el segundo juego de luces y el valor del potenciómetro se mostrará en el LCD.

*Sección 2*

En esta sección se analiza la salida y entrada de pines digitales. Mediante la manipulación de un *dip-switch* de 4 vías conectados a los puertos: 12, 11, 10 y 9 del Arduino Uno, se deberá observar en un *display* de 7 segmentos los dígitos del 0 al 9, el *display* será conectado a los pines: 8, 7, 6, 5, 4, 3 y 2 del Arduino Uno. Los elementos utilizados en la presente práctica son: *display* de 7 segmentos, *dip-switch* de 4 vías, cables de *protoboard* y Arduino Uno. En cuanto al *software*, se utiliza: Arduino, programa proporcionado por los creadores de las placas Arduino. La programación correspondiente a esta sección se la puede encontrar en hoja guía del instructor en el Anexo B 1 y la correspondiente hoja guía del estudiante de esta práctica se presenta en el Anexo C 1.

En la Figura 3.15 se observa el diagrama de flujo perteneciente a esta sección. En la primera parte se crea las variables que pertenecen a los pines, es decir se da un nombre al número del pin tanto para el dip-swtich como para el *display*. En la función *void setup,* se declaran los pines asignados para el *dip-switch* como pines de entrada (pull-up) y a los pines del *display*, como pines de salida digital. En *void loop,* se realiza el programa el cual, convertirá la entrada binaria del *dip-switch* a decimal en el *display*; es decir, si se ingresa en el *dip-switch* el número binario 0011, el *display* mostrará su correspondiente en decimal, es decir el número 3.

Asignación de pines PWM a variables
const int led1 = 11;
const int led2 = 10;
const int led3 = 9;
const int led4 = 6;
const int led5 = 5;
const int led6 = 3;

Asignación de pines Input a variables
const int bot1 = 0;
const int bot2 = 1;
const int bot3 = 2;

Asignación de pin Potenciómetro
const int pot = 3;

Asignación de variables genereales
int brillo;
int potenciometro;

VOID SETUP

e de bienvenida del LCD

ines de salidas pwm
d1,OUTPUT);
d2,OUTPUT);
d3,OUTPUT);
d4,OUTPUT);
d5,OUTPUT);
d6,OUTPUT);

Bucle n veces: ∞

VOID LOOP

LCD CLEAR

Potenciómetro = ADC

Valor ADC del potenciometro

ADC = 600

SÍ

NO

JUEGO DE LUCES 1

BRILLO =0

Bucle n veces: 254

analogWrite(led1, brillo)
analogWrite(led2, brillo-43)
analogWrite(led3, brillo-(43*2))
analogWrite(led4, brillo-(43*3))
analogWrite(led5, brillo-(43*4))
analogWrite(led6, brillo-(43*5))
brillo++/

JUEGO DE LUCES 1

BRILLO =256

analogWrite(led1, brillo);
analogWrite(led2, 0);
analogWrite(led3, 0);
analogWrite(led4, 0);
analogWrite(led5, 0);
analogWrite(led6, 0);

analogWrite(led1, brillo-150);
analogWrite(led2, brillo);
analogWrite(led3, 0);
analogWrite(led4, 0);
analogWrite(led5, 0);
analogWrite(led6, 0);

analogWrite(led1, brillo-180);
analogWrite(led2, brillo-150);
analogWrite(led3, brillo);
analogWrite(led4, 0);
analogWrite(led5, 0);
analogWrite(led6, 0);

analogWrite(led1, brillo-210);
analogWrite(led2, brillo-180);
analogWrite(led3, brillo-150);
analogWrite(led4, brillo);
analogWrite(led5, 0);
analogWrite(led6, 0);

analogWrite(led1, brillo-254);

ASIGNACIÓN DE VARIABLES A PINES DE ENTRADA
int bit0=12; //LSB
int bit1=11;
int bit2=10;
int bit3=9;  //MSB

ASIGNACIÓN DE VARIABLES A  PINES PARA SALIDA DE DISPLAY
int a=8; //A
int b=7; //B
int c=6; //C
int d=5; //D
int e=4; //E
int f=3; //F
int g=2; //G

ODO DE
TCH

PULL-UP
UP DEL PIN 12
UP DEL PIN 11
UP DEL PIN 10
UP DEL PIN 9

IDA PARA EL

PARA EL LED A
PARA EL LED B
PARA EL LED C
PARA EL LED D
PARA EL LED E
PARA EL LED F
PARA EL LED G

Bucle n veces: ∞

VOID SETUP

Lee los pines BIT1, BIT2, BIT3 y BIT4

SÍ — BIT0 = 0, BIT1 = 0, BIT2 = 0, BIT3 = 0 — NO
SÍ — BIT0 = 1, BIT1 = 0, BIT2 = 0, BIT3 = 0 — NO
SÍ — BIT0 = 0, BIT1 = 1, BIT2 = 0, BIT3 = 0 — NO
SÍ — BIT0 = 1, BIT1 = 1, BIT2 = 0, BIT3 = 0 — NO
SÍ — BIT

Muestra 0 en el display 7 seg
Muestra 1 en el display 7 seg
Muestra 2 en el display 7 seg
Muestra 3 en el display 7 seg
Muestra 4 en el

NO — BIT0 = 1, BIT1 = 0, BIT2 = 0, BIT3 = 1 — SÍ
NO — BIT0 = 0, BIT1 = 0, BIT2 = 0, BIT3 = 1 — SÍ
NO — BIT0 = 1, BIT1 = 1, BIT2 = 1, BIT3 = 0 — SÍ
NO — BIT0 = 0, BIT1 = 1, BIT2 = 1, BIT3 = 0 — SÍ
NO — BIT

Muestra 9 en el display 7 seg
Muestra 8 en el display 7 seg
Muestra 7 en el display 7 seg
Muestra 6 en el display 7 seg

## PRÁCTICA 2: MANEJO DEL MÓDULO *SHIELD ETHERNET* UTILIZANDO MÓDULOS ARDUINO.

Objetivo: Controlar elementos pasivos como LEDs y *buzzer* a través de una interfaz gráfica, proporcionada por el módulo *Shield Ethernet* previamente programada en el *IDE* de Arduino.

El módulo *Shield Ethernet*, ubicado sobre el Arduino Uno, almacena la configuración de la página web que, permitirá el encendido y apagado de 2 LEDs y un *buzzer* conectados a los pines digitales: 3, 4 y 5 del Arduino Uno. También permitirá observar la variación de un potenciómetro conectado al pin A0 del Arduino. Para esta práctica los elementos a utilizarse del módulo didáctico son: LEDs, potenciómetro, cables de *protoboard*, módulo *shield Ethernet* y *buzzer*; también se necesitará un cable de red RJ45 que no incluye el módulo didáctico.

Para la configuración de la página web, se utilizan librerías que vienen incluidas en el programa Sketch de Arduino [18] y un conocimiento básico de códigos HTML [18]. En cuanto al *software*, se utiliza: Arduino, programa proporcionado por los creadores de las placas Arduino. La programación correspondiente a la práctica se la puede encontrar en la hoja guía del instructor en el Anexo B2 y la correspondiente hoja guía del estudiante de esta práctica se presenta en el Anexo C 2.

A continuación se presenta, en la Figura 3.16, el diagrama de flujo. Se observa que primero se incluyen las librerías que utilizará el módulo *Ethernet*, también se configura las credenciales de la red como: la dirección IP, el Gateway y la Máscara de red. En el *void setup,* se configuran los pines que utilizarán los LEDs y *Buzzer* como pines de salida, también se inicializa la comunicación serial entre el módulo *Ethernet* y el Arduino. En la función *void loop,* se inicializa el servidor Web, el módulo espera que se conecte algún cliente para comenzar con el programa. En esta parte del programa se configura la página Web que se visualizará en el

monitor de la PC, se crean hipervínculos que permitirán el encendido y apagado de los LEDs y del *Buzzer*, también se muestra la variación del potenciómetro.

## PRÁCTICA 3: MANEJO DEL MÓDULO *WIFI*

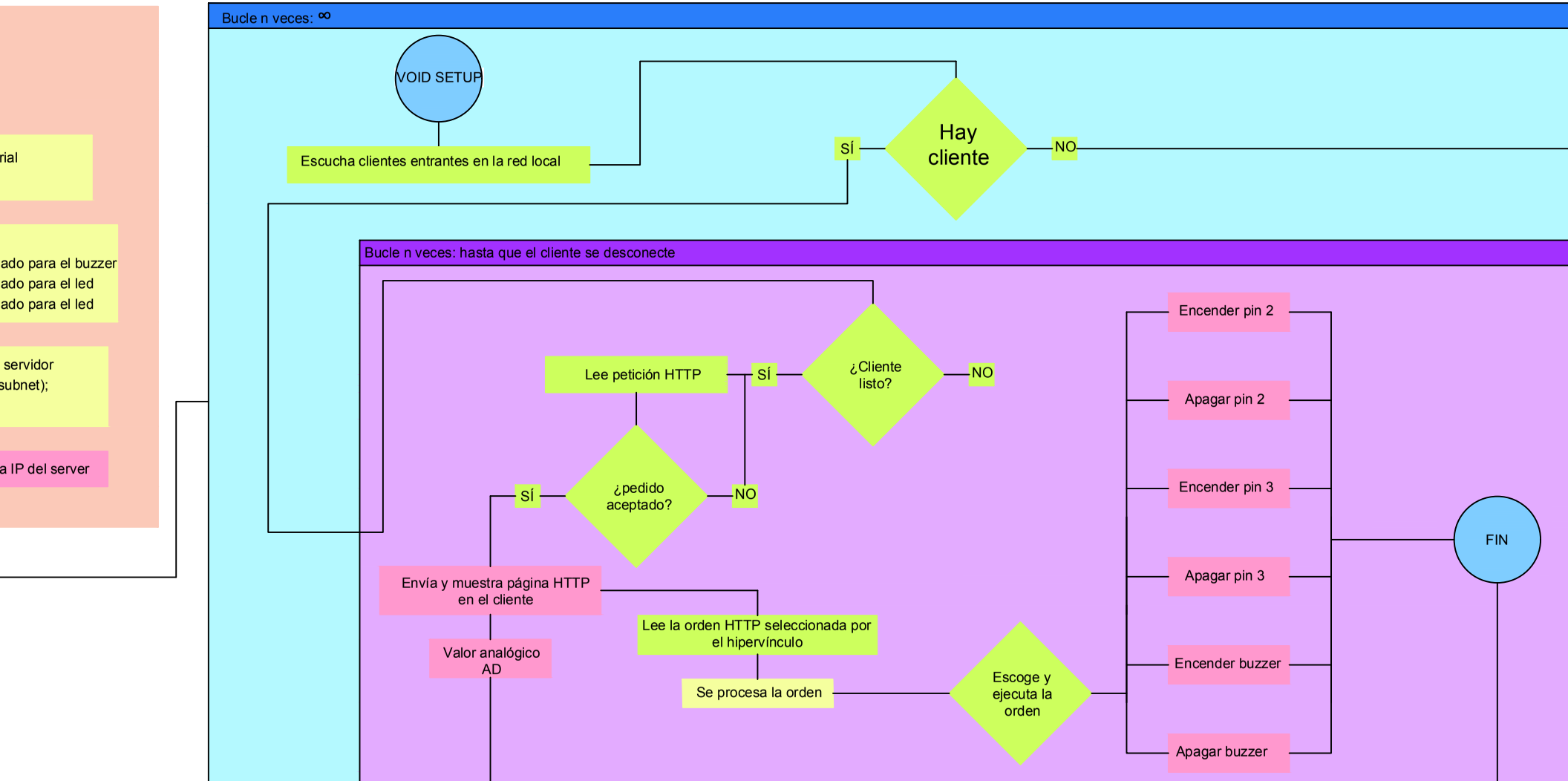
Objetivo: Controlar 2 LEDs a través de una interfaz gráfica, proporcionada por el módulo *WiFi*.

Los 2 LEDs, de la placa didáctica, serán conectados a los pines: GPIO_0 y GPIO_2, del módulo *WiFi*; y el *software* utilizado será *Arduino,* programa proporcionado por los creadores de las placas Arduino. Los elementos adicionales a usarse son: un teléfono celular, para compartir la red inalámbrica o un *Access Point* y un dispositivo con *Wireless* para conectarse a la red e ingresar a la página creada. Para la programación del módulo *WiFi* son necesarias ciertas líneas de código, que se las puede encontrar en foros de discusión sobre el ESP8266 [19]. La programación correspondiente a la práctica se la puede encontrar en la hoja guía del instructor en el Anexo B3 y la correspondiente hoja guía del estudiante de esta práctica se presenta en el Anexo C3

A continuación se presenta, en la Figura 3.17, el diagrama de flujo. Se observa que se inicializa el programa con la inclusión de las librerías ESP8266WIFIClient, *WIFIWebServer*, *WIFIimDNS*. Se añaden las credenciales de la red inalámbrica como: el nombre de la red y su contraseña y se asigna una variable a los pines a usarse. En la función *void setup¸* se configura la página Web y sus hipervínculos que permitirán el encendido y apagado de los LEDs. Se configuran los pines a usarse, como pines de salida y se realiza la conexión con el AP o dispositivo que proporciona la red inalámbrica.

Librerías a usar
#include <SPI.h>
#include <Ethernet.h>

Asignación de parámetros para el módulo ethernet
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //Dirección Física MAC
byte ip[] = { 192, 168, 1, 50 }; // IP Local que usted debe configurar
byte gateway[] = { 192, 168, 1, 1 }; // Puerta de enlace
byte subnet[] = { 255, 255, 255, 192 }; //Máscara de Sub Red
EthernetServer server(80); //El puerto 80 es el predeterminado para HTTP
String readString;



Asignación de parámetros para la conexión inalámbrica
const char* ssid = "Ejemplo"; // Nombre de la red Inalámbrica
const char* password = "*******"; // Clave de la red Inalámbrica
ESP8266WebServer server(80); // Puerto de HTTP
String webPage = "";

Asignación de pines variables
int gpio0_pin = 0; // el Pin0 del módulo WiFi tiene de nombre gpio0_pin
int gpio2_pin = 2; // el Pin2 del módulo WiFi tiene de nombre gpio2_pin;

VOID SETUP

Se carga la página web a usar a una variable

ce pines de salida y se inicializa
e(gpio0_pin, OUTPUT); // gpio0_pin como salida
/rite(gpio0_pin, LOW); //Se escribe en gpin0 0v
e(gpio2_pin, OUTPUT); // gpio2_pin como salida
/rite(gpio2_pin, LOW); //Se escribe en gpin2 0v

Inicializa la conexión serial
Serial.begin(115200);

Inicializa la conexión inalambrica
WiFi.begin(ssid, password);

¿Conexión exitosa?

SÍ

NO

uestra en la pantalla serial la IP del servidor

Bucle n veces: ∞

VOID LOOP

Escucha clientes entrantes en la red local

¿Hay clientes?

SÍ

NO

Bucle n veces: hasta que el cliente se desconecte

Lee petición HTTP

¿Cliente listo?

SÍ

NO

¿pedido aceptado?

SÍ

NO

Envía y muestra página HTTP en el cliente

Ejecuta la orden HTTP leída por el hipervínculo

Encender pin 2

Apagar pin 2

Escoge y

## PRÁCTICA 4: MANEJO DE MÓDULO *SHIELD* GSM SIM 800C

Objetivo: Enviar mensajes y realizar llamadas de voz, utilizando pulsadores y un *switch* controlados por el módulo *Shield* GSM y Arduino Uno.

Esta práctica consiste en el manejo del módulo *Shield* GSM que se encuentra sobrepuesto en el Arduino Uno. Mediante el uso de 3 pulsadores conectados a los pines digitales 3, 4 y 5 del Arduino y un *switch* al pin digital 6; realizar llamadas y enviar 3 distintos mensajes de texto. El *software* utilizado será: *Arduino,* programa proporcionado por los creadores de las placas Arduino; los elementos a usarse del módulo didáctico son: módulo GSM, Arduino, 3 pulsadores, un *switch* y cables de *protoboard*; adicionalmente se necesita un *MicroChip* de cualquier operadora de telefonía celular.

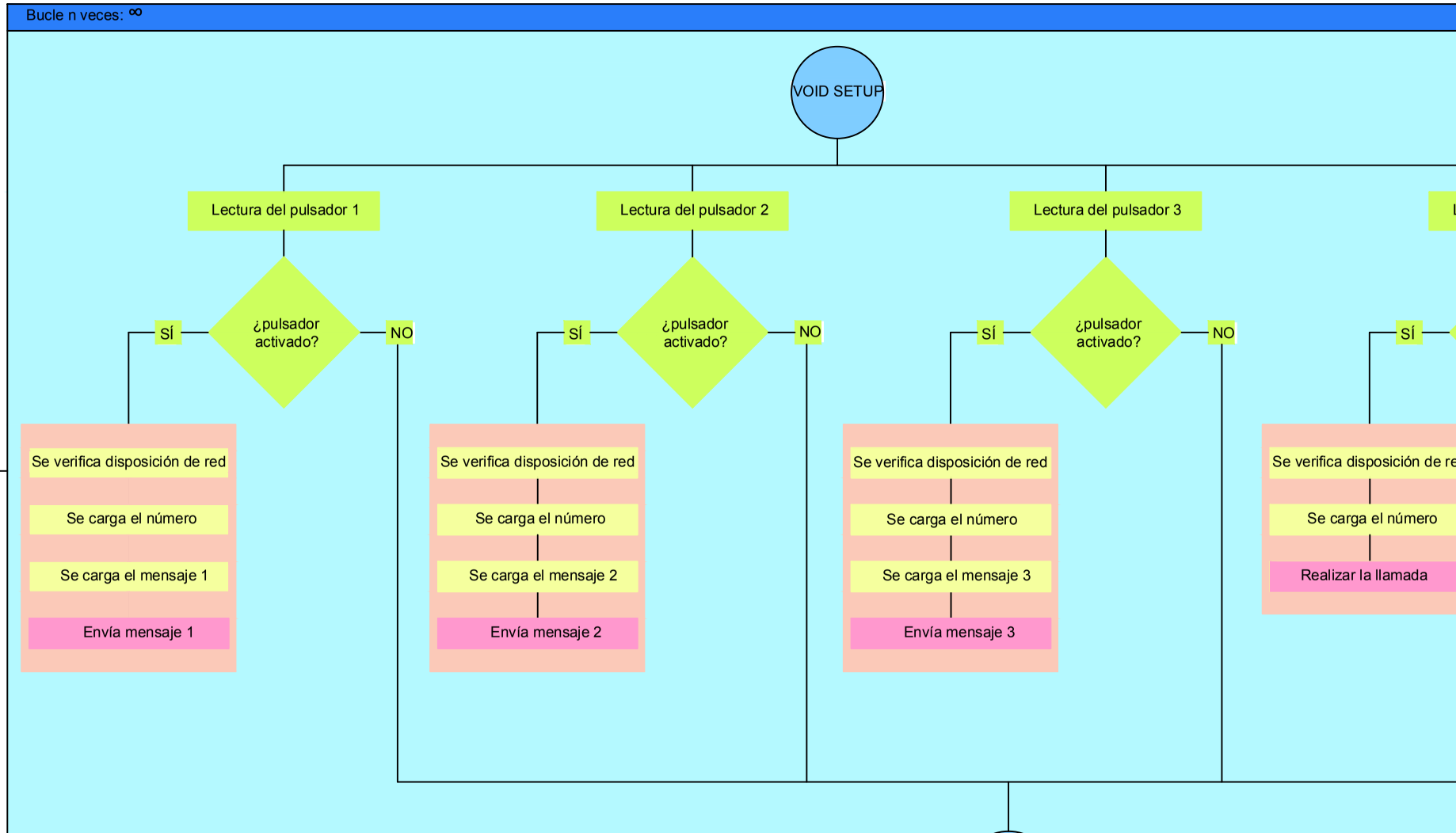
Para la programación del módulo GSM se utilizarán las librerías que el *software* ofrece, además de algunos códigos AT, los cuales son necesarios para el envío de mensajes y realización de llamadas de voz [20]. El desarrollo de las prácticas y su programación se la puede encontrar en la hoja guía del instructor en el Anexo B4 y la correspondiente hoja guía del estudiante de esta práctica se presenta en el Anexo C4.

A continuación se presenta, en la Figura 3.18, el diagrama de flujo. Se observa que se incluyen las librerías Serial y *String*, se configura la comunicación serial y se declaran las variables que utilizarán los pines de los pulsadores y *switch*. En la función *void setup,* se inicializa la comunicación serial y se configuran los pines de los pulsadores y *switch* como pines de entrada. Para enviar un mensaje de texto, se crea una función llamada: *void EnviarMensajeTexto*, la cual, mediante códigos AT, envía un mensaje de texto a un número celular cualquiera al oprimir un pulsador. Se crea otra función para realizar una llamada de voz: *void LlamadaDeVoz* esta función realiza una llamada de voz a un número celular al accionar el *switch*.

Librerías a usar
#include <SoftwareSerial.h>
#include <String.h>

Se crea una comunicación serial emulada en los pines 7 y 8
SoftwareSerial mySerial(7,8);

Se crea constantes para asignarlos a los pines de pulsadores
const int pul_sms_1 = 3 ,pul_sms_2 = 4 ,pul_sms_3 = 5,
dsw_call = 6;
int lectura1,lectura2,lectura3,lectura4;
unsigned int aux=0;



## 3.4 PRUEBAS DE FUNCIONAMIENTO

### PRÁCTICA 1: MANEJO DE PUERTOS DE ENTRADA Y SALIDA UTILIZANDO EL MÓDULO ARDUINO UNO.

Sección 1: Juego de luces

Para las pruebas de funcionamiento hay que tomar en cuenta que previamente se realizó la conexión, como se detalla en el Anexo B1 y se grabó el programa correspondiente.

En la Figura 3.19 se observa el primer juego de luces; los LEDs se encuentran encendidos y en la Figura 3.20 se aprecia el LCD mostrando el valor del potenciómetro y el mensaje: *"JUEGO DE LUCES 1".*

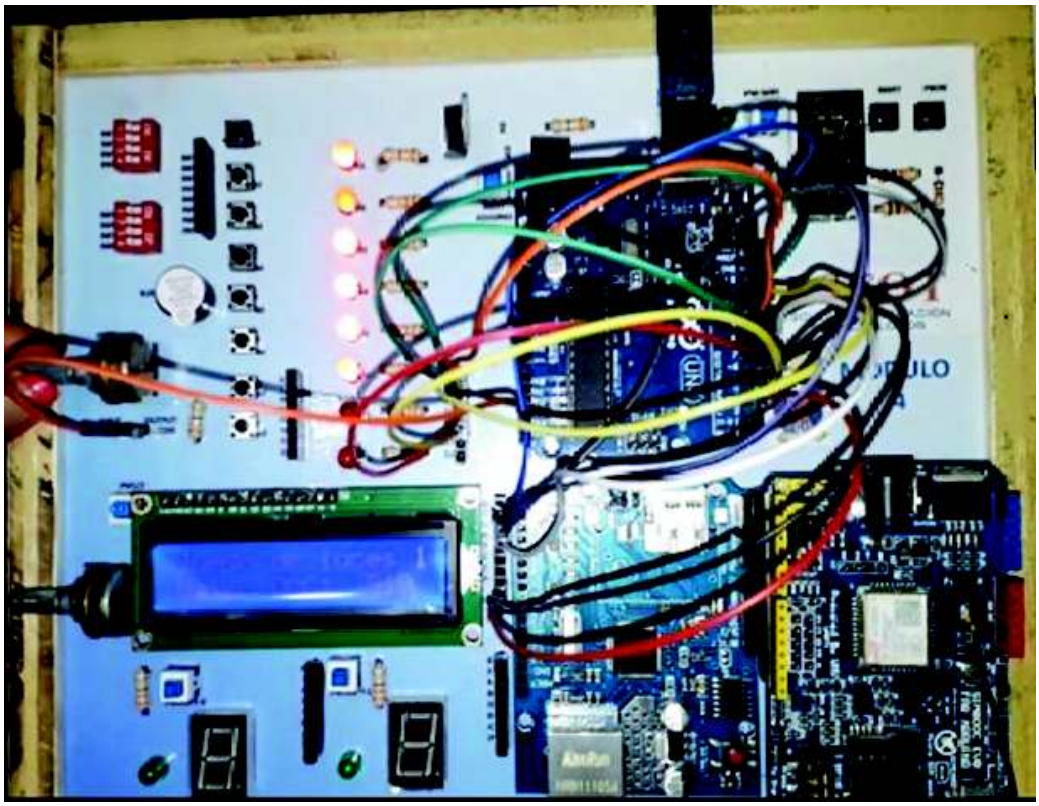


*Figura 3. 19 Prueba de funcionamiento práctica 1 sección 1, juego de luces 1*


*Figura 3. 20 Mensaje del primer juego de luces y valor del potenciómetro*

En las Figuras 3.21 y 3.22 se puede visualizar el juego de luces 2 y su mensaje en el LCD, respectivamente.

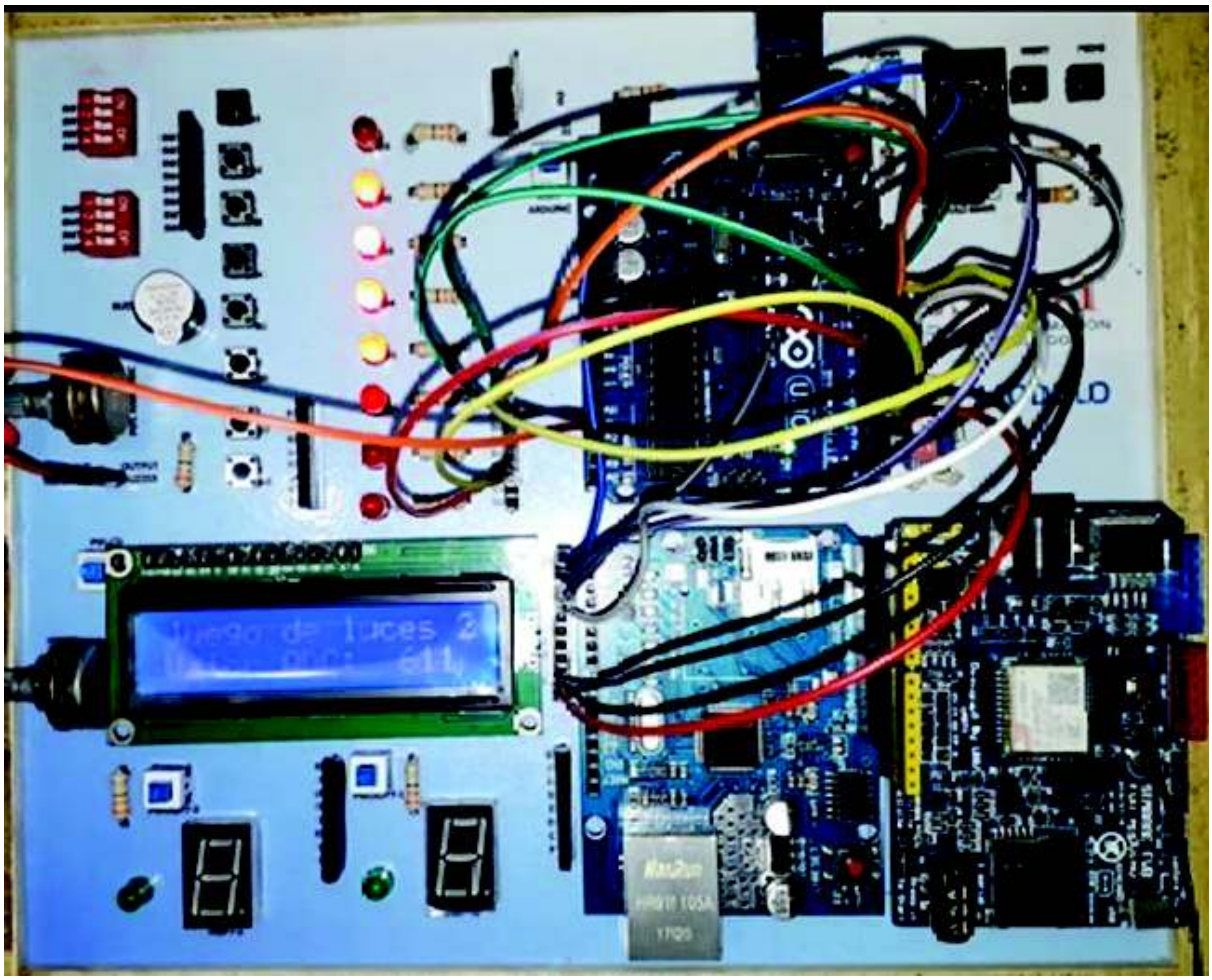


*Figura 3. 21 Prueba de funcionamiento práctica 1 sección 1, juego de luces 2*


*Figura 3. 22 Mensaje del segundo juego de luces y valor del potenciómetro*

En esta práctica se utilizan pines PWM los cuales, ayudan a tener variaciones de intensidad lumínica, siendo 0 apagado y 255 encendido; entre 0 y 255 se podrá apreciar la intensidad de la luz. Es importante recordar al usuario del módulo, que todos los pines que dispongan el símbolo: ~, son pines PWM, caso contrario serán digitales y no podrán ser usados como pines PWM; sin embargo, los pines PWM sí podrán ser usados como digitales [21].

Para la sección 2, se utilizó, un *dip-switch* de 4 vías, un *display* de 7 segmentos, cables de *protoboard* y Arduino Uno. Como se puede observar en la Figura 3.23 las conexiones correspondientes a esta práctica se encuentran realizadas, como se detalla en el Anexo B1. En la Figura 3.24, se puede observar el número en binario 0011 y en la Figura 3.25 se observa su equivalente en decimal (3).

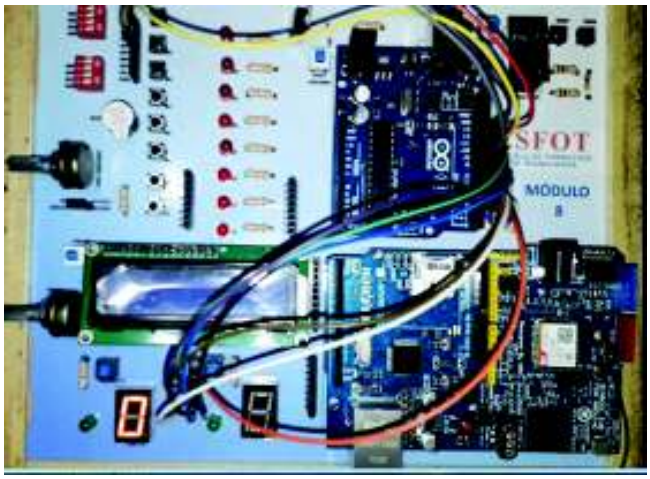



*Figura 3. 23 Ensamblaje del circuito de la sección 2 de la práctica 1*

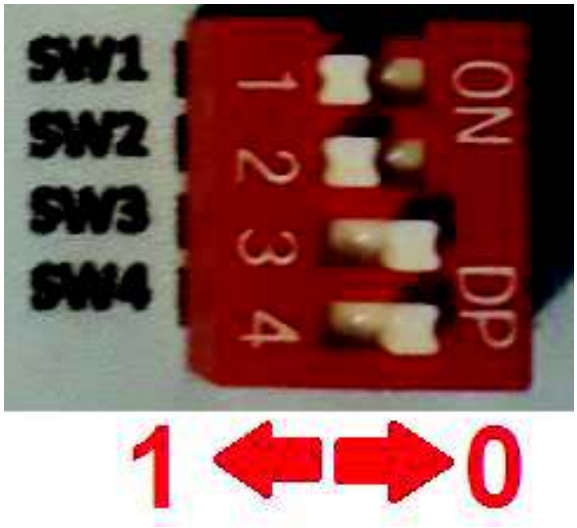



*Figura 3. 24 Número en binario 0011*

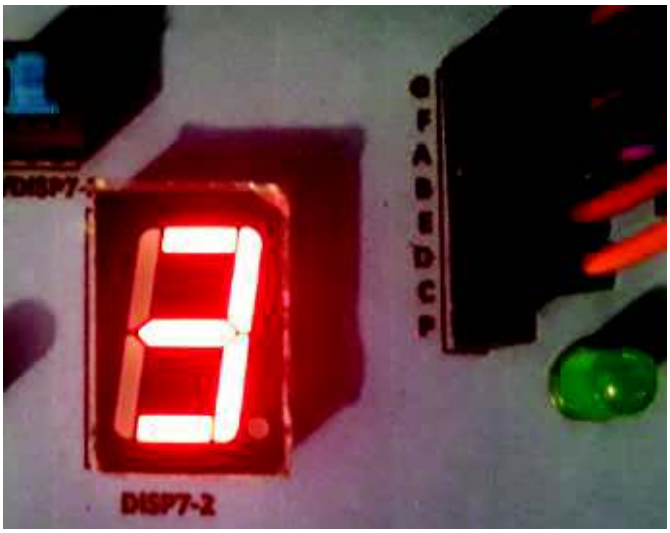



*Figura 3. 25 Número equivalente en decimal*

Para la realización de cada sección se necesitan aproximadamente 45 minutos, cada vez que se realice un circuito nuevo, se deberá desconectar todo y apagar todos los subcircuitos. La presente práctica 1, se encuentra desarrollada para que el usuario se familiarice con la nueva programación de Arduino y con su Arquitectura. Es posible añadir más elementos en el estudio de entradas/salidas digitales y analógicas ya que el módulo didáctico ofrece: pulsadores, *buzzer*, potenciómetro, LCD, *display* 7 segmentos, LEDs y *dip-switch*.

**PRÁCTICA 2: MANEJO DEL MÓDULO *SHIELD ETHERNET* UTILIZANDO MÓDULOS ARDUINO.**

Como se puede observar en la Figura 3.26, el circuito se encuentra ensamblado y el módulo *Shield Ethernet* fue desmontado de su lugar e instalado sobre el Arduino Uno, como se muestra en el Anexo B2.

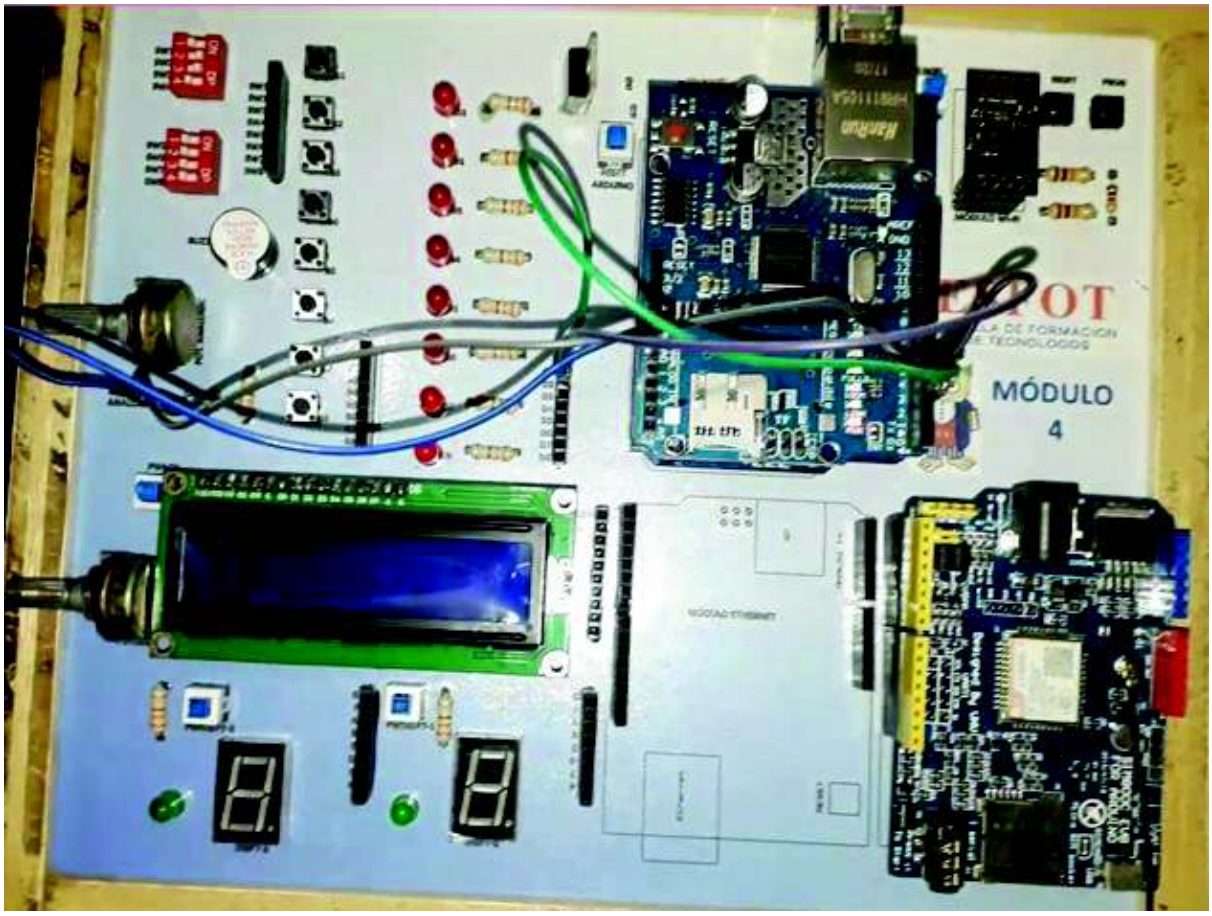



*Figura 3. 26 Circuito ensamblado de la práctica 2*

Una vez realizada la conexión entre la PC y el módulo *Ethernet*, se puede ingresar a la página web creada, como se muestra en la Figura 3.27. Se pueden observar todos los hipervínculos que fueron creados en el programa, los cuales ayudarán al encendido y apagado de los LEDs y *buzzer*; y también se observa en la parte inferior la lectura analógica de la entrada de un potenciómetro.


*Figura 3. 27 Página Web creada*

Si se selecciona el hipervínculo “Encender Pin2” el LED asociado a ese pin se encenderá, como se muestra en la Figura 3.28.

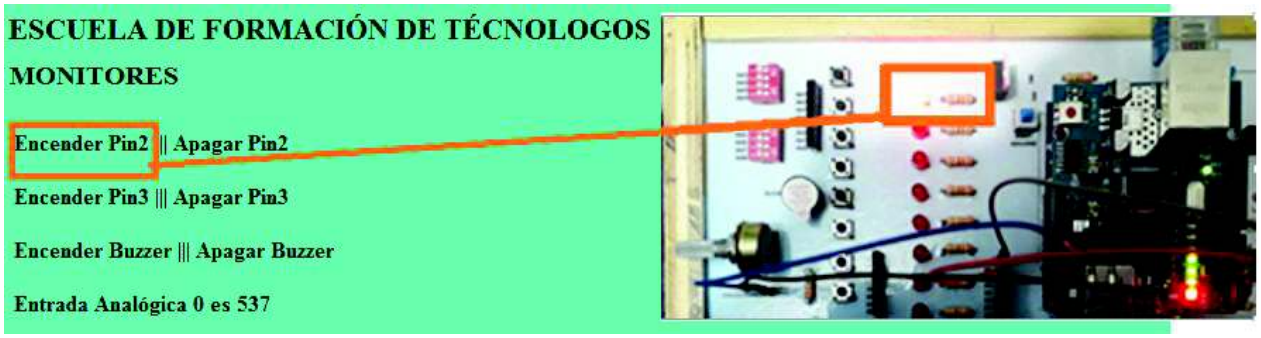



*Figura 3. 28 Selección del hipervínculo “Encendido Pin2”*

En la Figura 3.29 se observa el apagado del segundo LED al seleccionar el hipervínculo “Apagar Pin3”.

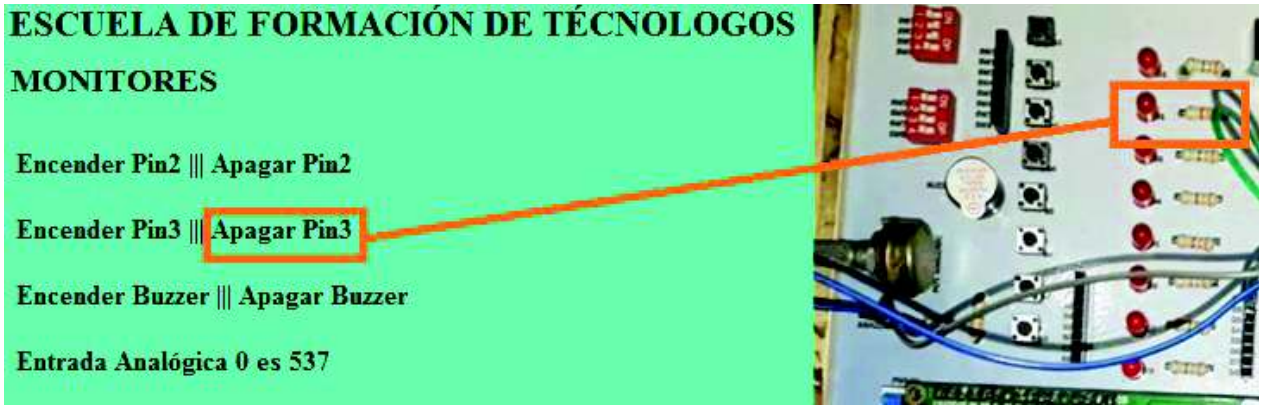



*Figura 3. 29 Selección del hipervínculo "Apagar Pin3"*

Para observar la variación de la entrada digital, dirigirse al final de la página web, como se muestra en la Figura 3.30. El valor de la línea de texto varía, según el movimiento del potenciómetro ubicado en el módulo didáctico.

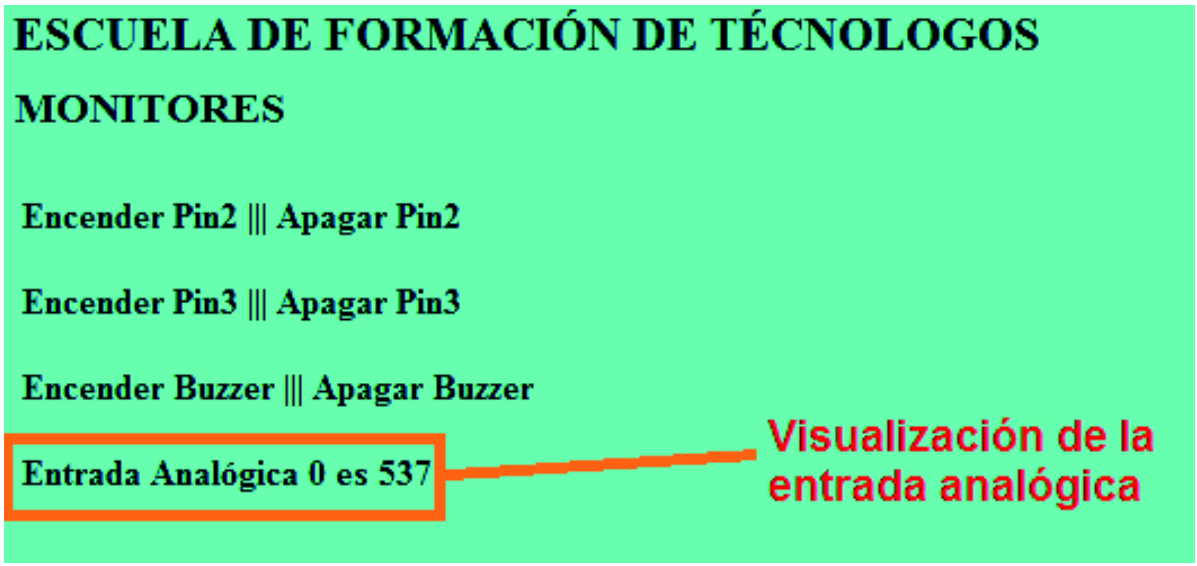



*Figura 3. 30 Visualización del valor de la entrada analógica*

El tiempo estimado para el desarrollo de esta práctica es de: 45 minutos a 1 hora. El usuario podrá añadir elementos que el módulo didáctico ofrece tales como: *display* de 7 segmentos, LCD, pulsadores y *dip-switch*; de esta manera, al añadir más elementos, se obtendrá prácticas más complejas que cubran varios temas de estudio.

## PRÁCTICA 3: MANEJO DEL MÓDULO *WIFI*

En la Figura 3.31, se observa la conexión realizada de la presente práctica como se detalla en el Anexo B3. Es importante recordar que el Arduino Uno simplemente funciona como puente de comunicación entre el módulo *WiFi* y el programa de Arduino para su programación, para que exista esta comunicación es necesario que el Arduino se encuentre en modo *"Reset".*


*Figura 3. 31 Circuito ensamblado de la práctica 3*

En la Figura 3.32 se puede observar la página web creada, la cual contiene los hipervínculos que encenderán y apagarán dos LEDs.


*Figura 3. 32 Página Web proporcionada por el módulo WiFi*

En la Figura 3.33 se observa una ventana serial que muestra la información de la red inalámbrica y la dirección IP de la página Web.

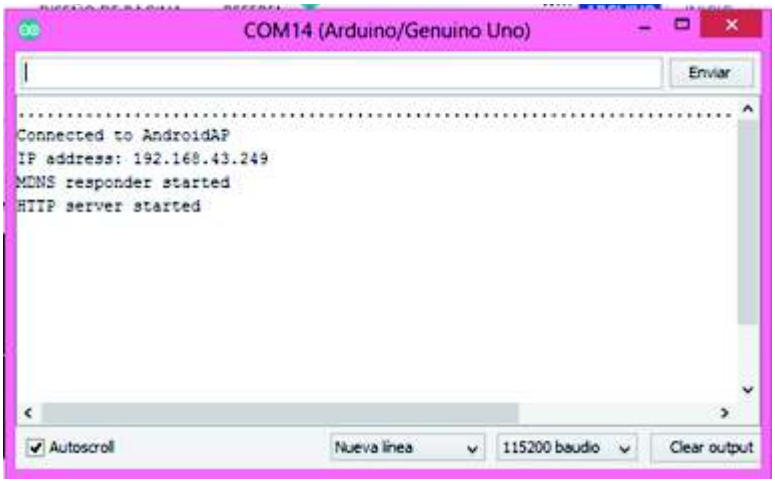



*Figura 3. 33 Pantalla serial del módulo WiFi*

Para el encendido de los LEDs, se selecciona los hipervínculos “Socket #1: ON” y Socket #2: ON”, como se muestra en la Figura 3.34.

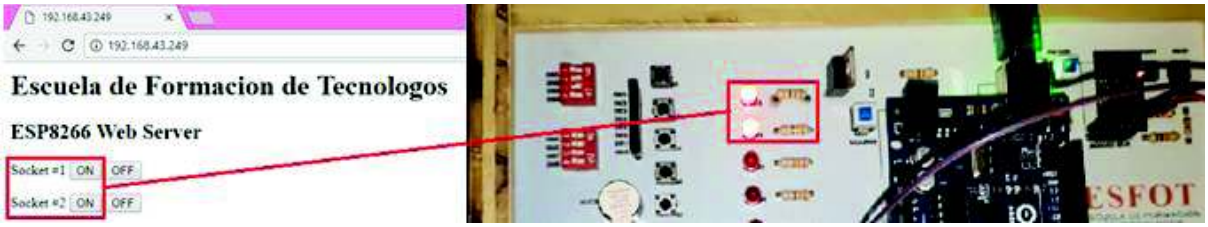



*Figura 3. 34 Encendido de LEDs*

En la Figura 3.35, se observa el apagado de los LEDs al seleccionar los hipervínculos: *"Socket #1: OFF"* y *Socket #2: OFF"*.

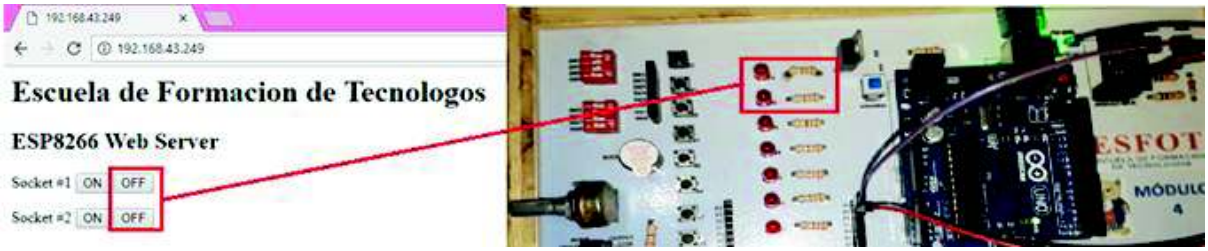



*Figura 3. 35 Apagado de LEDs*

Debido al limitado número de pines de entrada del módulo *WiFi*, es imposible añadir más elementos para controlar; sin embargo, se podría utilizar un LED y un pulsador o una vía del *dip-switch* para la realización de varias prácticas.

## PRÁCTICA 4: MANEJO DEL MÓDULO *SHIELD* GSM SIM800

En la Figura 3.36 se observa que el módulo GSM fue desprendido de su lugar y colocado encima del módulo Arduino, como se detalla en el Anexo B4. En la Figura 3.37 se observa el circuito ensamblado de la presente práctica, como se detalla en el Anexo B4.

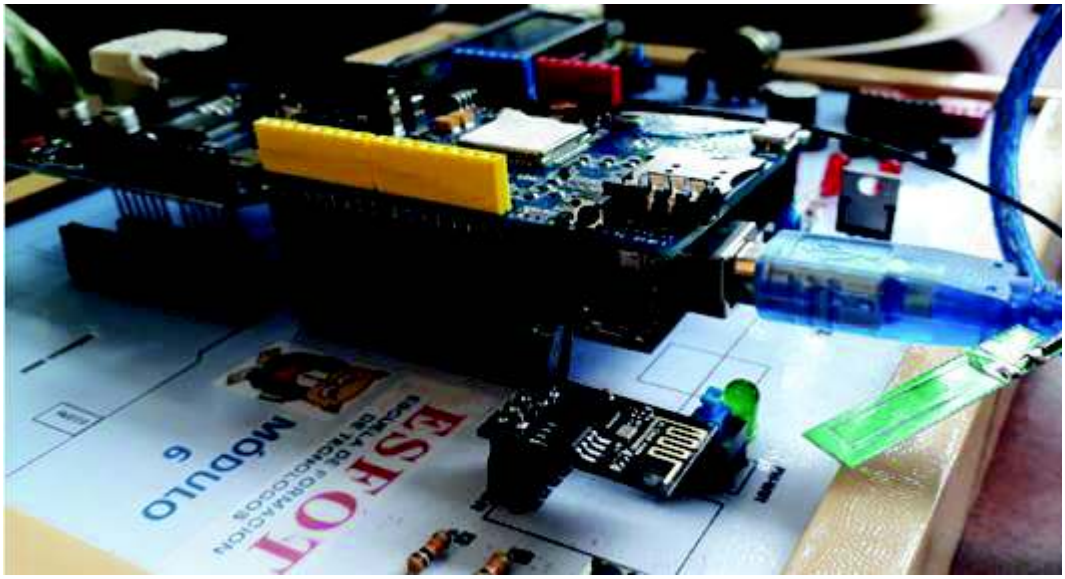



*Figura 3. 36 Instalación de módulo GSM*

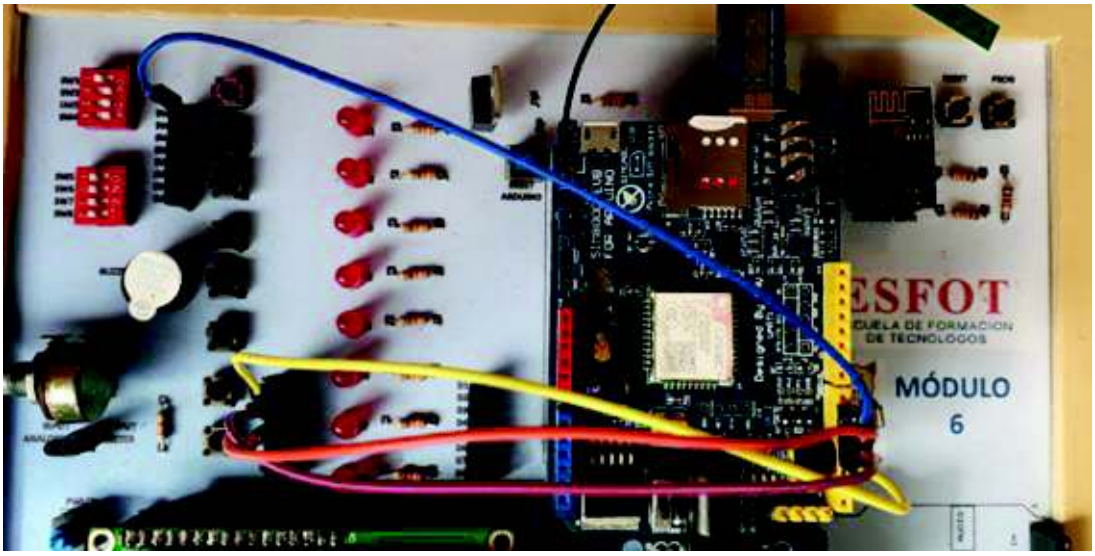



*Figura 3. 37 Circuito ensamblado de la práctica 5*

En la Figura 3.38 se observa cómo al activar el *dip-switch* se realiza la llamada.

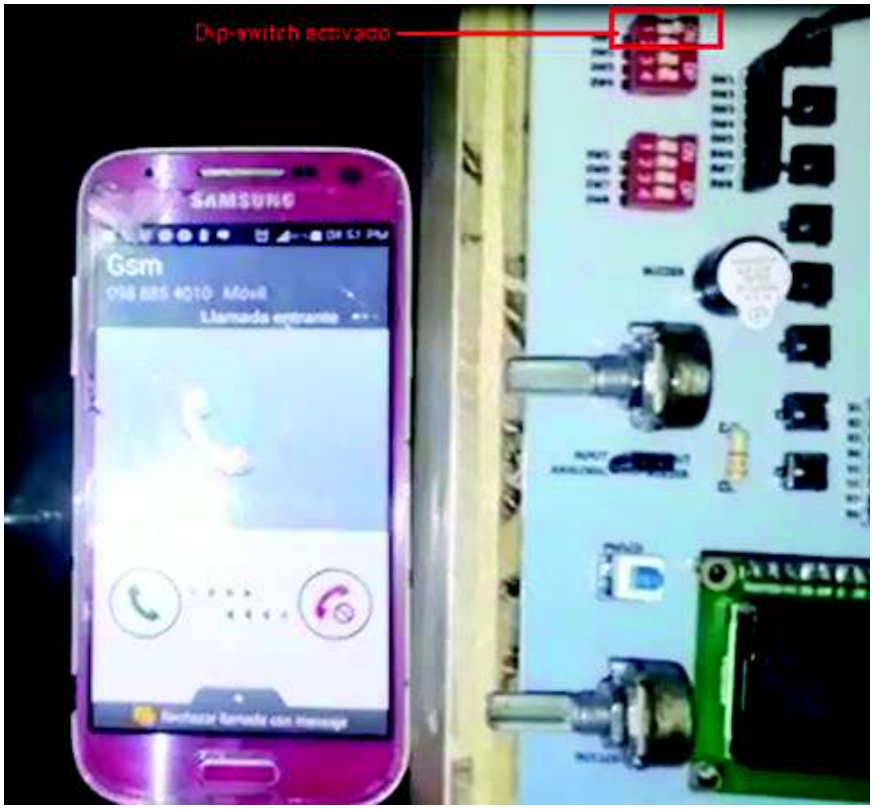



*Figura 3. 38 Llamada entrante del módulo GSM*

En la Figura 3.39 se observa los mensajes enviados por tres pulsadores; el Pulsador 1 envía un SMS 1, el Pulsador 2 envía el SMS 2, el pulsador 3 envía el SMS 3.


*Figura 3. 39 Mensajes de Texto enviados por el módulo GSM*

## 3.5 COSTO DEL PROYECTO

En la tabla 3.1 se puede observar, el costo unitario por módulo didáctico. En la Tabla 3.2 se encuentra el costo total de los 9 módulos, cabe recalcar que no se considera costos por el diseño y la mano de obra

***Tabla 3.1 Precio por módulo***

| DETALLE DEL MATERIAL | CANTIDAD | PRECIO | TOTAL |
|---|---|---|---|
| Módulo Arduino Uno | 1 | $ 13,70 | $ 13,70 |
| Módulo GPRS-GSM | 1 | $ 39,50 | $ 39,50 |
| Módulo WiFi | 1 | $ 5,00 | $ 5,00 |
| Módulo Ethernet | 1 | $ 17,25 | $ 17,25 |
| Resistencia 330Ω | 13 | $ 0,02 | $ 0,26 |
| Resistencia 1kΩ | 1 | $ 0,02 | $ 0,02 |
| Resistencia 2kΩ | 1 | $ 0,02 | $ 0,02 |
| Resistencia 10kΩ | 2 | $ 0,02 | $ 0,04 |
| Regulador LM1117 | 1 | $ 0,94 | $ 0,94 |
| Pulsadores 2P 5mm | 10 | $ 0,15 | $ 1,50 |
| Display LCD 16x2 | 1 | $ 5,08 | $ 5,08 |
| Switch | 5 | $ 0,30 | $ 1,50 |
| Potenciómetro 10KΩ | 1 | $ 0,45 | $ 0,45 |
| Potenciómetro 1MΩ | 1 | $ 0,45 | $ 0,45 |
| Diodo LED verde | 3 | $ 0,08 | $ 0,24 |
| Diodo LED rojo | 8 | $ 0,08 | $ 0,64 |
| Display de 7 Seg. Ánodo común | 2 | $ 0,45 | $ 0,90 |
| Buzzer | 1 | $ 0,58 | $ 0,58 |
| Dip-switch 4 vías | 2 | $ 0,80 | $ 1,60 |
| Espadines Hembra patas largas | 1 | $ 2,20 | $ 2,20 |
| Espadines Hembra | 5 | $ 2,20 | $ 11,00 |
| Espadines Macho patas largas | 1 | $ 2,20 | $ 2,20 |
| Baquelita 20x30 | 1 | $ 5,60 | $ 5,60 |
| Papel termotransferible | 1 | $ 2,30 | $ 2,30 |
| Marco de madera pintados | 1 | $ 18,60 | $ 18,60 |
| Seguros para el marco | 1 | $ 2,30 | $ 2,30 |
| Papel adhesivo | 1 | $ 1,50 | $ 1,50 |
| Varios (estaño, impresiones, pomada, ácido, brocas, etc.) | 1 | $ 6,25 | $ 6,25 |
| Cable Extensor de USB | 1 | $ 2,00 | $ 2,00 |
| Cables para protoboard | 1 | $ 4,50 | $ 4,50 |
| total | | | $ 148,12 |

***Tabla 3.2 Costo total de los módulos didácticos***

| DETALLE | CANTIDAD | PRECIO UNITARIO | TOTAL |
|---|---|---|---|
| Módulos didácticos basados en Arduino Uno, módulo *Shield Ethernet*, módulo *Shield* GSM y módulo *WiFi* | 9 | $ 148,12 | $ 1.333,08 |

# 4. CONCLUSIONES Y RECOMENDACIONES

## 4.1. CONCLUSIONES

- Para el diseño de los módulos didácticos, se utilizaron programas como: ISIS y *Ares 7 Profesional* que ayudan a la organización tanto lógica como física de los elementos tales como: diodos LEDs, *display* de 7 segmentos, pulsadores, *dip-switch*, potenciómetros, LCD, Arduino Uno, módulo *shield Ethernet*, módulo *WiFi* y módulo GSM; de esta manera se crearon los subcircuitos: del módulo *WiFi*, *display* LCD, *displays* 7 segmentos, módulo *Ethernet*, módulo Arduino uno, módulo GSM, pulsadores, *dip-switch* y diodos LEDs que facilitan el uso del módulo.
- Se logró integrar todos los elementos en un solo sistema gracias, a la creación de los *PCB* (*Printed Circuit Board*) para cada módulo incluido el *Arduino Uno ya que* el programa *Ares* no posee todos los *PCB*, los cuales son archivos que contienen las dimensiones reales de los elementos electrónicos a ser usados y que vienen prediseñados en este.
- Con el IDE de Arduino, se realiza la programación del Arduino y módulo *WiFi*, siguiendo las funciones del *void loop* y *void setup*; según el módulo que se utilizará (*Ethernet*, *WiFi* o GSM) se podrán utilizar las librerías que el IDE ofrece como por ejemplo: <*LiquidCrystal.h*> y <*Ethernet.h*>, ayudan al usuario en la programación. También se puede encontrar información en

los foros de Internet, sobre la programación de los módulos que se desee utilizar.

- Con el desarrollo e implementación de las prácticas *Ethernet* y *WiFi* se da a conocer los principios de la domótica como la automatización de viviendas y edificios para brindar seguridad, confort y ahorro de energía, de esta manera se incentiva al usuario a realizar proyectos más complejos y avanzados.
- El módulo GSM a diferencia de los otros módulos utiliza el cable USB-A/B azul, proporcionado en el módulo didáctico, y para comenzar con la comprobación de la práctica debe ser inicializado individualmente para que pueda funcionar y encontrar una red móvil para lo cual, se debe presionar un botón integrado al módulo por 4 segundos y esperar aproximadamente 30 segundos a 1 minuto para poder empezar a utilizarlo.
- El módulo *WiFi* tiene la particularidad de funcionar autónomamente es decir, no es necesario que un *Arduino* lo controle, por lo que es el único módulo que se le puede programar directamente con el IDE y utilizar sus propios puertos de salida llamados *GPIO0* y *GPIO2*. La función del *Arduino* para este módulo es únicamente de puente entre el ordenador y el módulo que permite grabar el programa.
- Para el desarrollo de las prácticas de *Ethernet* y *WiFi*, fue necesario investigar sobre el leguaje HTML para la creación de las páginas Web, de igual manera, con la práctica de GSM fue necesario indagar solo códigos AT que permitan realizar llamadas y enviar mensajes.
- En el mercado no existe en venta una placa didáctica similar a la que se construyó en el presente proyecto, debido al alto costo de los elementos y la dificultad de ensamblarlos en un solo sistema integrado. Este proyecto trata de solucionar este inconveniente utilizando módulos de fácil acceso y de bajo coste para beneficio del estudiante.

## 4.2. RECOMENDACIONES

- Mantener desconectado el módulo Arduino de la fuente de poder mientras se arman los circuitos, esto con el fin de evitar cortocircuitos por mala conexión.
- Para futuros proyectos inspirados en este prototipo, considerar utilizar el Arduino Mega, ya que posee más pines a disposición del usuario.
- En el caso de querer realizar un PCB (*Printed Circuit Board*) o placa de circuito impreso de algún módulo de este proyecto, se debe tener en cuenta que el software *Ares* permite una separación entre pines de 2.5mm, un decimal de precisión, pero la separación real de pines es de 2.56mm, es decir 2 decimales de precisión. Esto causa que el elemento no encaje con el diseño y el problema es mayor si se tiene pines consecutivos, para lo cual recomendamos que cada 4 pines se realice un ajuste en las medidas para minimizar el problema.
- Se debe conocer principios de programación para obtener grandes resultados, por lo que se recomienda tener bases sólidas de programación para utilizar estos módulos.
- Es necesario investigar lenguaje HTML o programación de páginas Web ya que es una herramienta necesaria para el uso de los módulos *Ethernet* y *WiFi*, como también lenguaje AT para el uso del módulo GSM.
- Para los módulos *Ethernet* y *WiFi* se recomienda aplicaciones de automatización como: control de luces y puertas, detección de intrusos con sensores de movimiento, sistema de alarmas de seguridad, entre otras que sí, se combina con el módulo GSM se puede gestionar remotamente vía SMS.
- Es importante seguir las indicaciones de mantenimiento preventivo y correctivo que, se indican en el manual de mantenimiento anexado a este documento, de esta forma se garantiza el funcionamiento de los módulos didácticos.

# BIBLIOGRAFÍA


[1] ESFOT, «MALLA CURRICULAR ET,» 17 12 2015. [En línea]. Available: http://esfot.epn.edu.ec/index.php/component/jdownloads/send/3-oferta-academica/4-malla-curricular-et. [Último acceso: 07 11 2017].

[2] ESFOT, *PROGRAMA DE ESTUDIOS POR ASIGNATURA - MICROPROCESADORES,* Quito, 2010.

[3] ESFOT, *PROGRAMA DE ESTUDIOS POR ASIGNATURA-CONTROL CON MICROPROCESADORES,* Quto, 2010.

[4] M. McRoberts, «Arduino Básico,» 09 2011. [En línea]. Available: http://alfasol.centroruthcardoso.org.br/wp-content/uploads/sites/2/2014/10/capitulo9788575222744.pdf. [Último acceso: 10 11 2017].

[5] Ó. T. Artero, «Arduino: curso práctico de formación,» 2013. [En línea]. Available: https://books.google.es/books?hl=es&lr=&id=6cZhDmf7suQC&oi=fnd&pg=PR15&dq=que+es+arduino+uno&ots=A_9yhUGwDJ&sig=9z1ZBdU7SXj2gO2xxhc1V0R0Qel#v=onepage&q&f=false. [Último acceso: 11 11 2017].

[6] H. Torres, «Arduino vs Microcontrolador, reseña y opinión,» 17 03 2015. [En línea]. Available: https://hetpro-store.com/TUTORIALES/arduino-vs-microcontrolador/. [Último acceso: 22 11 2017].

[7] L. J. R.-A. Luis Rincón Córcoles, «ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES Programación en ensamblador: conceptos básicos,» [En línea]. Available: https://previa.uclm.es/profesorado/licesio/Docencia/ETC/17_CBas-ProgEnsamblador_itis.pdf. [Último acceso: 15 03 2018].

[8] Arduino, «¿Qué es Arduino?,» [En línea]. Available: https://www.arduino.cc. [Último acceso: 11 11 2017].

[9] ARDUINO, «ARDUINO UNO,» [En línea]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Último acceso: 2017].

[10] Atmel, «Atmega328/P,» [En línea]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Summary.pdf. [Último acceso: 2017].

[11] A. Arduino, «Shield para Arduino,» [En línea]. Available: https://aprendiendoarduino.wordpress.com/2015/03/23/shields-para-arduino/. [Último acceso: 2017].

[12] J. M. R. Gutiérrez, «Arduino + Ethernet Shield,» [En línea]. Available: http://unicarlos.com/_ARDUINO/Arduino%20+%20Ethernet%20Shield%20(1).pdf.

[13] A. Forum, «No tengo la mac address de la arduino ethernet shield,» 26 05 2012. [En línea]. Available: http://forum.arduino.cc/index.php?topic=107416.0. [Último acceso: 2017].

[14] Mouser, «Arduino Ethernet Shield,» [En línea]. Available: https://www.mouser.com/catalog/specsheets/A000056_DATASHEET.pdf. [Último acceso: 2017].

[15] rambal, «Shield SIM800c GPRS/GSM para Arduino,» [En línea]. Available: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0ahUKEwjOkqHIspjbAhWnwFkKHREACqcQFggyMAE&url=http%3A%2F%2Frambal.com%2Findex.php%3Fcontroller%3Dattachment%26id_attachment%3D921&usg=AOvVaw10-tvXGWqtVG79BkU-6jjF. [Último acceso: 2017].

[16] Espressif, «ESP8266EX DATASHEET,» [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. [Último acceso: 2017].

[17] J. F. P. J. R. Robert Salas, «TECNICAS DE DISEÑO, DESARROLLO Y MONTAJE DE CIRCUITOS IMPRESOS,» [En línea]. Available: https://f20b4575-a-62cb3a1a-s-sites.googlegroups.com/site/labcircuitoselectricos/assignments/practica2/tecnicas_diseno.pdf?attachauth=ANoY7crZ93kK_j1BiyhN9gclCyc3RBegf066_NtaE2s9GwGmvhgrYwNx8wUQHmt1BYnTiUmLLihNdGvtLpY_54qGOWJdZLnNnPUMheXI-_frC0gRMAjLY4vJi. [Último acceso: 12 11 2017].

[18] ARDUINO, «Ethernet / Ethernet 2 library,» [En línea]. Available: https://www.arduino.cc/en/Reference/Ethernet.

[19] R. N. TUTORIALS, «How to Install the ESP8266 Board in Arduino IDE,» [En línea]. Available: http://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/.

[20] M. Sandel, «COMANDOS AT PARA MODEM GSM,» 05 11 2012. [En línea]. Available: https://miguelsandel.wordpress.com/2012/11/05/comandos-at-para-los-modem-gsm/.

[21] J. M. R. Gutiérrez, «Manual de Programación Arduino,» [En línea]. Available:

https://arduinobot.pbworks.com/f/Manual+Programacion+Arduino.pdf. [Último acceso: 12 11 2017].

[22] g. d. canarias, «Características técnicas del ARDUINO UNO,» 05 2013. [En línea]. Available: http://www3.gobiernodecanarias.org/medusa/ecoblog/ralvgon/files/2013/05/Caracter%C3%ADsticas-Arduino.pdf.

[23] ARDUINO, «MODULO SHIEL ETHERNET,» [En línea]. Available: https://store.arduino.cc/usa/arduino-ethernet-shield-2.

[24] J. M. R. Gutiérrez, «Manual de Programación Arduino,» [En línea]. Available: https://arduinobot.pbworks.com/f/Manual+Programacion+Arduino.pdf.

[25] A. Rubio, «10 códigos html para páginas web y para que sirven,» 19 07 2017. [En línea]. Available: https://pe.godaddy.com/blog/10-codigos-html-para-paginas-web-y-para-que-sirven/.

[26] PRPMETEC, «ARDUINO Y WIFI ESP8266,» [En línea]. Available: https://www.prometec.net/arduino-wifi/.

[27] E. S. I. Team, «ESP8266EX Datasheet,» [En línea]. Available: https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf.

[28] rambal, «Shield SIM800C GPRS/GSM para Arduino,» [En línea]. Available: http://rambal.com/shields-arduino/687-shield-sim800c-gprs-gsm-para-arduino.html.

[29] G. wiki, «GPRS Shield V2.0,» [En línea]. Available: http://www.geeetech.com/wiki/index.php/GPRS_Shield_V2.0.

# ANEXOS

## ANEXO A: SUSTENTO TEÓRICO

## ANEXO A1: ARDUINO UNO

La placa Arduino Uno posee un microcontrolador Atmel 328/P, el cual es el núcleo del Arduino, incluye reguladores de tensión, un puerto USB el cual permite programar el microcontrolador desde cualquier PC. También dispone de 14 pines que se pueden configurar como entradas o salidas digitales y pines de entradas analógicas, los cuales permiten recolectar datos de sensores en forma de variaciones de voltaje; los pines de salida analógica pueden ser utilizados para obtener señales PWM [22].

El Arduino Uno convencional tiene un microcontrolador Atmega en formato DIP (desmontable), se lo puede observar en la Figura A1.1

*Figura A1. 1 Arduino Uno convencional* [22]

### ENTRADAS Y SALIDAS

Los 14 pines digitales pueden ser usados como entradas o salidas, funcionan a 5V y puede suministrar hasta 40mA. Cada uno de estos pines dispone de una resistencia interna de pull-up que son usadas para eliminar los rebotes de corriente que existen al conectar pulsadores, interruptores o cualquier elemento que permita el paso de corriente. Cada resistencia interna tiene un valor de 20KΩ a 50KΩ, que siempre se encuentra desconectada hasta que el usuario decida usarlas. Los pines

digitales se encuentran a la izquierda del microcontrolador y son identificados con números que corresponden a los pines [22].

- Los pines 0 y 1 digitales pertenecen a la Transmisión y Recepción respectivamente, son usados para transmisiones serie de señales TTL.
- Los pines 2 y3 están configurados para que generen interrupciones en el Atmega. Las interrupciones se activan únicamente cuando se detecta un valor bajo en las entradas.
- Los pines que poseen el signo ~, pertenecen a los pines PWM; son 6 pines en total y están destinados a generar señales PWM de hasta 8 bits.
- Los pines 10, 11, 12 y 13, son pines SPI los cuales permiten una comunicación Full Dúplex en un entorno Maestro-Esclavo.

Por último, se cuenta con un bus $I^2C$ el cual es utilizado para la interconexión de sistemas embebidos, actualmente una diversidad de dispositivos utilizan esta interfaz [22].

## ALIMENTACIÓN DEL ARDUINO UNO

Existe dos formas de alimentación de la placa, la primera es a través del cable USB propio de la placa o mediante una fuente externa. Si se utiliza alimentación externa, esta debe estar entre 6 y 12 voltios [22].

## RESUMEN DE CARACTERÍSTICAS

El resumen de características se puede observar en la Tabla A1.1

***Tabla A1. 1 Resumen de características [22]***

| | |
|---|---|
| Microcontrolador | Atmega328 |
| Voltaje de operación | 5V |
| Voltaje de entrada (Recomendado) | 7 – 12V |
| Voltaje de entrada (Límite) | 6 – 20V |
| Pines para entrada- salida digital. | 14 (6 pueden usarse como salida de PWM) |
| Pines de entrada analógica. | 6 |
| Corriente continua por pin IO | 40 mA |
| Corriente continua en el pin 3.3V | 50 mA |
| Memoria Flash | 32 KB (0,5 KB ocupados por el bootloader) |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Frecuencia de reloj | 16 MHz |

## ANEXO A2: ATMEGA 328/P

El microcontrolador ATmega 328P es un microcontrolador de 8 bits, basado en la arquitectura RISC. Tiene un muy alto rendimiento, ya que cuenta con una memoria de 32KB ISP FLASH con capacidad de lectura y escritura y posee 1KB de memoria EEPROM. Incorpora 23 pines de entrada/salida de propósito general y 32 registros de propósito general, 3 timer y contadores, 1 puerto USART programable, en la Tabla A2.1 se puede observar más características [10].

***Tabla A2. 1 Características ATmega 328P [10]***

| Características | ATmega328/P |
|---|---|
| Número de Pines | 28/32 |
| Memoria Flash (Bytes) | 32K |
| Memoria SRAM (Bytes) | 2K |
| Memoria EEPROM (Bytes) | 1K |
| Tamaño del Vector de Interrupción (palabra de instrucción/vector) | 1/1/2 |
| Líneas de I/O de Propósito General | 23 |
| SPI | 2 |
| TWI (I2C) | 1 |
| USART | 1 |
| ADC | 10-bit 15kSPS |
| Canales ADC | 8 |
| Contador/Timer 8-bit | 2 |
| Contador/Timer 16-bit | 1 |

Este microcontrolador está diseñado para el uso en la automatización industrial y la automatización de viviendas y edificios.

Su frecuencia de trabajo es de máximo de 20MHz y trabaja con una alimentación entre 1.8 a 5.5 voltios [10].

La distribución de pines se lo puede observar en la Figura A2.1

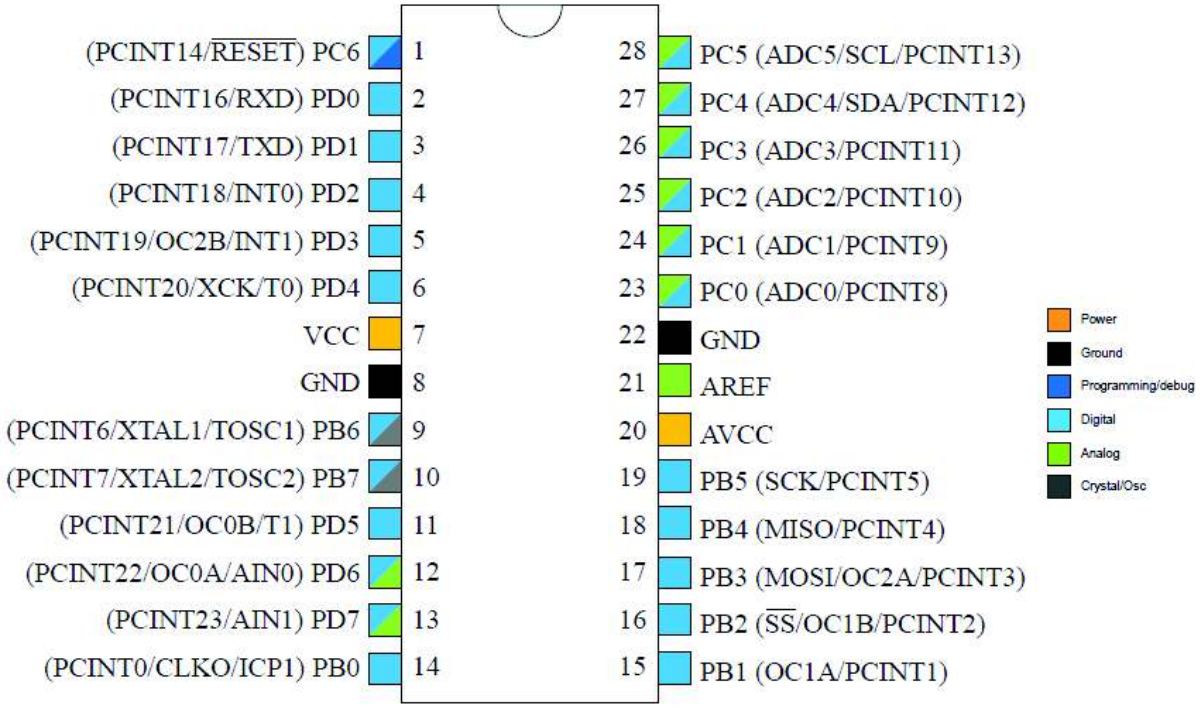



*Figura A2. 1 Distribución de pines ATmega 328P* [10]

El pórtico B, cuenta con 8 bits entrada/salida y resistencias pull-up; los pines PB6 y PB7 pueden usarse como entrada al amplificador de oscilador inversor.

El pórtico C, cuenta con 7 bits de entrada/salida de propósito general y resistencias internas pull-up; el pin PC6/RESET, es diferente al resto de pines de este pórtico ya que, posee un fusible no programado, si este es activado genera un reinicio en el microcontrolador.

El pórtico D, cuenta con 8 bits entrada/salida y resistencias pull-up [10].

## ANEXO A3: MÓDULO SHIELD ETHERNET

El módulo *Ethernet Shield* permite que el Arduino Uno se conecte a Internet. Se encuentra basado en el microcontrolador *Wiznet* W5500 *Ethernet* Chip, el cual proporciona una pila de red capaz de utilizar protocolos TCP y UDP. También admite hasta ocho conexiones simultáneas.

Este módulo tiene una conexión estándar RJ-45, con un transformador de líneas integrado y alimentación a través de *Ethernet*. Posee un controlador de reinicio para garantizar que el *Wiznet* W5500 se reinicie correctamente.

Tiene la posibilidad de ingresar una tarjeta micro-SD que permite almacenar archivos que pueden servir en la red como, la configuración de una página Web más compleja.

Cumple con el estándar IEEE802.3af, tiene un rango de voltaje de entrada de 36V a 57V, protección contra sobrecargas y cortocircuitos y salida de 12V [23].

El módulo Arduino utiliza el bus SPI para comunicarse con las tarjetas W5500 y SD, las cuales se encuentran ubicadas en los pines 10, 11, 12 y 13 del Arduino; estos pines no se podrán usar como pines de entrada/salida general. También, se debe tener en cuenta que debido a que el W5500 y el SD utilizan el bus SPI, solo se puede trabajar uno cada vez, caso contrario generaría conflictos [23].

El módulo *Shield Ethernet* posee LEDs que describen el funcionamiento del mismo, como se muestra en la Figura A3.1 [23]

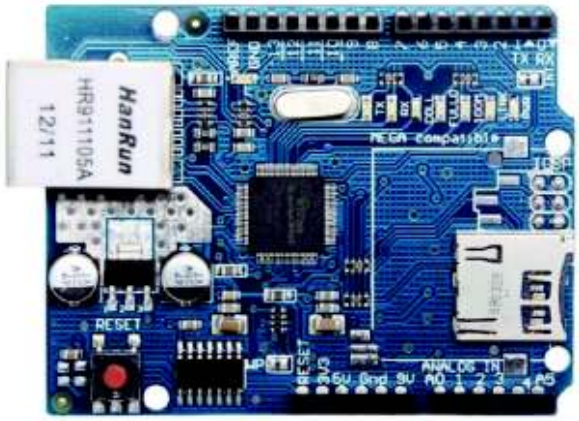



*Figura A3. 1 Módulo Shield Ethernet* [23]

PWR: indica que la placa y el *shield* están alimentados.

LINK: indica la presencia de un enlace de red y parpadea cuando el *shield* envía o recibe datos.

FULLD: indica que la conexión de red es full dúplex.

100M: indica la presencia de un enlace de red de 100Mb/s.

RX: si parpadea indica que se encuentra recibiendo datos.

TX: si parpadea indica que se encuentra transmitiendo datos.

COLL: parpadea si detecta colisiones en la red.

## ANEXO A4: MÓDULO *SHIELD* GSM

El módulo GSM/GPRS 800C utiliza la red de telefonía celular GSM (*Global System for Mobile communication*). Puede recibir información desde una ubicación remota; además, este módulo es compatible con las placas Arduino Uno. Figura A4.1 **[15]**.

*Figura A4. 1 Módulo Shield GSM SIM 800C* [15]

Este módulo *Shield* puede enviar y recibir mensajes de texto, recibir y realizar llamadas y hacer seguimiento por medio de GPS ya que entrega señales GSM y GPRS a una frecuencia de 1900MHz para el servicio de audio, SMS y GPRS **[15]**.

El *Shield* posee un bajo consumo de energía alrededor de 0.6mA; además, cumple con las normas GSM de Clase 1, las cuales trabajan a una frecuencia de 1800 a 1900MHz y una potencia de 1 W; y Clase 4 con una frecuencia de 850 a 900 MHz y una potencia de 2 W [15].

**Especificaciones técnicas**

- Voltaje de entrada: 5 a 18 V.
- Voltaje de operación: 5V.

- Comunicación al módulo Arduino: UART.
- Frecuencia de trabajo: 850, 900, 1800 y 1900MHz.
- Posee 2 antenas, una antena GPRS tradicional y otra antena más compacta y de fácil instalación.
- Posee una entrada para la instalación de un Chip de telefonía móvil tipo MicroSim.
- Control a través de comandos AT para enviar y recibir mensajes de texto y llamadas.
- Envía y recibe datos GPRS como comunicaciones con protocolos TCP/UDP/FTP y HTTP
- LEDs para indicar: estado de la fuente de alimentación, red y modo de funcionamiento [15].

## ANEXO A5: MÓDULO *WIFI* ESP8266EX

El módulo ESP8266 ofrece una solución *WiFi*, altamente integrada que satisface las necesidades de los usuarios, y de las demandas continuas de uso de energía eficiente, diseño y rendimiento confiable, se lo puede observar en la Figura A5.1 [16].

*Figura A5.1 Módulo WiFi ESP8266EX [16].*

El ESP8266 se puede comunicar con cualquier microcontrolador y con todas las placas Arduino y utiliza su lenguaje de programación a través de las interfaces que posee, SPI/I2C/UART [16].

Puede controlar  sensores externos al interconectarlos a sus pines GPIO.

Soporta la tecnología 802.11 b/g/n. Es compatible con infraestructura BSS (*Basic Service Set*), P2P y *SoftAP* [16].

Trabaja a una frecuencia de 2.4GHz.

A continuación, en la Tabla A5.1, se presenta más especificaciones técnicas.

## *Tabla A5. 1 Especificaciones ESP8266 [16]*

| Categoría | Detalle | Parámetros |
|---|---|---|
| *WiFi* | Certificación | *WiFi* Alianza |
| | Protocolos | 802.11 b/g/n |
| | Rango de Frecuencia | 2,5G - 2,5G (2400M - 2483.5M) |
| | Poder de Transmisión | 802.11 b: +20 dBm |
| | | 802.11 g: +17 dBm |
| | | 802.11 n: +14 dBm |
| | Sensibilidad de Recepción | 802.11 b: -91 dBm (11 Mbps) |
| | | 802.11 b: -75 dBm (54 Mbps) |
| | | 802.11 b: -72 dBm (MCS7) |
| | Antena | Vestigio PBC, Externo, IPEX conector, Chip Cerámico |
| *Hardware* | CPU | Procesador Tensilica L106 32-bit |
| | Interface Periférica | UART/SDIO/SPI/I2C/I2S/IR Control Remoto |
| | | GPIO/ADC/PWM/LED Luz y Botón |
| | Voltaje de operación | 2.5V - 3.6V |
| | Corriente de operación | Valor Promedio: 80 mA |
| | Rango de Temperatura de Operación | -40°C - 125°C |
| | Rango de Temperatura de Almacenamiento | -40°C - 125°C |
| | Medida del Encapsulado | QFN32-pin (5 mm x 5mm) |
| | Interface externo | - |
| *Software* | *WiFi* Mode | Station/SoftAP/SoftAP+Station |
| | Seguridad | WPA/WPA2 |
| | Encriptación | WEP/TKIP/AES |
| | Actualización de *Firmware* | UART Descarga/ OTA (vía red) |
| | Desarrollo de Software | Soporta Desarrollo de Servidor en la Nube/ Firmware y SDK para rápida programación en Chip encendido |
| | Protocolos de Red | IPV4, TCP/UDP/HTTP/FTP |
| | Configuración de Usuario | Conjunto de instrucciones AT, Servidor en la Nube, Android/iOS App |

# ANEXO B: HOJAS GUÍA DE INSTRUCTOR

# ANEXO B1
# PRÁCTICA 1: MANEJO DE PUERTOS DE ENTRADA Y SALIDA UTILIZANDO EL MÓDULO ARDUINO UNO.

# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

## HOJA GUÍA PARA EL INSTRUCTOR

## PRÁCTICA 1

**1. TEMA:** Manejo de puertos de entrada y salida utilizando el módulo Arduino Uno.

**2. OBJETIVOS**

2.1. Relacionar al estudiante con la programación de módulos Arduino utilizando el IDE del mismo.

2.2. Implementar un circuito utilizando las placas didácticas de Arduino Uno para el manejo de puertos de entrada/salida.

**3. DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA**

3.1. Elaborar un programa que permita realizar dos juegos de luces controlados por un potenciómetro.

Primer juego de luces: Cuando el potenciómetro se encuentre de la mitad hacia abajo de su capacidad (menor a 512 del ADC del Arduino), 6 leds totalmente encendidos deben apagarse uno a uno hasta llegar al final, y los que queden atrás del apagado deben encenderse gradualmente y repetir indefinidamente.

Segundo juego de luces: Cuando el potenciómetro se encuentre de la mitad hacia arriba de su capacidad (mayor a 512del ADC del Arduino) los 6 leds deberán encenderse ya sea de abajo hacia arriba o viceversa gradualmente y repetir indefinidamente.

3.2. El valor del potenciómetro deberá ser indicado en el LCD de tal forma que el mínimo de su capacidad debe ser 0 y su máximo 1024.

3.3. Armar el circuito necesario utilizando las placas didácticas de Arduino Uno, grabar en el Arduino el programa desarrollado y comprobar su funcionamiento

3.4. Elaborar otro programa que permita ingresar el código binario a través de 4 puertos de un dipswitch y que se presente el dígito correspondiente en un display de 7 segmentos ánodo común. Se deberá observar en el display los dígitos del 0 al 9.

3.5. Armar el circuito necesario utilizando las placas didácticas de Arduino Uno, grabar en el Arduino el programa desarrollado y comprobar su funcionamiento

**4. MATERIALES**

- Placa didáctica Arduino Uno
- Computadora con el IDE de Arduino instalado
- Cable USB tipo A/B
- Cables para protoboard

## 5. ENSAMBLAJE DEL CIRCUITO

***Nota 1:** Se aconseja desconectar el USB mientras se ensambla para evitar corto-circuitos por cables mal conectados, además se recomienda una revisión antes de energizar.

***Nota 2:** Tratar de no usar el potenciómetro de contraste del LCD, y si es necesario no llevarlo a sus extremos por que podría ocasionar una ligera salida de humo del potenciómetro debido a que en su valor  minino se produciría un corto-circuito.

**EJEMPLO 1:**

5.1. Conexión del LCD al Arduino: Con los cables de protoboard proporcionados, conectar:

- El pin 2 del Arduino a RS de la Placa Didáctica (LCD).
- El pin 4 del Arduino a E de la Placa Didáctica (LCD).
- El pin 7 del Arduino a D4 de la Placa Didáctica (LCD).
- El pin 8 del Arduino a D5 de la Placa Didáctica (LCD).
- El pin 12 del Arduino a D6 de la Placa Didáctica (LCD).
- El pin 13 del Arduino a D7 de la Placa Didáctica (LCD).

5.2. Conexión de leds al Arduino, conectar:

- El pin 11 del Arduino a D1 de la Placa Didáctica.
- El pin 10 del Arduino a D2 de la Placa Didáctica.
- El pin 9 del Arduino a D3 de la Placa Didáctica.
- El pin 6 del Arduino a D4 de la Placa Didáctica.
- El pin 5 del Arduino a D5 de la Placa Didáctica.
- El pin 3 del Arduino a D6 de la Placa Didáctica.

5.3. Conexión del potenciómetro al Arduino, conectar:

- El pin A3 del Arduino a INPUT ANALOGIC de la Placa Didáctica

**EJEMPLO 2:**

5.4. Conexión del dipswitch al Arduino, conectar:

-El pin 12 del Arduino a SW1 de la Placa Didáctica.

-El pin 11 del Arduino a SW2 de la Placa Didáctica.

-El pin 10 del Arduino a SW3 de la Placa Didáctica.

-El pin 9 del Arduino a SW4 de la Placa Didáctica.

5.5. Conexión del display de 7 segmentos al Arduino.

En este caso se puede utilizar cualquier display ubicado en la Placa Didáctica.

-El pin 8 del Arduino a A (Display de 7 segmentos) de la Placa Didáctica.

-El pin 7 del Arduino a B (Display de 7 segmentos) de la Placa Didáctica.

-El pin 6 del Arduino a C (Display de 7 segmentos) de la Placa Didáctica.

-El pin 5 del Arduino a D (Display de 7 segmentos) de la Placa Didáctica.

-El pin 4 del Arduino a E (Display de 7 segmentos) de la Placa Didáctica.

-El pin 3 del Arduino a F (Display de 7 segmentos) de la Placa Didáctica.

-El pin 2 del Arduino a G (Display de 7 segmentos) de la Placa Didáctica.

## 6. DESARROLLO DE LA PRÁCTICA

**EJERCICIO 1**

**TIEMPO ESTIMADO DE LA PRÁCTICA:** 1 hora – 1 hora y 30 minutos

6.1. Abrir el programa: *"ARDUINO"*

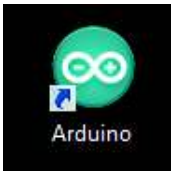



*Figura B1. 1 Icono del software Arduino*

6.2. Se abrirá una ventana como se muestra a continuación.

*Figura B1. 2 Ventana del software Arduino*

Se puede observar que ya se encuentra escrito las dos partes importantes del programa, las cuales son: el void setup y void loop

6.3. Se declaran las variables que se utilizarán en el programa.

Estas variables deben ser escritas arriba del *void setup (),* de la siguiente manera:

```
1  //Declaración de constantes para salidas pwm
2  const int led1 = 11;
3  const int led2 = 10;
4  const int led3 = 9;
5  const int led4 = 6;
6  const int led5 = 5;
7  const int led6 = 3;
8  //Declaracion del potenciometro
9  const int pot = 3;
10 //declaracion de Variables
11 int brillo;
12 int tiempo=1,tiempo2=100;
13 int boton1,boton2,boton3;
14 int potenciometro;
```

*Figura B1. 3 Inicialización de variables*

6.4. Como se va a utilizar un LCD, el IDE de Arduino nos facilita librerías que nos ayuda a la programación de este componente.

Dirigirse a **Archivo→Ejemplos→LiquidCristal→Display.**

Se abrirá una ventana como la siguiente:

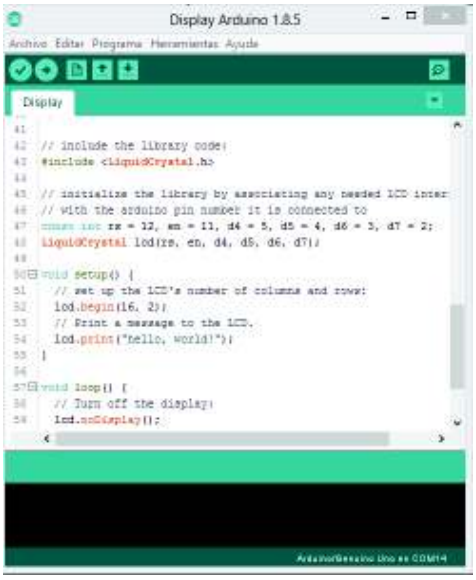



*Figura B1. 4 Void loop*

En esta ventana (Figura B1.4) se observa varias líneas de código, de las cuales por el momento utilizaremos:

#include <LiquidCrystal.h> y LiquidCrystal lcd(RS, EN, D4, D5, D6, D7).

En el programa colocar los códigos anteriores en el mismo lugar donde se declararon las variables y cerramos el ejemplo que abrimos.

```
1  //LCD Incialización
2  #include <LiquidCrystal.h>
3  LiquidCrystal lcd(2, 4, 7, 8, 12, 13); //RS,E,D4,D5,D6,D7
4  //Declaración de constantes para salidas pwm
5  const int led1 = 11;
6  const int led2 = 10;
7  const int led3 = 9;
8  const int led4 = 6;
9  const int led5 = 5;
10 const int led6 = 3;
11 //Declaracion del potenciometro
12 const int pot = 3;
13 //declaracion de Variables
14 int brillo;
15 int tiempo=1,tiempo2=100;
16 int boton1,boton2,boton3;
17 int potenciometro;
```

*Figura B1. 5*

Es importante saber que los pines a utilizarse para el LCD pueden ser modificados como se puede observar en la Figura B1.5, pero se modifica también las conexiones del circuito.

6.5. En el *void setup.*

```
18 void setup() {
19   //Mensaje de bienvenida del LCD
20   lcd.begin(16,2); //Posición de texto en el LCD
21   lcd.print("Bienvenidos"); //Mensaje
22   delay(2000);
23   lcd.clear(); // Se limpia el LCD
24   //Establecer salidas pwm
25   pinMode(led1,OUTPUT);
26   pinMode(led2,OUTPUT);
27   pinMode(led3,OUTPUT);
28   pinMode(led4,OUTPUT);
29   pinMode(led5,OUTPUT);
30   pinMode(led6,OUTPUT);
31 }
```

*Figura B1. 6*

Se puede observar que se da un mensaje de bienvenida en el LCD, el cual aparecerá una sola vez en el programa, también se puede notar que se establecieron salidas PWM.

Es importante recalcar que led1, led2, led3… tienen un valor correspondiente a un número el cual, se puede observar en la sección de la declaración de variables y que corresponde a un pin del Arduino.

```
const int led1 = 11;
const int led2 = 10;
const int led3 = 9;
const int led4 = 6;
const int led5 = 5;
const int led6 = 3;
```

*Figura B1. 7*

6.6. Una vez declaradas todas las variables y pines se procede a la programación, la cual puede variar en cada estudiante, sin embargo aquí se presenta una opción.

```
void loop() {
 lcd.clear();
potenciometro=analogRead(pot);
lcd.setCursor(0, 1);
lcd.print("Valor ADC: ");
lcd.setCursor(12, 1);
 lcd.print(potenciometro);
if(potenciometro<600)
{
//Juego de luces 1
lcd.setCursor(0, 0);
 lcd.print("Juego de luces 1");
brillo=0;
while(brillo<255)
{
analogWrite(led1, brillo);
delay(tiempo);
analogWrite(led2, brillo-43);
delay(tiempo);
analogWrite(led3, brillo-(43*2));
delay(tiempo);
analogWrite(led4, brillo-(43*3));
delay(tiempo);
analogWrite(led5, brillo-(43*4));
delay(tiempo);
analogWrite(led6, brillo-(43*5));
delay(tiempo);
brillo++;
```

*Figura B1. 8*

```
60  }
61  }
62  if(potenciometro>=600)
63  {
64  //Juego de luces 2
65  lcd.setCursor(0, 0);
66   lcd.print("Juego de luces 2");
67
68  brillo=256;
69  while(brillo > 255)
70  {
71  delay(tiempo2);
72  analogWrite(led1, brillo);
73  analogWrite(led2, 0);
74  analogWrite(led3, 0);
75  analogWrite(led4, 0);
76  analogWrite(led5, 0);
77  analogWrite(led6, 0);
78  delay(tiempo2);
79  analogWrite(led1, brillo-150);
80  analogWrite(led2, brillo);
81  analogWrite(led3, 0);
82  analogWrite(led4, 0);
83  analogWrite(led5, 0);
84  analogWrite(led6, 0);
85  delay(tiempo2);
86  analogWrite(led1, brillo-180);
87  analogWrite(led2, brillo-150);
```

*Figura B1. 9*

```
88  analogWrite(led3, brillo);
89  analogWrite(led4, 0);
90  analogWrite(led5, 0);
91  analogWrite(led6, 0);
92  delay(tiempo2);
93  analogWrite(led1, brillo-210);
94  analogWrite(led2, brillo-180);
95  analogWrite(led3, brillo-150);
96  analogWrite(led4, brillo);
97  analogWrite(led5, 0);
98  analogWrite(led6, 0);
99  delay(tiempo2);
100 analogWrite(led1, brillo-254);
101 analogWrite(led2, brillo-210);
102 analogWrite(led3, brillo-180);
103 analogWrite(led4, brillo-150);
104 analogWrite(led5, brillo);
105 analogWrite(led6, 0);
106 delay(tiempo2);
107 analogWrite(led1, brillo-254);
108 analogWrite(led2, brillo-250);
109 analogWrite(led3, brillo-210);
110 analogWrite(led4, brillo-180);
111 analogWrite(led5, brillo-150);
112 analogWrite(led6, brillo);
113 delay(tiempo2);
114 analogWrite(led1, 0);
115 analogWrite(led2, 0);
```

*Figura B1. 10*

```
116  analogWrite(led3, 0);
117  analogWrite(led4, 0);
118  analogWrite(led5, 0);
119  analogWrite(led6, brillo);
120  delay(tiempo2);
121  analogWrite(led1, 0);
122  analogWrite(led2, 0);
123  analogWrite(led3, 0);
124  analogWrite(led4, 0);
125  analogWrite(led5, brillo);
126  analogWrite(led6, brillo-150);
127  delay(tiempo2);
128  analogWrite(led1, 0);
129  analogWrite(led2, 0);
130  analogWrite(led3, 0);
131  analogWrite(led4, brillo);
132  analogWrite(led5, brillo-150);
133  analogWrite(led6, brillo-180);
134  delay(tiempo2);
135  analogWrite(led1, 0);
136  analogWrite(led2, 0);
137  analogWrite(led3, brillo);
138  analogWrite(led4, brillo-150);
139  analogWrite(led5, brillo-180);
140  analogWrite(led6, brillo-210);
141  delay(tiempo2);
142  analogWrite(led1, 0);
143  analogWrite(led2, brillo);
```

*Figura B1. 11*

```
144  analogWrite(led3, brillo-150);
145  analogWrite(led4, brillo-180);
146  analogWrite(led5, brillo-210);
147  analogWrite(led6, brillo-250);
148  delay(tiempo2);
149  analogWrite(led1, brillo);
150  analogWrite(led2, brillo-150);
151  analogWrite(led3, brillo-180);
152  analogWrite(led4, brillo-210);
153  analogWrite(led5, brillo-250);
154  analogWrite(led6, brillo-254);
155  delay(tiempo2);
156  brillo-- ;
157  }
158  }
159  }
```

*Figura B1. 12*

6.7 Una vez concluido el programa, se procede a guardar el archivo en una carpeta para eso dirigirse a: ***Archivo →Salvar →"Seleccionar donde se guardará el archivo"***. Ahora se puede cargar el programa en el módulo Arduino Uno, para ello conectar el módulo Arduino a la PC con el cable USB-AB (Figura B1.13) y seleccionar la placa que se va a usar (Figura B1.14), y el puerto COM al que se encuentra conectado (Figura B1.15). Ya configurado lo anterior seleccionar el botón subir (Figura B1.16)

*Figura B1. 13 Cable USB-AB*

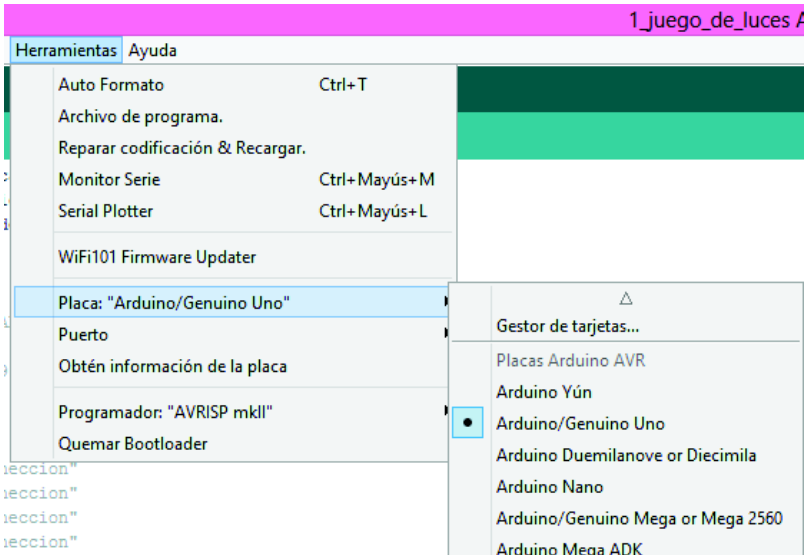



*Figura B1. 14 Selección de placa*

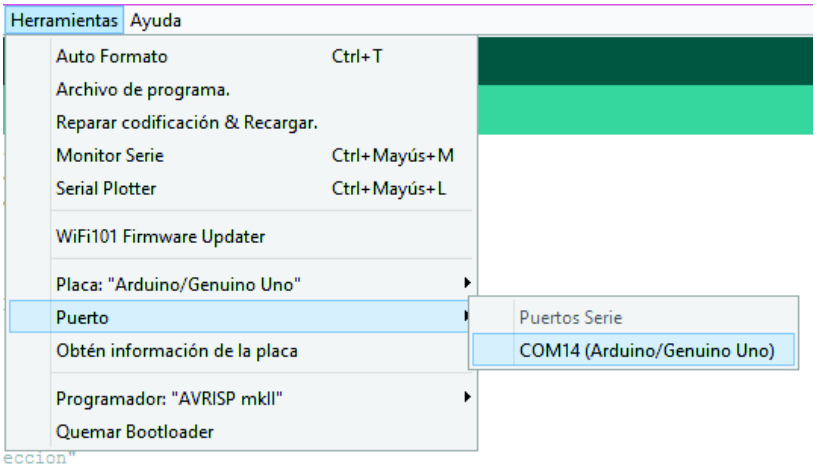



*Figura B1. 15*


*Figura B1. 16 Selección de puerto*

6.8 La Figura B1.17 muestra que se subió con éxito el programa

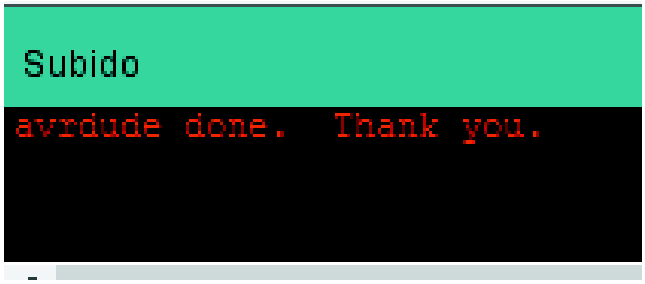



*Figura B1. 17 Confirmación de carga*

6.9. Presionar el botón PWLCD para encender el LCD

6.10 En las Figura B1.18 y Figura B1.19 se observa los dos juegos de luces respectivamente

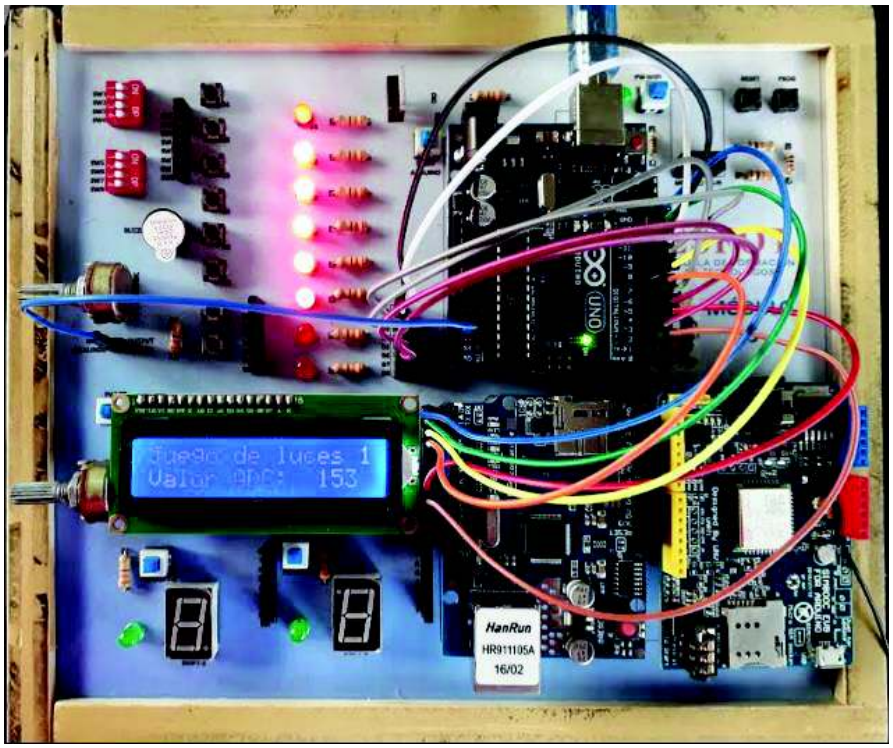



*Figura B1. 18 Juego de luces 1*

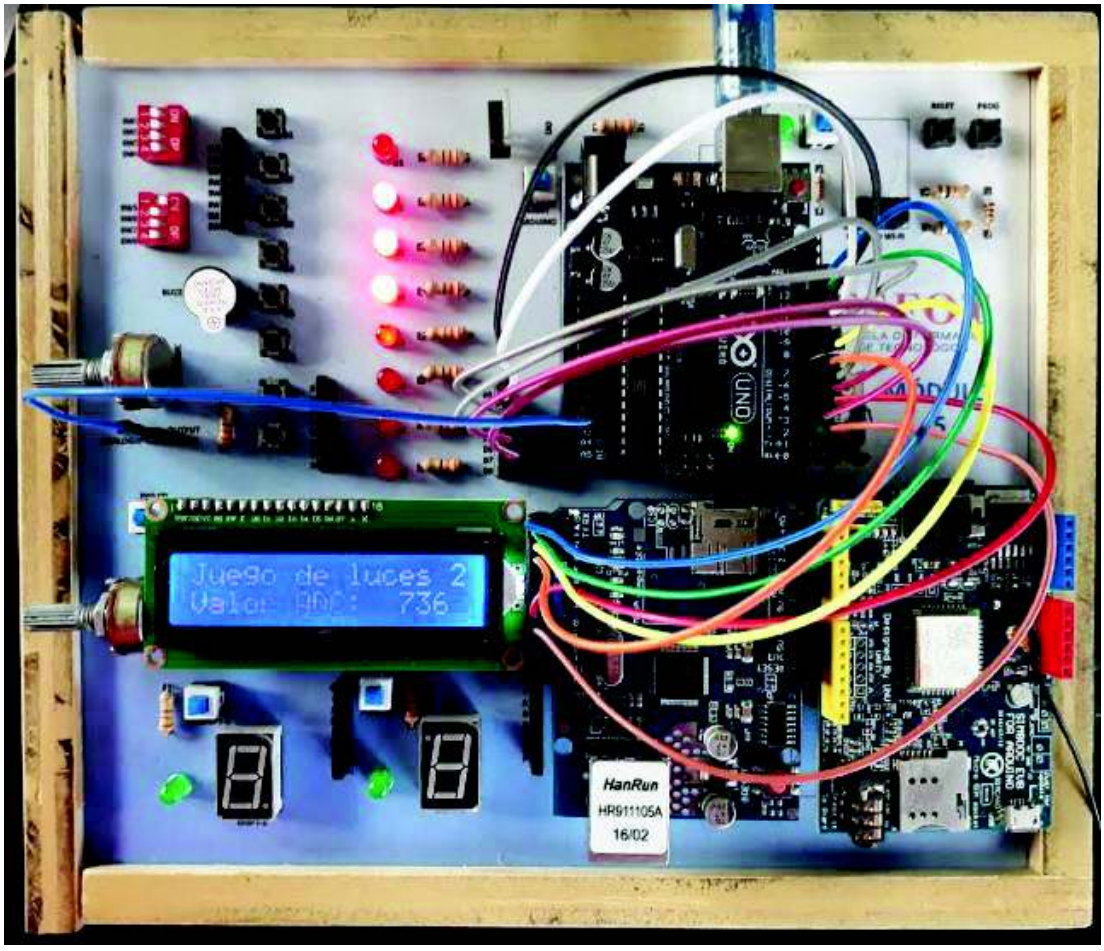



*Figura B1. 19 Juego de luces 2*

***NOTA 3**: Es necesario verificar que todos los botones (PWDISP7-1, PWDISP7-2, RESET ARDUINO y PW-WIFI) no se encuentren presionados.

***NOTA 4:** Una vez terminada la práctica es necesario que se retire todos los cables y se regresen todos sus componentes a sus posiciones originales

***NOTA 5:** En el caso de aparecer caracteres extraños en el LCD aplastar el botón **Reset del Arduino** o desconectar y conectar el cable USB de la placa.

## EJERCICIO 2

**TIEMPO ESTIMADO DE LA PRÁCTICA:** 45 minutos

6.11. Abrir el programa: *"ARDUINO"*


*Figura B1. 20 Icono del software Arduino*

6.12. Se abrirá una ventana como se muestra a continuación

*Figura B1. 21 Ventana del software Arduino*

Se observa que ya se encuentra escrito las dos partes importantes del programa, las cuales son: el void setup y void loop

6.13 Asignar los pines correspondientes al dipswitch

```
1 // ASIGNACION DE PINS DE ENTRADA DE BITS
2 int bit0=12; //LSB
3 int bit1=11;
4 int bit2=10;
5 int bit3=9;  //MSB
```

*Figura B1. 22*

Donde el bit menos significativo es el pin 12 y el más significativo es el 9.

6.14 Se asigna los pines correspondientes al display de 7 segmentos.

```
6  //ASIGNACIÓN DE PINS PARA SALIDA DISPLAY
7  int a=8; //A
8  int b=7; //B
9  int c=6; //C
10 int d=5; //D
11 int e=4; //E
12 int f=3; //F
13 int g=2; //G
```

*Figura B1. 23*

6.15 Se declara el modo de los pines del Arduino, como se muestra en la Figura B1.24

```
14 void setup() {
15 // PINS DE ENTRADA PARA EL DIPSWITCH
16 pinMode(bit0,INPUT); // ENTRADA PARA EL BIT LSB
17 pinMode(bit1,INPUT); // ENTRADA PARA EL BIT1
18 pinMode(bit2,INPUT); // ENTRADA PARA EL BIT2
19 pinMode(bit3,INPUT); // ENTRADA PARA EL BIT MSB
20 digitalWrite(bit0,HIGH);//PULL-UP DEL PIN 12
21 digitalWrite(bit1,HIGH);//PULL-UP DEL PIN 11
22 digitalWrite(bit2,HIGH);//PULL-UP DEL PIN 10
23 digitalWrite(bit3,HIGH);//PULL-UP DEL PIN 9
24
25 // PINS DE SALIDA PARA EL DISPLAY DE 7 SEGMENTOS
26
27 pinMode(a,OUTPUT); // SALIDA PARA EL LED A DEL DISPLAY DE 7 SEGMENTOS
28 pinMode(b,OUTPUT); // SALIDA PARA EL LED B DEL DISPLAY DE 7 SEGMENTOS
29 pinMode(c,OUTPUT); // SALIDA PARA EL LED C DEL DISPLAY DE 7 SEGMENTOS
30 pinMode(d,OUTPUT); // SALIDA PARA EL LED D DEL DISPLAY DE 7 SEGMENTOS
31 pinMode(e,OUTPUT); // SALIDA PARA EL LED E DEL DISPLAY DE 7 SEGMENTOS
32 pinMode(f,OUTPUT); // SALIDA PARA EL LED F DEL DISPLAY DE 7 SEGMENTOS
33 pinMode(g,OUTPUT); // SALIDA PARA EL LED G DEL DISPLAY DE 7 SEGMENTOS
34 }
```

*Figura B1. 24*

6.16 Realizar el código para el número 0 y 1

```
void loop() {
// LEER EL NÚMERO BINARIO 0000 INGRESADO EN EL DIPSWITCH Y MOSTRAR EN EL DISPLAY DE 7 SEGMENTOS
if(digitalRead(bit0)==0&&digitalRead(bit1)==0&&digitalRead(bit2)==0&&digitalRead(bit3)==0){
  digitalWrite(a,LOW); //SALIDA DE 0 LÓGICO, YA QUE ES UN DISPLAY ÁNODO COMÚN
  digitalWrite(b,LOW); //SALIDA DE 0 LÓGICO
  digitalWrite(c,LOW); //SALIDA DE 0 LÓGICO
  digitalWrite(d,LOW); //SALIDA DE 0 LÓGICO
  digitalWrite(e,LOW); //SALIDA DE 0 LÓGICO
  digitalWrite(f,LOW); //SALIDA DE 0 LÓGICO
  digitalWrite(g,HIGH); //SALIDA DE 1 LÓGICO
}
// LEER EL NÚMERO BINARIO 0001 INGRESADO EN EL DIPSWITCH Y MOSTRAR EN EL DISPLAY DE 7 SEGMENTOS
int valora=digitalRead(bit0);
if(digitalRead(bit0)==1&&digitalRead(bit1)==0&&digitalRead(bit2)==0&&digitalRead(bit3)==0){
  digitalWrite(a,HIGH);
  digitalWrite(b,LOW);
  digitalWrite(c,LOW);
  digitalWrite(d,HIGH);
  digitalWrite(e,HIGH);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
}
```

*Figura B1. 25*

6.17 Para los números 2 y 3

```
// LEER EL NÚMERO BINARIO 0010 INGRESADO EN EL DIPSWITCH Y MOSTRAR EN EL DISPLAY DE 7 SEGMENTOS
if(digitalRead(bit0)==0&&digitalRead(bit1)==1&&digitalRead(bit2)==0&&digitalRead(bit3)==0){
  digitalWrite(a,LOW);
  digitalWrite(b,LOW);
  digitalWrite(c,HIGH);
  digitalWrite(d,LOW);
  digitalWrite(e,LOW);
  digitalWrite(f,HIGH);
  digitalWrite(g,LOW);
}
// LEER EL NÚMERO BINARIO 0011 INGRESADO EN EL DIPSWITCH Y MOSTRAR EN EL DISPLAY DE 7 SEGMENTOS
if(digitalRead(bit0)==1&&digitalRead(bit1)==1&&digitalRead(bit2)==0&&digitalRead(bit3)==0){
  digitalWrite(a,LOW);
  digitalWrite(b,LOW);
  digitalWrite(c,LOW);
  digitalWrite(d,LOW);
  digitalWrite(e,HIGH);
  digitalWrite(f,HIGH);
  digitalWrite(g,LOW);
}
```

*Figura B1. 26*

6.18 Para los números 4 y 5

```
// LEER EL NÚMERO BINARIO 0100 INGRESADO EN EL DIPSWITCH Y MOSTRAR EN EL DISPLAY DE 7 SEGMENTOS
if(digitalRead(bit0)==0&&digitalRead(bit1)==0&&digitalRead(bit2)==1&&digitalRead(bit3)==0){
  digitalWrite(a,HIGH);
  digitalWrite(b,LOW);
  digitalWrite(c,LOW);
  digitalWrite(d,HIGH);
  digitalWrite(e,HIGH);
  digitalWrite(f,LOW);
  digitalWrite(g,LOW);
}
// LEER EL NÚMERO BINARIO 0101 INGRESADO EN EL DIPSWITCH Y MOSTRAR EN EL DISPLAY DE 7 SEGMENTOS
if(digitalRead(bit0)==1&&digitalRead(bit1)==0&&digitalRead(bit2)==1&&digitalRead(bit3)==0){
  digitalWrite(a,LOW);
  digitalWrite(b,HIGH);
  digitalWrite(c,LOW);
  digitalWrite(d,LOW);
  digitalWrite(e,HIGH);
  digitalWrite(f,LOW);
  digitalWrite(g,LOW);
}
```

*Figura B1. 27*

6.19 Para los números 6 y 7

```
// LEER EL NÚMERO BINARIO 0110 INGRESADO EN EL DIPSWITCH Y MOSTRAR EN EL DISPLAY DE 7 SEGMENTOS
if(digitalRead(bit0)==0&&digitalRead(bit1)==1&&digitalRead(bit2)==1&&digitalRead(bit3)==0){
  digitalWrite(a,LOW);
  digitalWrite(b,HIGH);
  digitalWrite(c,LOW);
  digitalWrite(d,LOW);
  digitalWrite(e,LOW);
  digitalWrite(f,LOW);
  digitalWrite(g,LOW);
}
// LEER EL NÚMERO BINARIO 0111 INGRESADO EN EL DIPSWITCH Y MOSTRAR EN EL DISPLAY DE 7 SEGMENTOS
if(digitalRead(bit0)==1&&digitalRead(bit1)==1&&digitalRead(bit2)==1&&digitalRead(bit3)==0){
  digitalWrite(a,LOW);
  digitalWrite(b,LOW);
  digitalWrite(c,LOW);
  digitalWrite(d,HIGH);
  digitalWrite(e,HIGH);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
}
```

*Figura B1. 28*

6.20 Para los números 8 y 9

```
// LEER EL NÚMERO BINARIO 1000 INGRESADO EN EL DIPSWITCH Y MOSTRAR EN EL DISPLAY DE 7 SEGMENTOS
if(digitalRead(bit0)==0&&digitalRead(bit1)==0&&digitalRead(bit2)==0&&digitalRead(bit3)==1){
  digitalWrite(a,LOW);
  digitalWrite(b,LOW);
  digitalWrite(c,LOW);
  digitalWrite(d,LOW);
  digitalWrite(e,LOW);
  digitalWrite(f,LOW);
  digitalWrite(g,LOW);
}
// LEER EL NÚMERO BINARIO 1001 INGRESADO EN EL DIPSWITCH Y MOSTRAR EN EL DISPLAY DE 7 SEGMENTOS
if(digitalRead(bit0)==1&&digitalRead(bit1)==0&&digitalRead(bit2)==0&&digitalRead(bit3)==1){
  digitalWrite(a,LOW);
  digitalWrite(b,LOW);
  digitalWrite(c,LOW);
  digitalWrite(d,HIGH);
  digitalWrite(e,HIGH);
  digitalWrite(f,LOW);
  digitalWrite(g,LOW);
}
}
```

*Figura B1. 29*

6.21 Una vez concluido el programa, se procede a guardar el archivo en una carpeta para eso dirigirse a: ***Archivo →Salvar →"Seleccionar donde se guardará el archivo"***. Ahora se puede cargar el programa en el módulo Arduino Uno, para ello conectar el módulo Arduino a la PC con el cable USB-AB (Figura B1.30) y seleccionar la placa que se va a usar (Figura B1.31), y el puerto COM al que se encuentra conectado (Figura B1.32). Ya configurado lo anterior seleccionar el botón subir (Figura B1.33)

*Figura B1. 30 Cable USB-AB*

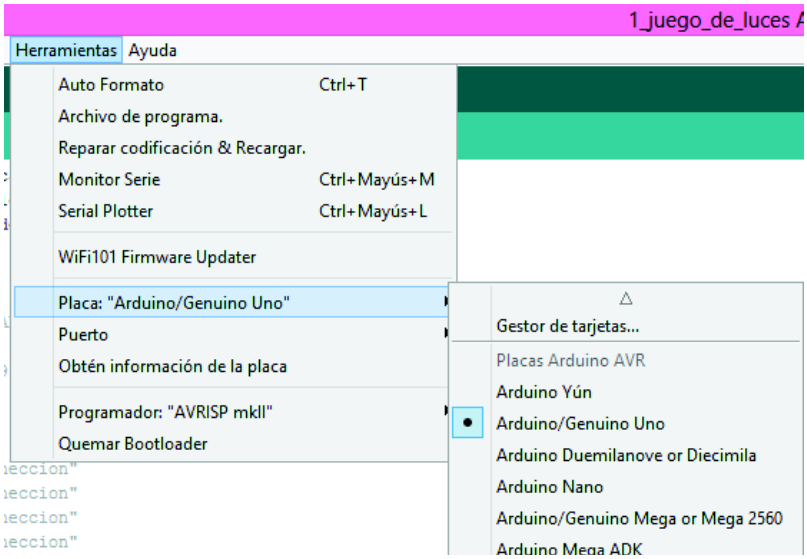



*Figura B1. 31 Selección de placa*

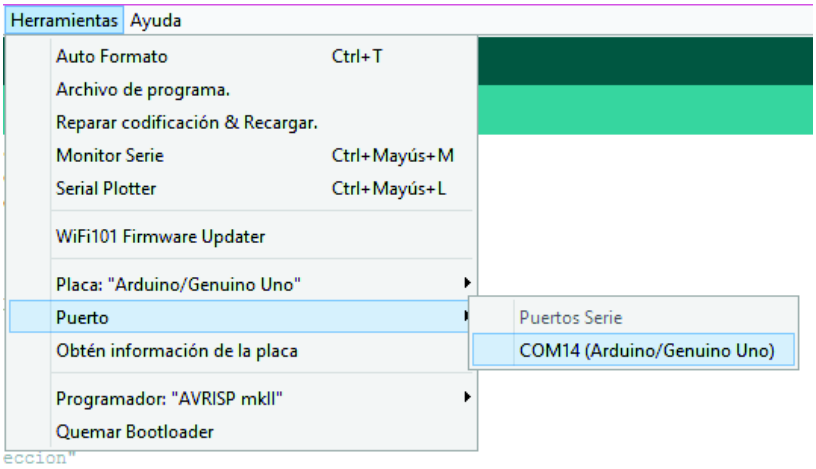



*Figura B1. 32 Selección de puerto*


*Figura B1. 33*

6.22 La Figura B1.34 muestra que se subió con éxito el programa

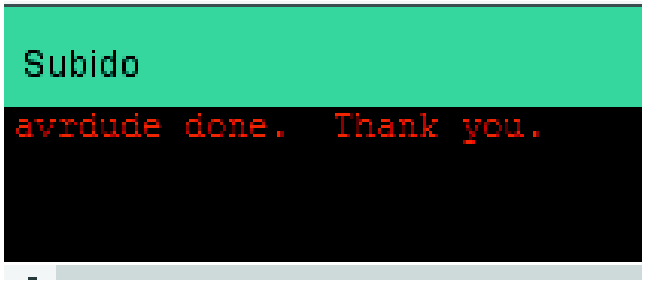



*Figura B1. 34 Confirmación de carga*

6.23 Se presiona el botón PWDISP7-1 si se utiliza el display DISP7-1 o se presiona el botón PWDISP7-2 si se utiliza el display DISP7-2

6.24 En las Figura B1.35 se observa el ejercicio en la placa didáctica

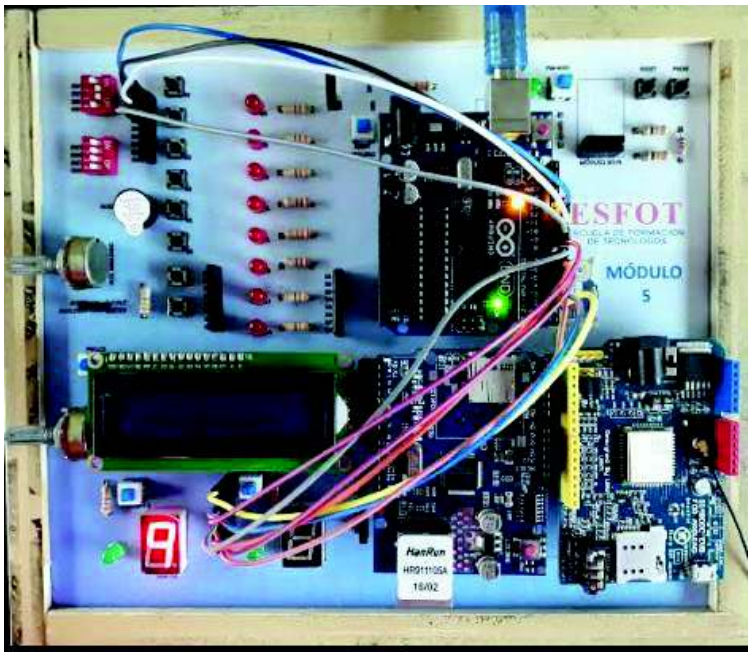



*Figura B1. 35 Resultado de conversor*

***NOTA 1**: Es necesario verificar que todos los botones (PWLCD, PWDISP7-1, PWDISP7-2, RESET ARDUINO y PW-WIFI) no se encuentren presionados.

***NOTA 2:** Una vez terminada la práctica es necesario que se retire todos los cables y se regresen todos sus componentes a sus posiciones originales

## 7. RECOMENDACIONES

- Es importante prestar mucha atención al declarar con nombre el número del pin, ya que puede haber confusión al momento de programar

- Al utilizar la librería de LCD, los pines indicados a utilizarse en ésta pueden ser modificados a gusto del usuario

- Hay que tomar en cuenta la naturaleza del pin a utilizarse, es decir, si se va a ejecutar el comando digitalRead() o digitalWrite () los pines deberán ser digitales. Lo mismo pasa con los analógicos.

- Cuando se necesite leer entradas digitales es necesario activar el Pull-UP de dichas entradas, para esto, se ingresa los siguientes códigos:

```
digitalMode(pin1,INPUT);          // Pin1 como entrada digital

digitalWrite(pin1, HIGH);         // Activación de la resistencia interna del pin
```

- Es importante que se declaren los dos anteriores códigos seguidos y en el ***void setup***

- Tener en cuenta que el Arduino posee solo un puerto de comunicación serial que es compartido con el USB y los pines 0 y 1, por lo que al momento de subir el sketch estos pines deben estar libres, y una vez subido el programa se los puede conectar y usar normalmente.

## 8. REFERENCIAS


[1] J. M. R. Gutiérrez, «Manual de Programación Arduino,» [En línea]. Available: https://arduinobot.pbworks.com/f/Manual+Programacion+Arduino.pdf.

**ANEXO B2**

**PRÁCTICA 2: MANEJO DEL MÓDULO *SHIELD ETHERNET* UTILIZANDO MÓDULOS ARDUINO.**

# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

## HOJA GUÍA PARA EL INSTRUCTOR

## PRÁCTICA 2

**1. TEMA:** Manejo del Módulo Shield Ethernet utilizando módulos Arduino.

### 2. OBJETIVOS

2.1. Relacionar al estudiante con la programación de módulos Arduino utilizando el IDE del mismo.

2.2. Implementar un circuito utilizando las placas didácticas de Arduino Uno para el manejo de módulos Shield.

2.3. Controlar el estado de los elementos pasivos como leds y buzzers a través de una interfaz gráfica

### 3. DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA

3.1. Elaborar un programa que permita controlar 2 leds, un buzzer y observar la variación de un potenciómetro desde una interfaz gráfica ubicada en un servidor web.

La página web debe contener: encabezado de bienvenida, opciones para el encendido y apagado de los leds, buzzer y se pueda observar la variación de la entrada analógica.

3.2. Armar el circuito necesario utilizando las placas didácticas, grabar el programa desarrollado y comprobar su funcionamiento

### 4. MATERIALES

- Placas didácticas Arduino Uno
- Cable de red (RJ45) Cat A o B, y si es directo a la pc, sin un switch cable cruzado.
- Computadora con el IDE de arduino instalado.
- Cable USB tipo A/B
- Cables para protoboard

### 5. DESARROLLO DE LA PRÁCTICA

**TIEMPO ESTIMADO DE LA PRÁCTICA:** 45 minutos – 1 hora

5.1 Abrir el programa: *"ARDUINO"*


*Figura B2. 1 Icono del software Arduino*

5.2 Se abrirá una ventana como se muestra a continuación

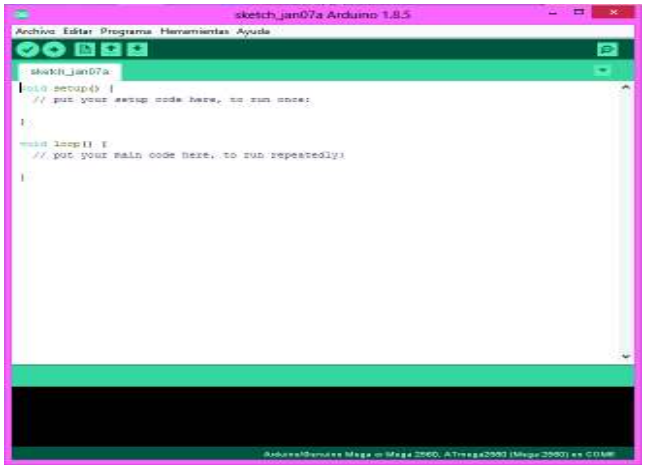



*Figura B2. 2 Ventana del software Arduino*

Se puede observar que ya se encuentra escrito las dos partes importantes del programa, las cuales son: el void setup y void loop

5.3 En el IDE de Arduino, dirigirse a ***Archivo*** →***Ethernet*** →***WebServer***. Como se muestra en la Figura B2.3

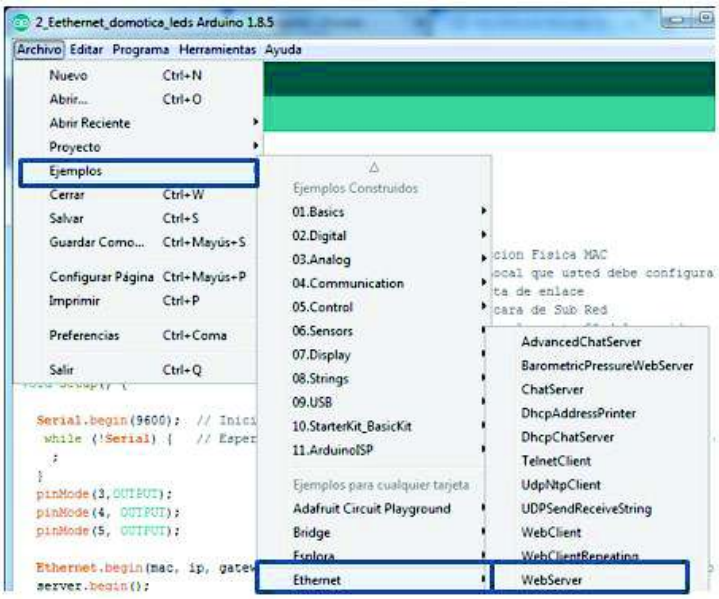



*Figura B2. 3 Ejemplos de códigos*

5.4 A continuación saldrá una nueva ventana como se indica en la Figura B2.4

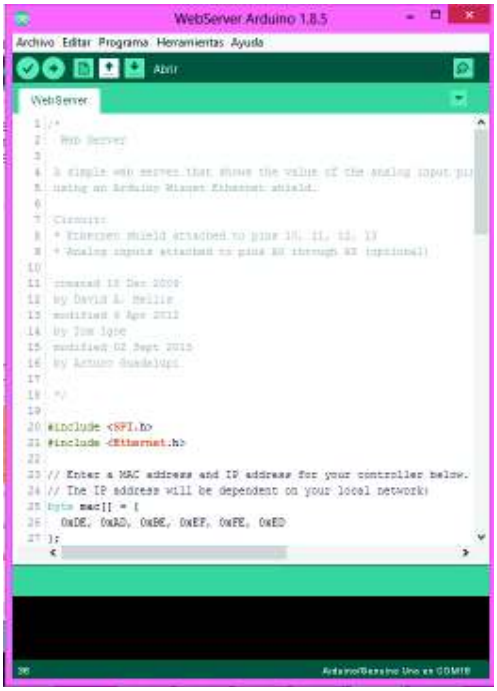



*Figura B2. 4*

5.5 De la nueva ventana que se muestra en la Figura B2.4 se pueden copiar y modificar los siguientes códigos, los cuales se muestran en la Figura B2.5

```
#include <SPI.h>
#include <Ethernet.h>
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };  //Direccion Fisica MAC
byte ip[] = { 192, 168, 0, 3 };                       // IP Local que usted debe configurar
byte gateway[] = { 192, 168, 0, 1 };                   // Puerta de enlace
byte subnet[] = { 255, 255, 255, 0 };                 //Mascara de Sub Red
EthernetServer server(80);             //El puerto 80 es el predeterminado para HTTP
String readString;
```

*Figura B2. 5*

5.6 En la función de ***void setup*** añadir la configuración de los pines que se utilizarán para los leds y buzzer, también se inicializa el puerto serial y la conexión Ethernet

```
void setup() {
   Serial.begin(9600);  // Inicializa el puerto serial
   while (!Serial) {   // Espera a que el puerto serial sea conectado
      }
  pinMode(3,OUTPUT);  // Pin designado para el buzzer
  pinMode(4, OUTPUT); // Pin designado para el buzzer
  pinMode(5, OUTPUT); // Pin designado para el buzzer

  Ethernet.begin(mac, ip, gateway, subnet); // Inicializa la conexion Ethernet y el servidor
  server.begin();
  Serial.print("El Servidor es: ");
  Serial.println(Ethernet.localIP());    // Imprime la direccion IP Local

}
```

*Figura B2. 6*

5.7 La siguiente parte corresponde a ***void loop*** , en la cual se puede ir modificando los comandos que se mostraron en la Figura B2.4. A continuación se muestra los comandos modificados

```
void loop() {
  // Crea una conexion Cliente
  EthernetClient client = server.available(); // Escucha los clientes entrantes
  if (client) {
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();

        if (readString.length() < 100) { //Lee caracter por caracter HTTP
            readString += c;            //Almacena los caracteres a un String
         }
         // si el requerimiento HTTP fue finalizado
         if (c == '\n') {
           Serial.println(readString); //Imprime en el monitor serial
          client.println("HTTP/1.1 200 OK");           //Envia un encabezado de respuesta http
          client.println("Content-Type: text/html");
          client.println("Refresh: 1"); // Actualiza la página automáticamente cada 1 segundo
          client.println();
```

*Figura B2. 7*

Esta parte del programa ayuda a la creación de la página web con códigos HTML

***Nota 1:** En esta parte poner especial atención porque en código HTML no existe compilador de sintaxis, sino simplemente no aparecerá nada en la página web o saldrá incompleto.

5.8 Para la creación del encabezado de la pestaña del navegador Web colocar:

```
          client.println("<HTML>");
          client.println("<HEAD>"); //Comando HTML para inicio de encabezado
          client.println("<TITLE>Ethernet Arduino</TITLE>"); //Se inserta el titulo del encabezado
          client.println("</HEAD>"); //Comando para cerrar el encabezado
          client.println("<BODY>"); //Comando para iniciar el cuerpo de la página
          client.println("<hr />"); //Salto de párrafo
          client.println("<TITLE>Practica 2 Ethernet</TITLE>"); //Titulo del cuerpo
          client.println("</HEAD>");
```

*Figura B2. 8*

Como se puede observar el comando ***"<TITLE>...../TITLE>"*** permite introducir un título, el cual aparecerá en la página web.

Se puede seguir introduciendo códigos que permitan crear un encabezado de la siguiente manera:

```
          client.println("<body style=background-color:#11E7CF;>"); //Estilo de letra y color
          client.println("</body>");
          client.println("<BODY>");
          client.println("<H1>ESCUELA DE FORMACION DE TECNOLOGOS</H1>"); //texto que solo ocupa una sola línea
          client.println("<H2>MONITORES</H2>");
          client.println("<br />");
```

*Figura B2. 9*

5.9 A continuación se crea las opciones (hipervínculos) que permitirán encender y apagar los leds

```
client.println("<a href=\"/?button2on\"\"> Encender Pin2</a> ");// El hipervínculo "button2on" se relaciona con que el pin2 tenga salida de 5v
client.println(" | | | ");
client.println("<a href=\"/?button2off\"\"> Apagar Pin2</a><br /> ");// El hipervínculo "button2off" se relaciona con que el pin2 tenga salida de 0v
client.println("<br />");
```

*Figura B2. 10*

Los mismos comandos se pueden replicar para los hipervínculos que sean necesarios, en este caso será necesario repetir una vez más, con la diferencia que se cambia el número del hipervínculo (button) y número de Pin

```
59          client.println("<br />");
60          client.println("<a href=\"/?button3on\"\"> Encender Pin3</a> ");
61          client.println(" | | | ");
62          client.println("<a href=\"/?button3off\"\"> Apagar Pin3</a><br /> ");
63          client.println("<br />");
64          client.println("<br />");
65          client.println("<a href=\"/?button4on\"\"> Encender Buzzer</a> ");
66          client.println(" | | | ");
67          client.println("<a href=\"/?button4off\"\"> Apagar Buzzer</a><br /> ");
68          client.println("<br />");
```

*Figura B2. 11*

En la Figura B2.11 se puede observar el hipervínculo “button4on” el cual se utilizará para encender un buzzer

5.10 En la Figura B2.12 se configura la entrada analógica cuya variación se mostrará en la página web

```
69          int lecturaSensor = analogRead(0);   // Lee el puerto analógico A0
70          client.print("Entrada Analoga ");
71          client.print(0);
72          client.print(" es ");
73          client.print(lecturaSensor); //Imprime la lectura de del puerto analógico A0
74          client.println("<br />");
75          client.println("<hr />");
76          client.println("</BODY>");
77          client.println("</HTML>");
78           delay(1);
79          client.stop();//detiene el cliente servidor
```

*Figura B2. 12*

5.11 A continuación se describe lo que cada hipervínculo “button” de la página web hará en el módulo didáctico.

```
80        //control del arduino si un boton es presionado
81         if (readString.indexOf("?button2on") >0){ //Lee el estado del hipervínculo, si el button2on fue-
82             digitalWrite(4, HIGH);              //seleccionado el pin4 del Arduino tendra una salida de 5v
83         }
84         if (readString.indexOf("?button2off") >0){//Lee el estado del hipervínculo, si el button2off fue-
85             digitalWrite(4, LOW);               //seleccionado el pin4 del Arduino tendra una salida de 0v
86         }
87
88         if (readString.indexOf("?button3on") >0){
89             digitalWrite(5, HIGH);
90         }
91         if (readString.indexOf("?button3off") >0){
92             digitalWrite(5, LOW);
93         }
```

*Figura B2. 13*

El procedimiento se repite para el hipervínculo “button3”, como se observa en la Figura B2.13

5.12 Una vez ya programados los hipervínculos para los leds se pasa al buzzer

```
            // Para el buzzer
            if (readString.indexOf("?button4on") >0){
                TCCR2B =3; // Selecciona el TIMER 2 a una frecuencia de 980.39Hz
                analogWrite(3, 200); // El pin3 tiene una salida PWM
                delay(1000);
                TCCR1B =2; // TIMER 1
                TCCR2B =2; // TIMER 2
                analogWrite(3, 200);
                delay(1000);
                TCCR1B =1; // TIMER 1
                TCCR2B =1; // TIMER 2
                analogWrite(3, 200);
                delay(1000);
            }
            if (readString.indexOf("?button4off") >0){
                analogWrite(3, 0); // El pin3 tiene como salida 0v
                delay(500);
            }
            readString="";
          }
        }
    }
}}
```

*Figura B2. 14*

5.13 Una vez concluido el programa, se procede a guardar el archivo en una carpeta para eso dirigirse a: ***Archivo →Salvar →"Seleccionar donde se guardará el archivo"***. Ahora se puede cargar el programa en el módulo Arduino Uno, para ello conectar el módulo Arduino a la PC con el cable USB-AB (Figura B2.15) y seleccionar la placa que se va a usar (Figura B2.16), y el puerto COM al que se encuentra conectado (Figura B2.17). Ya configurado lo anterior seleccionar el botón subir (Figura B2.18)

*Figura B2. 15 Cable USB-AB*

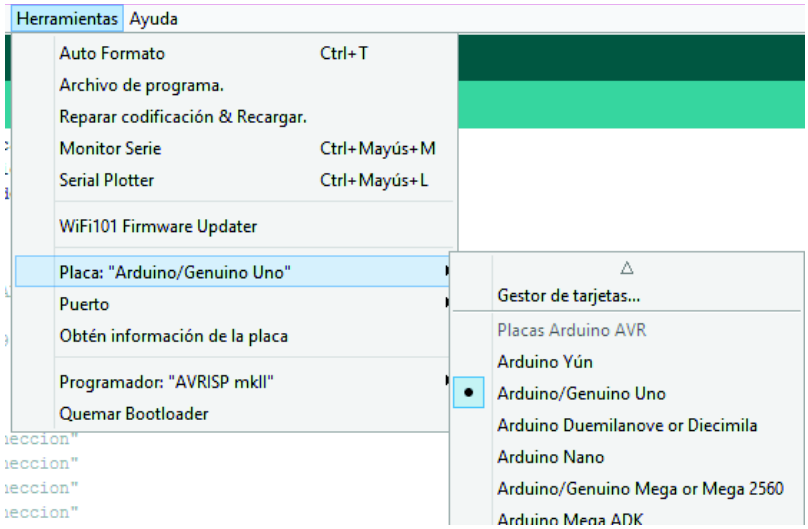



*Figura B2. 16 Selección de placa*

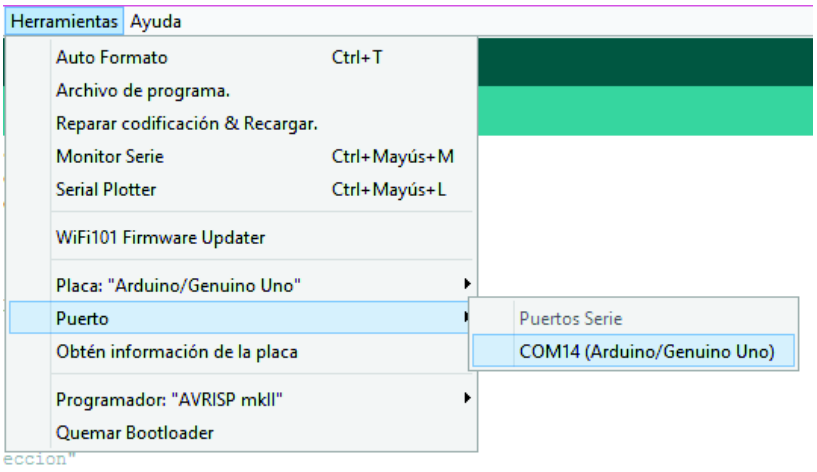



*Figura B2. 17 Selección de puerto*


*Figura B2. 18*

5.14 La Figura B2.19 muestra que se subió con éxito el programa

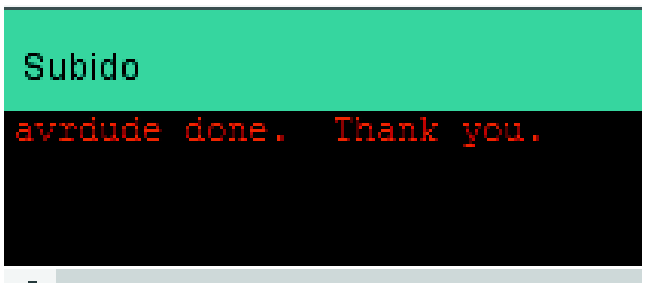



*Figura B2. 19 Confirmación de carga*

5.15 Una vez grabado el programa en el Arduino, Se debe asignar una dirección IP a la PC, la cual debe pertenecer a la misma red que se encuentra el módulo Ethernet. Para asignar la dirección IP en la PC dirigirse a: ***Panel de control →Centro de redes y recursos compartidos*** (Figura B2.20)

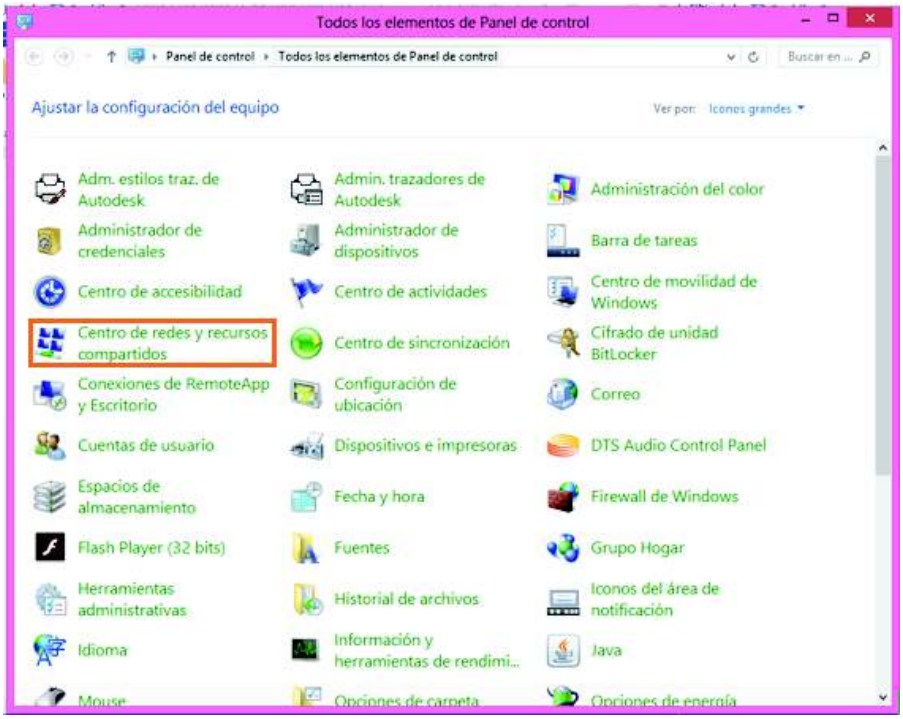



*Figura B2. 20 Panel de control*

5.16 Seleccionar ***Cambiar configuración del adaptador***

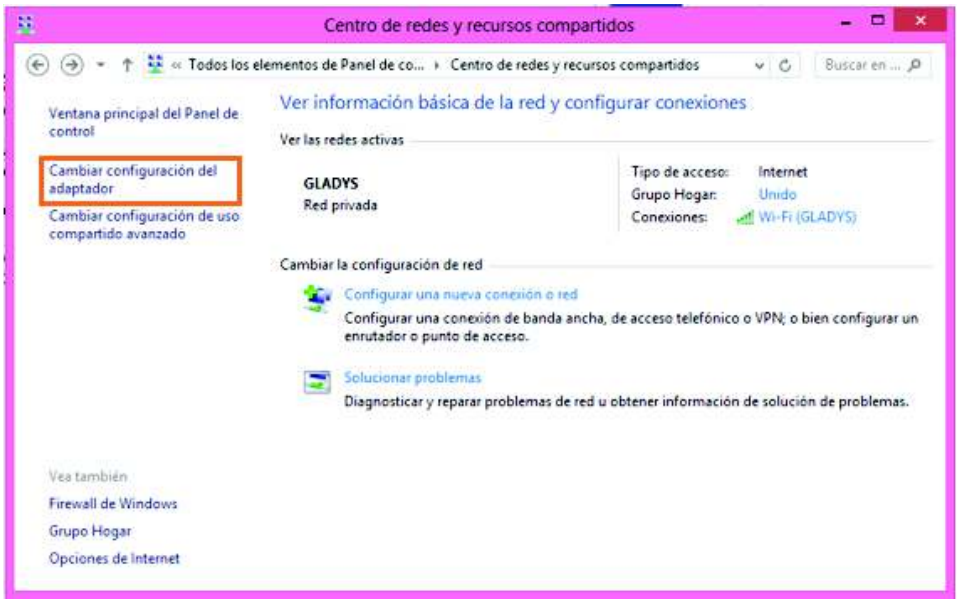



*Figura B2. 21 Centro de redes y recursos compartidos*

5.17 Seleccionar ***Ethernet***

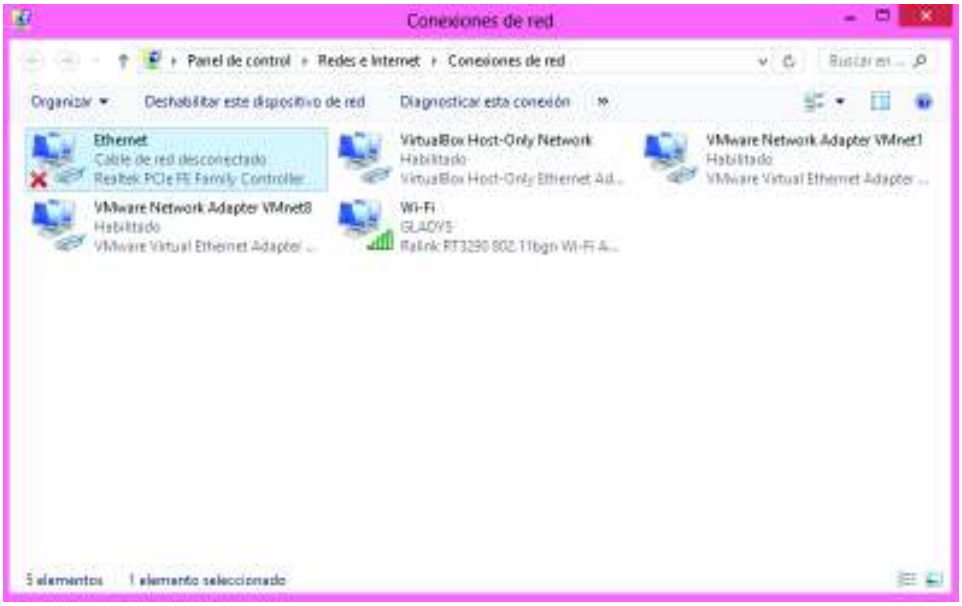



*Figura B2. 22 Conexiónes de red*

Se puede observar que en Ethernet señala: ***Cable de red desconectado.*** Para habilitarlo realizar el siguiente procedimiento:

5.18 Realizar la conexión siguiente:

Se coloca el módulo Shield Ethernet sobre el Arduino Uno como se muestra a continuación

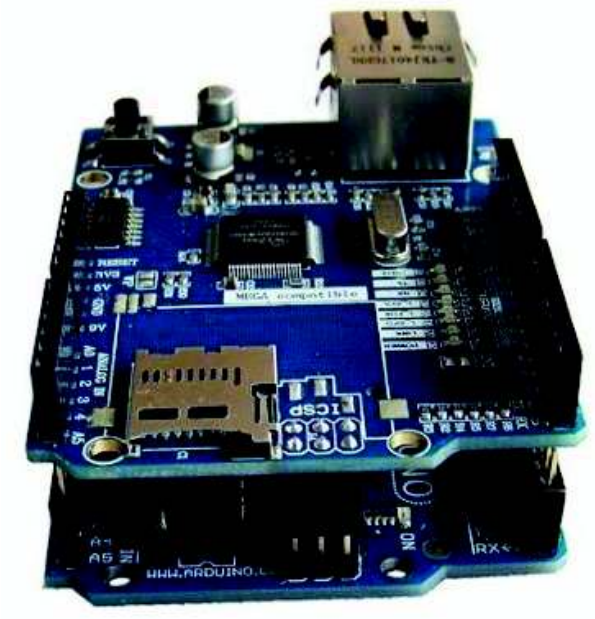


*Figura B2. 23 Montaje Shield*

Se puede conectar el módulo Ethernet a un switch (Figura B2.24) que pertenece a la misma red de la máquina a utilizarse, como se muestra a continuación

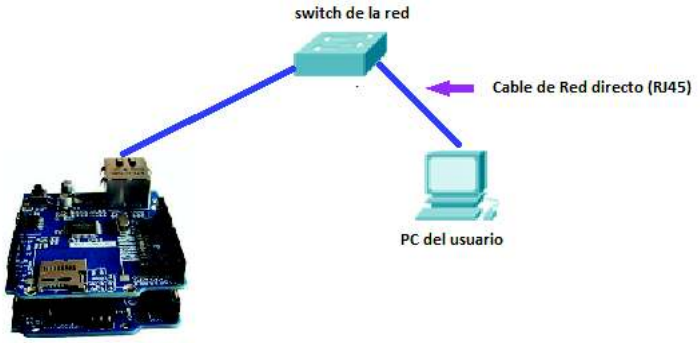



*Figura B2. 24 Conexión con switch*

Si no se cuenta con un Switch o algún otro equipo de red se puede realizar la siguiente conexión (Figura B2.25):

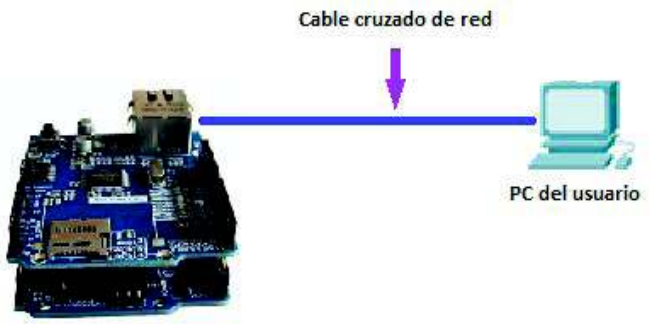



*Figura B2. 25 conexión directa*

5.19 Una vez realizada la conexión deberá habilitarse, como se muestra en la Figura B2.26

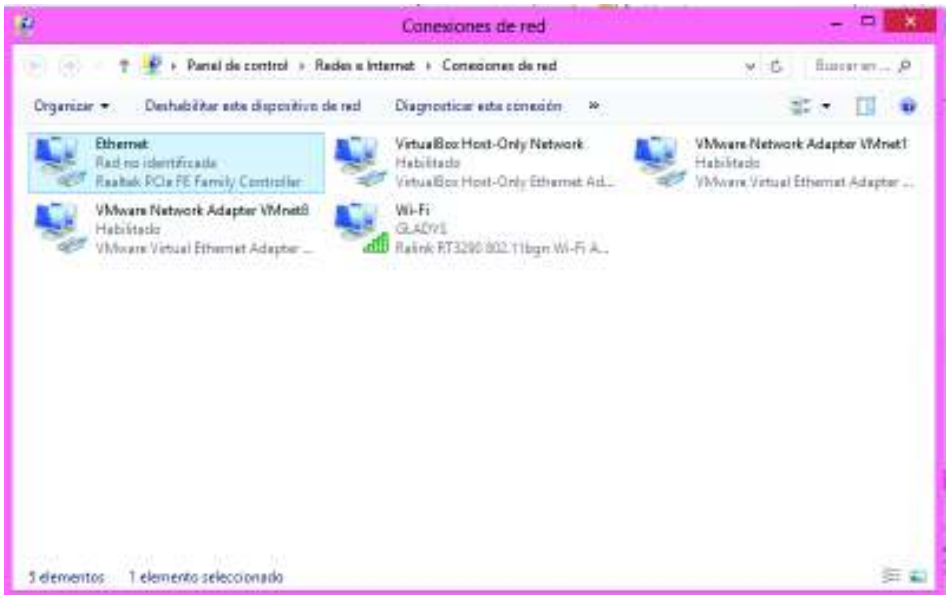



*Figura B2. 26 Conexiones de red*

5.20 En Ethernet dar click derecho y se selecciona ***Propiedades*** y deberá aparecer una ventana como se muestra en la Figura B2.27

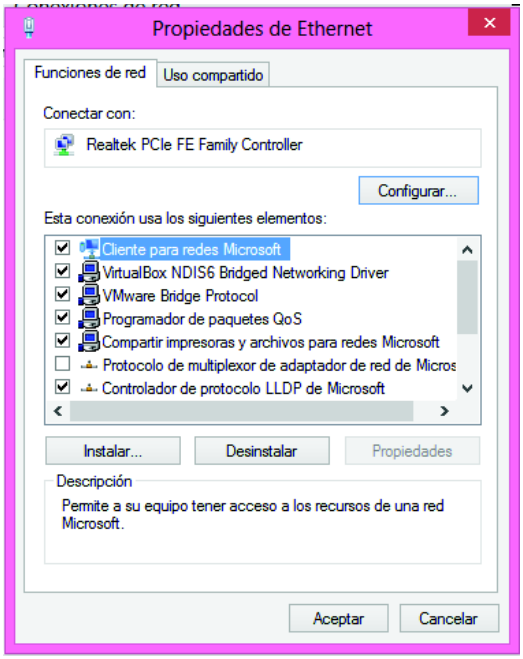



*Figura B2. 27 Propiedades de Ethernet*

5.21 Se busca la opción ***Protocolo de Internet versión 4 (TCP/IPv4)*** seleccionar y dar click en propiedades

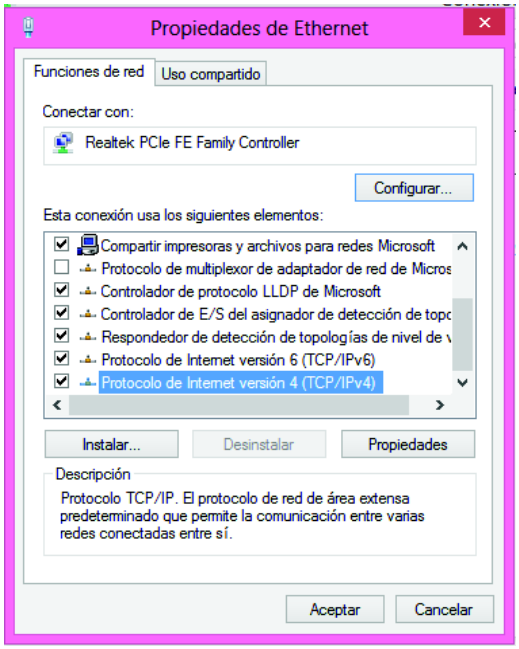



*Figura B2. 28 Protocolo IPv4*

5.22 En la ventana que aparece (Figura B2.29), seleccionar: ***Usar la siguiente dirección IP*** y colocar la siguiente dirección IP, Mascara de subred y puerta de enlace predeterminada; y seleccionar: ***Validar configuración*** al salir.

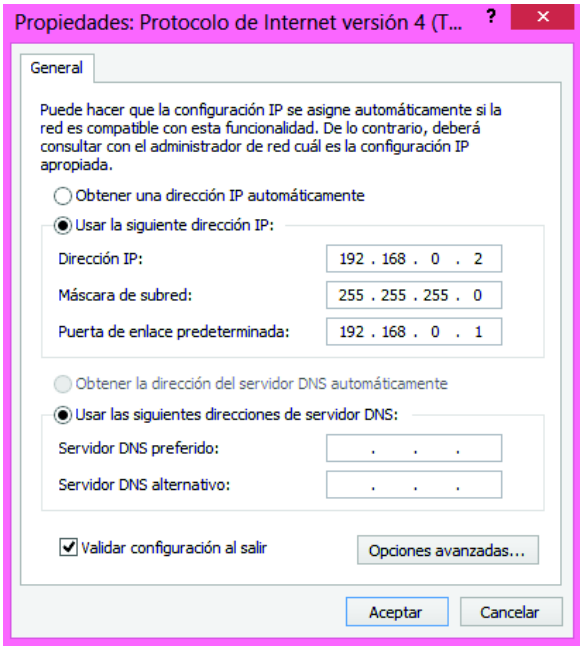



*Figura B2. 29 Configuración de red*

## 6. ENSAMBLAJE DEL CIRCUITO

***Nota 2:** Se recomienda desconectar el Arduino uno antes de realizar las conexiones

6.1. Se realiza las siguientes conexiones con los cables de protoboard proporcionados.

- El pin A0 del Arduino a INPUT ANALOGIC de la Placa Didáctica.

- El pin D3 del Arduino a BUZZER OUTPUT de la Placa Didáctica. (D3 = pin 3 digital del Arduino)

- El pin D4 del Arduino a D1 de la Placa Didáctica.

- El pin D5 del Arduino a D2 de la Placa Didáctica.

## 7. PROGRAMA EN FUNCIONAMIENTO

7.1 Dirigirse al IDE de Arduino y seleccionar: ***Monitor Serie*** en 9600 baudios.


*Figura B2. 30*

7.2 Aparecerá una pantalla como la siguiente:

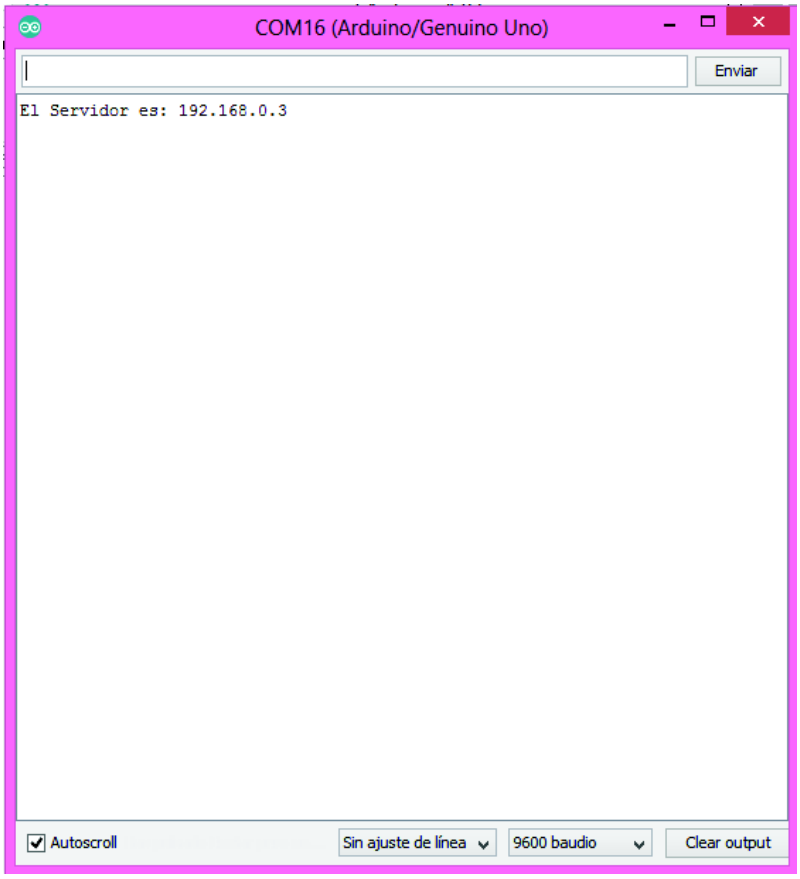



*Figura B2. 31*

El monitor serial indica la dirección IP del servidor en este caso es: ***192.168.0.3***

7.3 Abrir un navegador web y colocar la dirección IP que se muestra en la Figura B2.31

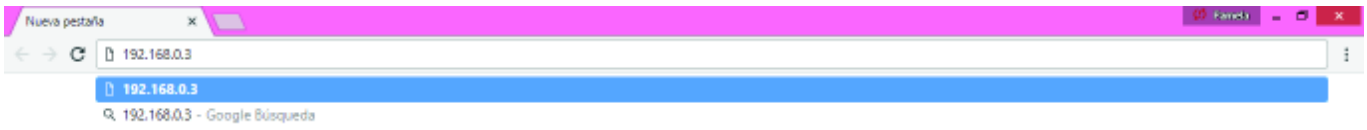



*Figura B2. 32*

7.4 A continuación se podrá visualizar la página web (Figura B2.33) que se programó en el IDE de Arduino, la cual se encuentra guardada en el módulo Ethernet

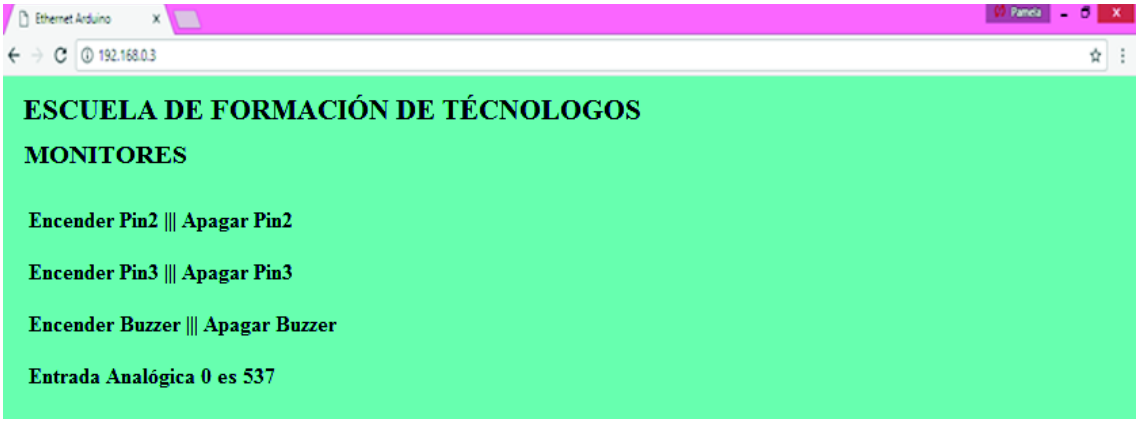



*Figura B2. 33*

7.5 Seleccionar el Hipervínculo ***"Encender Pin2"*** (Figura B2.34) se puede observar que se encendió el diodo led que pertenece al pin 2 (Figura B2.35)


*Figura B2. 34*

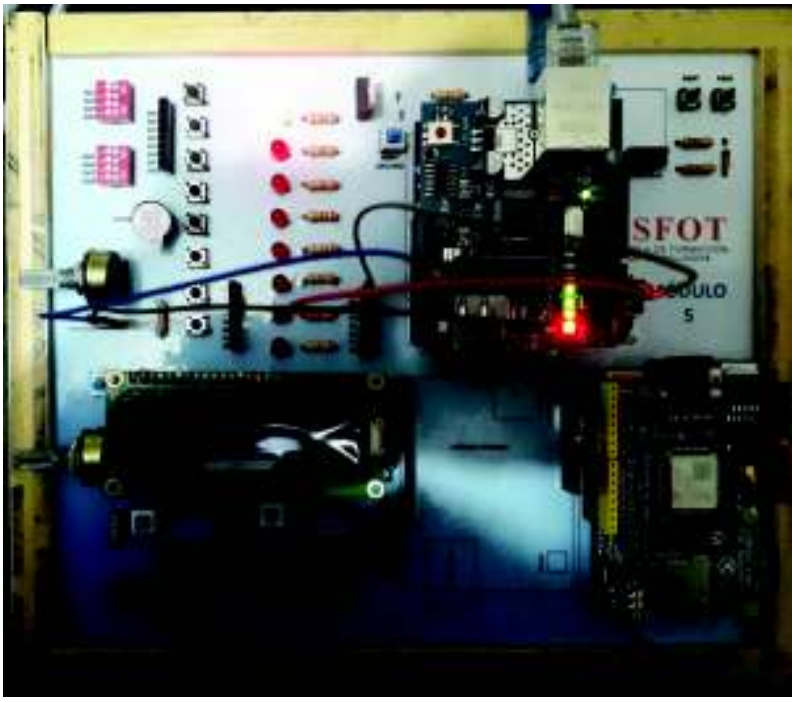



*Figura B2. 35*

***NOTA 3**: Es necesario verificar que todos los botones (PWLCD, PWDISP7-1, PWDISP7-2, RESET ARDUINO y PW-WIFI) no se encuentren presionados.

***NOTA 4:** Una vez terminada la práctica es necesario que se retire todos los cables y se regresen todos sus componentes a sus posiciones originales

## 8. RECOMENDACIONES

- Es recomendable que se tenga ciertos conocimientos básicos de códigos HTML para la creación de la página Web.

- Es recomendable que si se tiene varias máquinas en una red y si se desea realizar varias páginas web utilizando varios módulos Ethernet, es importante que cada uno tenga una dirección IP estática; esto evitara conflictos en la red al momento de realizar las prácticas.

- Si se utiliza la conexión directa entre el módulo Ethernet y la PC es necesario que el cable sea cruzado es decir que un lado posea el estándar 568-A y en el otro lado del cable sea el estándar 568-B.

- Es necesario saber los TIMER que el Arduino utiliza, cuáles son sus frecuencias de operación y sobretodo cuales son los pins correspondiente a estos.

## 9. REFERENCIAS


[1] J. M. R. Gutiérrez, «Arduino + Ethernet Shield,» [En línea]. Available: http://unicarlos.com/_ARDUINO/Arduino%20+%20Ethernet%20Shield%20(1).pdf.

[2] ARDUINO, «Ethernet / Ethernet 2 library,» [En línea]. Available: https://www.arduino.cc/en/Reference/Ethernet.

[3] A. Rubio, «10 códigos html para páginas web y para que sirven,» 19 07 2017. [En línea]. Available: https://pe.godaddy.com/blog/10-codigos-html-para-paginas-web-y-para-que-sirven/.

# ANEXO B3
# PRÁCTICA 3: MANEJO DE MÓDULO WIFI.

# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

## HOJA GUÍA PARA EL INSTRUCTOR

## PRÁCTICA 3

**1. TEMA:** Manejo del Módulo WiFi.

**2. OBJETIVOS:**

2.1. Relacionar al estudiante con la programación que proporciona el IDE de Arduino.

2.2. Implementar un circuito utilizando las placas didácticas de Arduino Uno para el manejo de un módulo WiFi.

2.3. Controlar el estado de los elementos pasivos como leds a través de una interfaz gráfica proporcionada por una página web

**3. DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA**

3.1. Elaborar un programa que permita controlar 2 leds desde una interfaz gráfica ubicada en un servidor web.

La página web debe contener: encabezado de bienvenida y opciones para el encendido y apagado de los leds.

3.2. Armar el circuito necesario utilizando las placas didácticas, grabar el programa desarrollado y comprobar su funcionamiento

**4. MATERIALES**

- Placa didáctica Arduino Uno
- Access Point (también puede usarse un celular que funcione como AP *"conexión compartida"*)
- Computadora con IDE de arduino instalado
- Cable USB tupo A/B
- Cables de protoboard

**5. DESARROLLO DE LA PRÁCTICA**

**TIEMPO ESTIMADO DE LA PRÁCTICA:** 45 minutos – 1 hora

5.1 Para comenzar la práctica es necesario tener un punto de acceso a una red inalámbrica, puede ser un Access point o la conexión compartida de un celular; en este caso se utilizará esta última. Para ello dirigirse a, conexión compartida en el celular que se utilizará (Figura B3.1)


*Figura B3. 1*

5.2 Al momento de seleccionar la conexión compartida se mostrarán las credenciales como: ***Nombre de la red*** y ***Contraseña*** (Figura B3.2).

Tanto el nombre de la red ***"AndroidAP"*** como su contraseña ***"practica wifi"***, servirán para la configuración del módulo WiFi


*Figura B3. 2*

5.3 Abrir el programa: *"ARDUINO"*

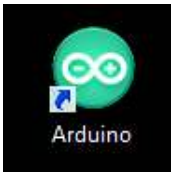



*Figura B3. 3 Icono del software Arduino*

5.4 Se abrirá una ventana como se muestra a continuación

*Figura B3. 4 Ventana del software Arduino*

Se puede observar que ya se encuentra escrito las dos partes importantes del programa, las cuales son: el void setup y void loop

5.5 Incluir las siguientes librerías para que el módulo WiFi pueda funcionar correctamente

```
1 #include <ESP8266WiFi.h>
2 #include <WiFiClient.h>
3 #include <ESP8266WebServer.h>
4 #include <ESP8266mDNS.h>
5 MDNSResponder mdns;
```

*Figura B3. 5*

5.6 Colocar las credenciales mencionadas en la Figura B3.2 como se muestra en la Figura B3.6

```
6  // Reemplazar por las credenciales a usar
7  const char* ssid = "AndroidAP"; // Nombre de la red Inalámbrica
8  const char* password = "practica wifi"; // Clave de la red Inalámbrica
9  ESP8266WebServer server(80); // Puerto de HTTP
10 String webPage = "";
```

*Figura B3. 6*

5.7 Especificar los pines a usar

```
11 int gpio0_pin = 0; // el Pin0 del módulo WiFi tiene de nombre gpio0_pin
12 int gpio2_pin = 2; // el Pin2 del módulo WiFi tiene de nombre gpio2_pin
```

*Figura B3. 7*

5.8 En el void setup configurar el encabezado y los hipervínculos a utilizarse (para mayor nitidez dirigirse a Anexo de esta práctica)

*Figura B3. 8*

5.9 Declarar el modo de uso de los pines antes señalados, colocar una taza de transferencia de 115200 y llamar el ssid y el password como se muestra en la Figura B3.9

```
// preparing GPIOs
  pinMode(gpio0_pin, OUTPUT); // gpio0_pin como salida
  digitalWrite(gpio0_pin, LOW); //Se escribe en gpin0 0v
  pinMode(gpio2_pin, OUTPUT); // gpio2_pin como salida
  digitalWrite(gpio2_pin, LOW); //Se escribe en gpin2 0v
  delay(1000);
  Serial.begin(115200); // Taza de transferencia de datos
  WiFi.begin(ssid, password); //Se llama al nombre y contraseña de la red inalambrica
  Serial.println("");
```

*Figura B3. 9*

5.10 Colocar los siguientes comandos (Figura B3.10) los cuales permitirán la conexión con el módulo

```
    // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

*Figura B3. 10*

5.11 Los comandos siguientes permitirán visualizar, en un monitor serial la dirección IP perteneciente al módulo WiFi

```
  Serial.println("");
  Serial.print("Connected to ");// Se escribe en la pantalla del terminal virtual
  Serial.println(ssid);         // Aparece el nombre de la red inlámbrica a la cual se esta conectando
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  if (mdns.begin("esp8266", WiFi.localIP())) {
    Serial.println("MDNS responder started");
  }
  server.on("/", [](){
    server.send(200, "text/html", webPage);
  });
```

*Figura B3. 11*

5.12 En los siguientes códigos se configura el hipervínculo On y OFF para el encendido y apagado del pin ***GPIO0***

```
server.on("/socket1On", [](){
    server.send(200, "text/html", webPage); // En la pagina web aparecerá la opción
    digitalWrite(gpio0_pin, HIGH);         // de encender el pin gpino0_pin
    delay(1000);
  });
  server.on("/socket1Off", [](){
    server.send(200, "text/html", webPage);// En la pagina web aparecerá la opción
    digitalWrite(gpio0_pin, LOW);         // de apagar el pin gpino0_pin
    delay(1000);
  });
```

*Figura B3. 12*

5.13 También se configura los códigos para el hipervínculo On y OFF para el encendido y apagado del pin ***GPIO2***

```
  server.on("/socket2On", [](){
    server.send(200, "text/html", webPage);
    digitalWrite(gpio2_pin, HIGH);
    delay(1000);
  });
  server.on("/socket2Off", [](){
    server.send(200, "text/html", webPage);
    digitalWrite(gpio2_pin, LOW);
    delay(1000);
  });
  server.begin();
  Serial.println("HTTP server started");
}
```

*Figura B3. 13*

5.14 Para el ***void loop*** , únicamente se digita el código:

```
void loop(void){
  server.handleClient();
}
```

*Figura B3. 14*

5.15 Una vez concluido el programa, se procede a guardar el archivo en una carpeta, para eso dirigirse a: ***Archivo →Salvar →"Seleccionar donde se guardará el archivo"***. Conectar el módulo Arduino a la PC con el cable USB-AB (Figura B3.15)

*Figura B3. 15 Cable USB-AB*

5.16 Presionar el botón "***Reset Arduino"*** de la placa didáctica

5.17 Mantener presionado el pulsador "***PROG"*** del módulo WiFi, mientras se presiona el botón de encendido del módulo WiFi ***"PW-WIFI"***, y seguir presionando el pulsador "***PROG"*** del módulo WiFi por 5 segundos más.

5.18 Dirigirse nuevamente al sketch de Arduino y seleccionar ***Herramientas →Placa →Generic ESP8266 Module"***, como se muestra en la Figura B3.16.

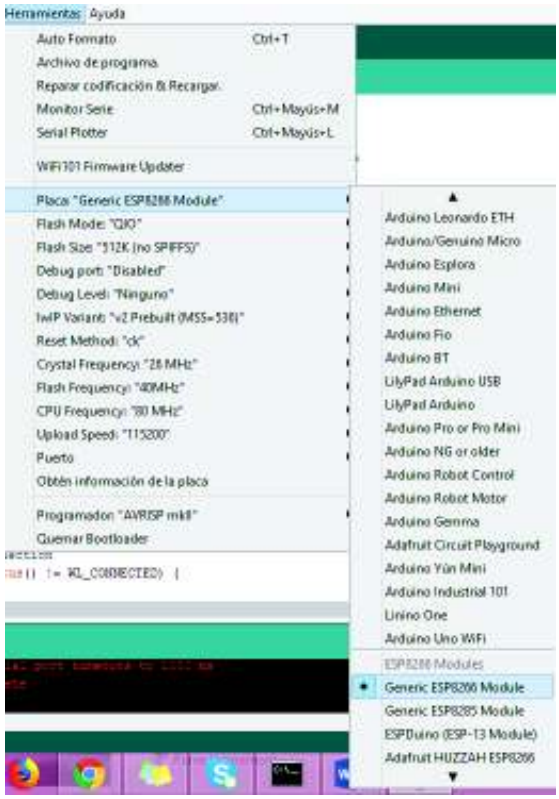



*Figura B3. 16 Selección de placa*

*** Nota 1**: Antes de seleccionar el módulo ESP es sumamente importante que el campo del puerto se encuentre vacío como se muestra en la Figura B3.17


*Figura B3. 17Selección de puerto*

5.19 Seleccionar el puerto ***Herramientas →Puerto→COM # (Arduino/Genuino Uno)***

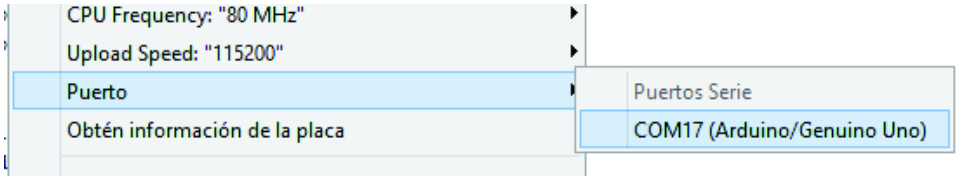



*Figura B3. 18 Selección de puerto*

5.20 Seleccionar el botón Subir (Figura B3.19)


*Figura B3. 19 Botón subir*

5.21 En la figura B3.20 se indica que el programa se subió correctamente

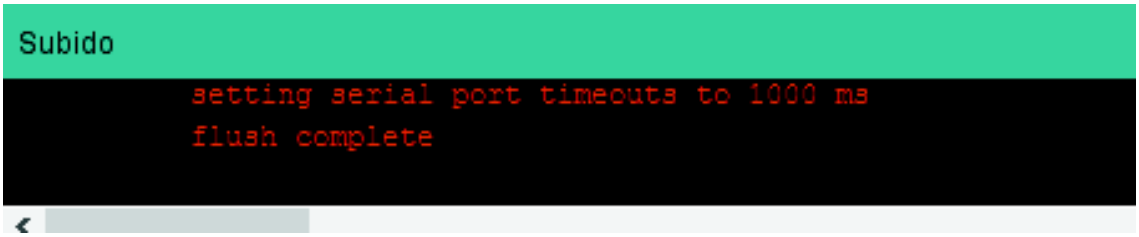



*Figura B3. 20 Confirmación de carga*

5.22 A continuación se muestra la distribución de pines del módulo WiFi

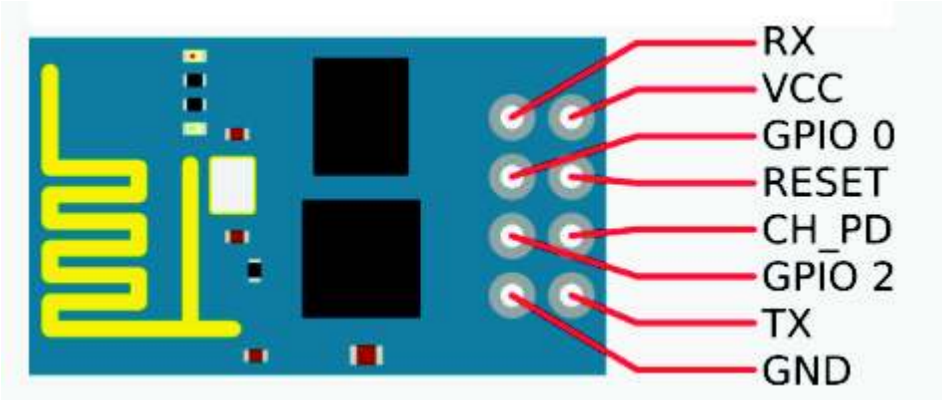



*Figura B3. 21 Distribución de pines ESP8266*

5.23 Conectar:

- GPIO_0 del módulo WiFi al D1 de la placa didáctica
- GPIO_2 del módulo WiFi al D2 de la placa didáctica

5.24 Para acceder al módulo WiFi desde la PC; la PC también debe estar conectada a la red inalámbrica que se indicó en el punto 5.2


*Figura B3. 22 Conexión WiFi de celular*

5.25 Una vez que la PC se haya enganchado a la red abrir el terminal serial del IDE de Arduino

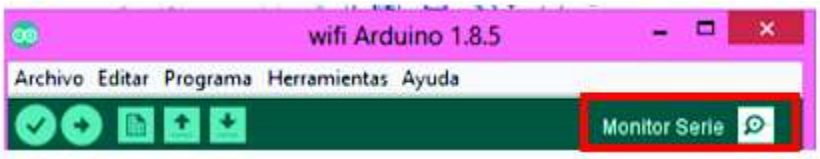



*Figura B3. 23 Botón de monitor serial*

5.26 A parecerá una ventana como la siguiente:

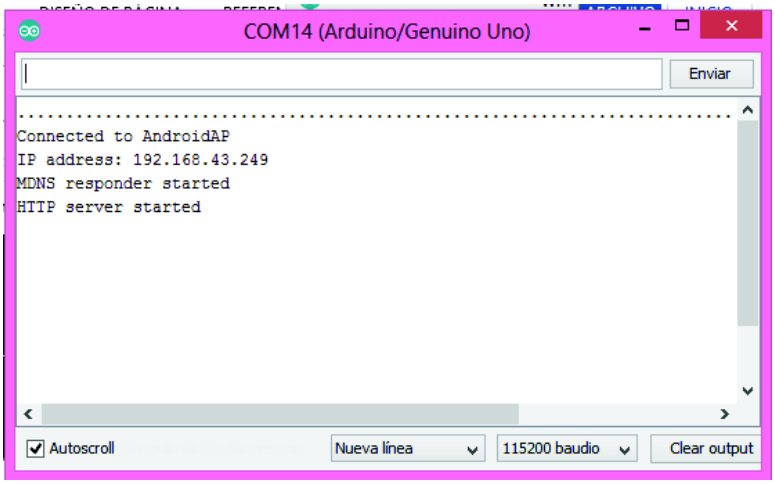



*Figura B3. 24 Conexión serial*

***NOTA 2:** Se debe tomar en cuenta que la ventana terminal debe estar en Nueva Línea y 115200 baudios

***NOTA 3:** Si no se puede visualizar el mensaje en la ventana se puede apagar y encender el módulo WiFi

5.27 Abrir un servidor WEB y digitar la dirección IP que se muestra en la ventana serial del Arduino, en este caso como se muestra en la Figura B3.25 digitar la dirección 192.168.43.249


*Figura B3. 25 Página web*

5.28 Al presionar ON del Socket 1 y 2 se encenderán los diodos led D1 Y D2 de la placa didáctica, de igual manera, si se presiona OFF del Socket 1 y 2 se apagarán los diodos led D1 Y D2

## 6. RECOMENDACIONES

- Se recomienda que para las prácticas de laboratorio se tenga a disposición un AP para realizar las pruebas

- Es importante recordar la dirección IP que aparece en la ventana del serial para que poder ingresar a la página web

## 7. REFERENCIAS

[1] PRPMETEC, «ARDUINO Y WIFI ESP8266,» [En línea]. Available: https://www.prometec.net/arduino-wifi/.

[2] E. S. I. Team, «ESP8266EX Datasheet,» [En línea]. Available: https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf.

[3] R. N. TUTORIALS, «How to Install the ESP8266 Board in Arduino IDE,» [En línea]. Available: http://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/.

## 8. ANEXO

```
void setup(void){

  webPage += "<h1>Escuela de Formacion de Tecnologos</h1><h2>ESP8266 Web Server</h2><p>Socket #1 <a href=\"socket1On\"><button>ON</button></a> <a href=\"socket1Off\"><button>OFF</button></a></p>";

  webPage += "<p>Socket #2 <a href=\"socket2On\"><button>ON</button></a> <a href=\"socket2Off\"><button>OFF</button></a></p>";
```

# ANEXO B4

# PRÁCTICA 4: MANEJO DEL MÓDULO *SHIELD* GSM SIM800C.

# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

## HOJA GUÍA PARA EL INSTRUCTOR

## PRÁCTICA 4

**1. TEMA:** Manejo del Módulo Shield GSM SIM800C.

**2. OBJETIVOS:**

2.1. Relacionar al estudiante con la programación que proporciona el IDE de Arduino.

2.2. Implementar un circuito utilizando las placas didácticas de Arduino Uno para el manejo de un módulo Shield GSM SIM800C.

2.3. Enviar mensajes de texto usando el Módulo Shield GSM SIM800C.

**3. DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA**

3.1. Elaborar un programa que permita enviar 3 SMS al presionar un pulsador 1,un pulsador 2 y un pulsador 3 respectivamente y, que realice una llamada de voz al encender un dipswitch a cualquier número celular.

**4. MATERIALES**

- Placa didáctica Arduino Uno
- MicroChip de cualquier operadora de telefonía celular
- Cable tipo A/B
- Cables para protoboard

**5. DESARROLLO DE LA PRÁCTICA.**

**TIEMPO ESTIMADO DE LA PRÁCTICA:** 45 minutos – 1 hora

5.1 Abrir el programa: *"ARDUINO"*

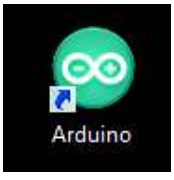



*Figura B4. 1 Icono del software Arduino*

5.2 Se abrirá una ventana como se muestra a continuación


*Figura B4. 2 Ventana del software Arduino*

Se puede observar que ya se encuentra escrito las dos partes importantes del programa, las cuales son: el void setup y void loop

5.3 Para iniciar el programa escribir al inicio, fuera del ***void setup()***, las librerías que se muestran en la Figura B4.3.

```
1 //Se inicializan las librerias para usar un modulo GSM-GPRS generico
2 #include <SoftwareSerial.h>
3 #include <String.h>
```

*Figura B4. 3*

Estas librerías ayudarán al correcto funcionamiento del módulo GSM.

5.4 Se introduce los comandos necesarios para la comunicación serial y se declara las variables que se utilizarán para los pines designados a los pulsadores

```
4 //Se crea una comunicacion serial emulada en los pines 7 y 8
5 SoftwareSerial mySerial(7,8);
6 //Constantes que contienen el puertos a usar
7 const int pul_sms_1 = 3 ,pul_sms_2 = 4 ,pul_sms_3 = 5, dsw_call = 6;
8 int lectura1,lectura2,lectura3,lectura4;
```

*Figura B4. 4*

5.5 En el ***void setup ()*** se declara la comunicación serial y el modo de los pines que se utilizarán

```
9  void setup()
10 {
11   //Se utiliza una comunicacion de 19200 baudios establecido por el fabricante del modulo
12   mySerial.begin(19200);
13   Serial.begin(19200);
14   delay(500);
15   //Se establecen entradas digitales
16   pinMode(pul_sms_1, INPUT);
17   pinMode(pul_sms_2, INPUT);
18   pinMode(pul_sms_3, INPUT);
19   pinMode(dsw_call, INPUT);
20   //Se activa el modo PULL-UP en las entradas digitales
21   digitalWrite(pul_sms_1, HIGH);
22   digitalWrite(pul_sms_2, HIGH);
23   digitalWrite(pul_sms_3, HIGH);
24   digitalWrite(dsw_call, HIGH);
25 }
```

*Figura B4. 5*

5.6 Se crea varias funciones que permitirán enviar mensajes de texto

```
26 //Funcion para enviar texto
27 void EnviarMensajeTexto1()
28 {
29   mySerial.print("AT+CMGF=1\r");    //Codigo AT para disponibilidad de enviar un sms
30   delay(100);
31   mySerial.println("AT+CMGS=\"+593995474166\"");//Codigo AT para enviar un sms, EDITAR NUMERO AQUI!!
32   delay(100);
33   mySerial.println("HOLA COMO ESTAS?");//Contenido del mensaje hasta encontrar un Ctrl-Z
34   delay(100);
35   mySerial.println((char)26);//Codigo para Cntrl-Z en ascii
36   delay(100);
37   mySerial.println();
38   analogWrite(9, 200);
39   delay(1000);
40   analogWrite(9,0);
41 }
```

*Figura B4. 6*

Como se puede observar en la Figura B4.6, en la línea 31 se puede modificar el número del destinatario del mensaje de texto y en la línea 33 se introduce el texto del SMS en este caso es ***"HOLA COMO ESTAS?"***

5.7 Crear dos funciones más (***EnviarMensajeTexto2() y EnviarMensajeTexto3()***) estas pueden variar de número de destinatario y de texto. Figura B4.7

```
//Funcion para enviar texto
void EnviarMensajeTexto2()
{
  mySerial.print("AT+CMGF=1\r");    //Codigo AT para disponibilidad de enviar un sms
  delay(100);
  mySerial.println("AT+CMGS=\"+593995474166\"");//Codigo at para enviar un sms, EDITAR NUMERO AQUI!!
  delay(100);
  mySerial.println("Llamame después estoy ocupado");//Contenido del mensaje hasta encontrar un Ctrl-Z
  delay(100);
  mySerial.println((char)26);//Codigo para Cntrl-Z en ascii
  delay(100);
  mySerial.println();
  analogWrite(9, 200);
  delay(1000);
  analogWrite(9,0);
}
//Funcion para enviar texto
void EnviarMensajeTexto3()
{
  mySerial.print("AT+CMGF=1\r");    //Codigo AT para disponibilidad de enviar un sms
  delay(100);
  mySerial.println("AT+CMGS=\"+593995474166\"");//Codigo at para enviar un sms, EDITAR NUMERO AQUI!!
  delay(100);
  mySerial.println("Sensor de movimiento activado");//Contenido del mensaje hasta encontrar un Ctrl-Z
  delay(100);
  mySerial.println((char)26);//Codigo para Cntrl-Z en ascii
  delay(100);
  mySerial.println();
  analogWrite(9, 200);
  delay(1000);
  analogWrite(9,0);
}
```

*Figura B4. 7*

5.8 Ahora se crea una función como se observa en la Figura B4.8 para realizar una llamada de voz

```
//Funcion para realizar llamadas de voz
void LlamadaDeVoz()
{
  mySerial.println("ATD + +593998487091;");//Editar el numero a llamar AQUI!!
  delay(100);
  mySerial.println();
  analogWrite(9, 200);
  delay(1000);
  analogWrite(9,0);
}
void ShowSerialData()
{
  while(mySerial.available()!=0)
    Serial.write(mySerial.read());
}
```

*Figura B4. 8*

5.9 En el ***void loop(),*** se llama a las funciones anteriormente creadas (Figura B4.9). Estas funciones se ejecutarán únicamente si los botones designados a ellas se oprimieron

```
void loop()
{
//Cuando la comunicacion serial este lista espera a que se pulse algún
//botón para enviar mensajes ya establecidos o una llamada de voz
lectura1=digitalRead(pul_sms_1); // Se guarda el valor del pulsador en una variable lectura1
    if(lectura1 == LOW)
    {
      EnviarMensajeTexto1(); // Si se pulsó el botón 1 se enviara el sms
      if (mySerial.available())
    Serial.write(mySerial.read());
    }
lectura2=digitalRead(pul_sms_2);
    if(lectura2 == LOW)
    {
      EnviarMensajeTexto2();
      if (mySerial.available())
    Serial.write(mySerial.read());
    }
lectura3=digitalRead(pul_sms_3);
    if(lectura3 == LOW)
    {
      EnviarMensajeTexto3();
      if (mySerial.available())
    Serial.write(mySerial.read());
    }
lectura4=digitalRead(dsw_call);
    if(lectura4 == LOW)
    {
      LlamadaDeVoz();
      if (mySerial.available())
    Serial.write(mySerial.read());
    } //Se imprime en la comunicacion serial lo que el modulo transmite de retorno
}
```

*Figura B4. 9*

5.10 Una vez concluido el programa, se procede a guardar el archivo en una carpeta para eso dirigirse a: ***Archivo →Salvar →"Seleccionar donde se guardará el archivo"***. Ahora se puede cargar el programa en el módulo Arduino Uno, para ello conectar el módulo Arduino a la PC con el cable USB-AB (Figura B4.10) y seleccionar la placa que se va a usar (Figura B4.11), y el puerto COM al que se encuentra conectado (Figura B4.12). Ya configurado lo anterior se selecciona el botón subir (Figura B4.13)

*Figura B4. 10 Cable USB-AB*

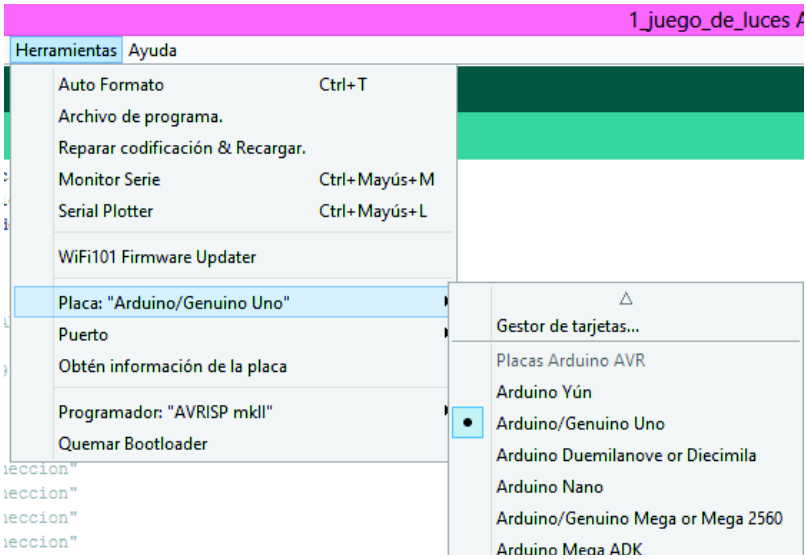



*Figura B4. 11 Selección de placa*

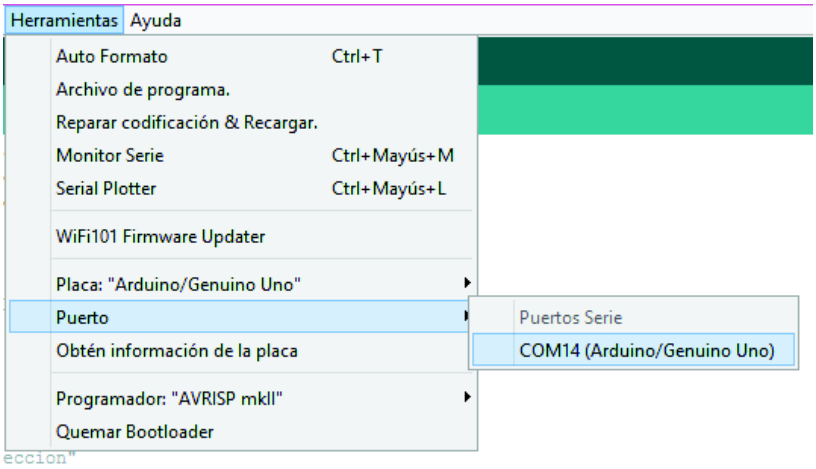



*Figura B4. 12 Selección de puerto*


*Figura B4. 13*

5.11 La Figura B4.14 muestra que se subió con éxito el programa

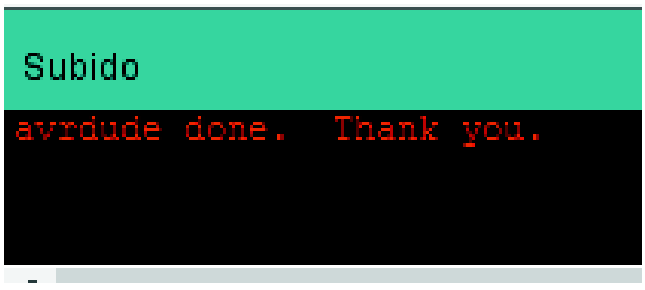



*Figura B4. 14 Confirmación de carga*

5.12 Una vez grabado el programa en el Arduino, se desconecta de la PC hasta realiza las conexiones

5.13 Colocar el módulo Shield GSM encima del Arduino Uno como se muestra en la Figura B4.15

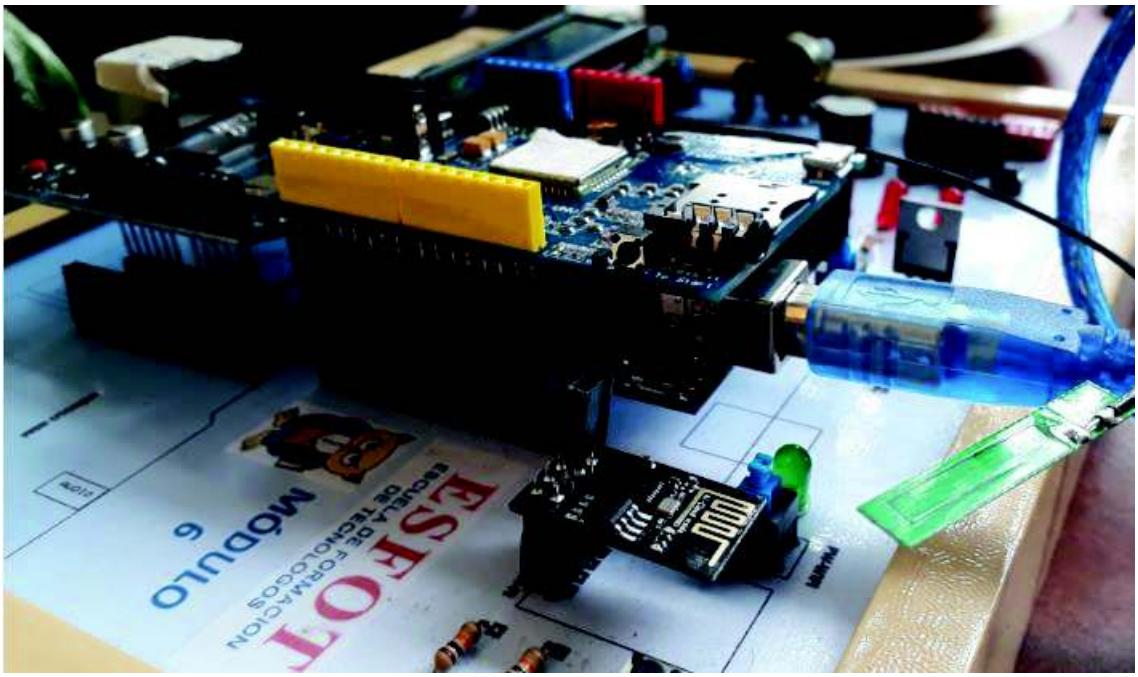



*Figura B4. 15 Ensamblaje shield GSM-Arduino*

5.14 Insertar el MicroChip en la ranura, como se muestra en la Figura B4.16 y se vuelve a conectar el modulo a la PC

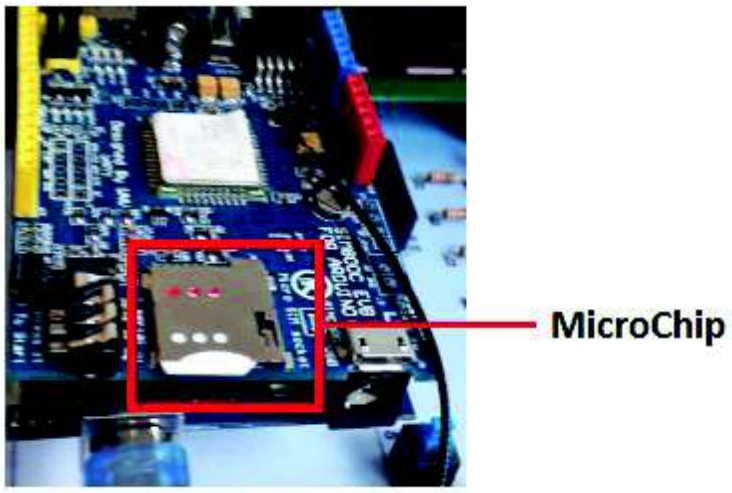



*Figura B4. 16 Ubicación del micro-chip GSM*

***NOTA 1:** El MicroChip puede ser de cualquier operadora

***NOTA 2:** El destaje que posee en microchip en una de sus esquinas debe estar como en la imagen, en la parte de afuera y boca abajo las conexiones y presionar hasta sentir el switch.

5.15 Realizar las siguientes conexiones: (Las conexiones se las puede observar en la Figura B4.17)

- D3 del Arduino a B1 de la Placa Didáctica

- D4 del Arduino a B2 de la Placa Didáctica

- D5 del Arduino a B3 de la Placa Didáctica

- D6 del Arduino a SW1 de la Placa Didáctica

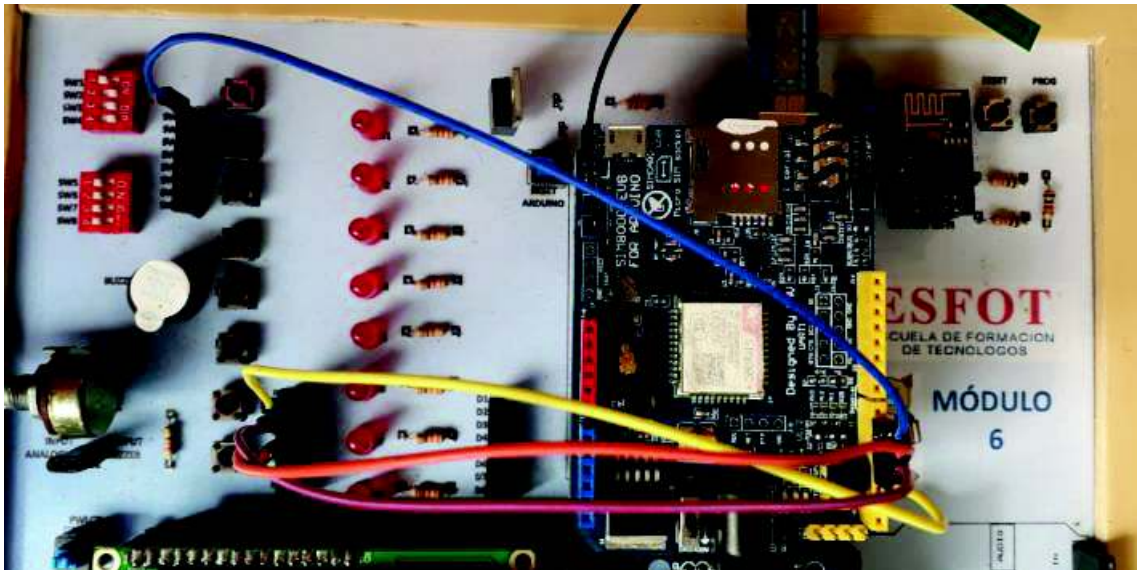



*Figura B4. 17 Conexión en la palca didáctica*

5.16 Una vez conectado el modulo se debe tener en cuenta que el módulo GSM se encuentre en modo Arduino. El diodo PWR del módulo GSM indicará si se encuentra en este modo. (Figura B4.18)

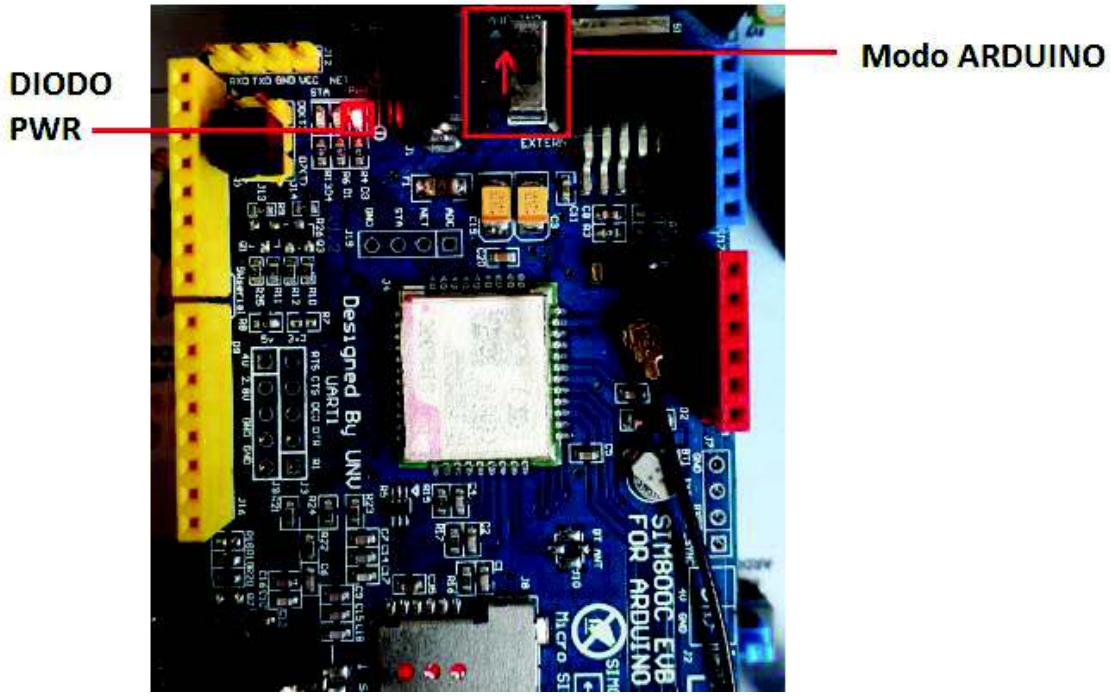



*Figura B4. 18 Switch de alimentación*

5.16 Presionar el botón de inicialización (Figura B4.19) del módulo hasta que el diodo STA se encienda y el diodo NET comience a parpadear (Figura B4.20); este último parpadeará de forma rápida durante unos

segundos indicando que está buscando una red de telefonía móvil; cuando parpadee de forma lenta se puede proceder a enviar los mensaje de texto y a realizar las llamadas

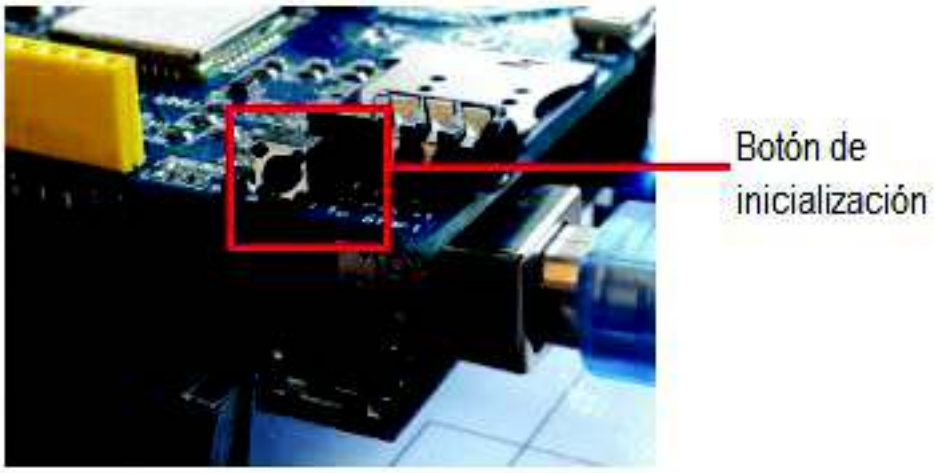



*Figura B4. 19 Botón de iniciación*

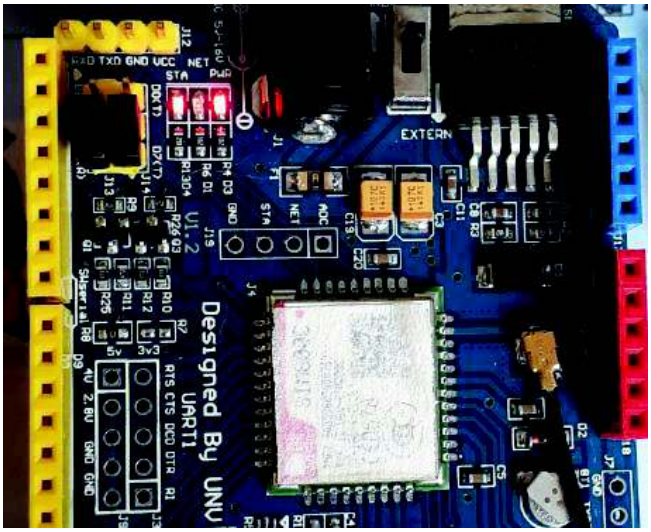



*Figura B4. 20 Confirmación de red*

***NOTA 3:** Una vez terminada la práctica es necesario que se retire todos los cables y se regresen todos sus componentes a sus posiciones originales

## 6. RECOMENDACIONES

- Se recomienda que se tenga conocimientos básicos de códigos AT

- Se necesita un microChip de cualquier operadora con saldo disponible

- Cuando las conexiones del circuito se encuentren realizadas y, se desee probar el funcionamiento del módulo GSM, es necesario desconectar la extensión USB y conectar directamente el módulo Arduino a la PC utilizando, el cable USB- A/B de color azul que se proporciona con el módulo didáctico

## 7. REFERENCIAS

[1] rambal, «Shield SIM800C GPRS/GSM para Arduino,» [En línea]. Available: http://rambal.com/shields-arduino/687-shield-sim800c-gprs-gsm-para-arduino.html.

[2] M. Sandel, «COMANDOS AT PARA MODEM GSM,» 05 11 2012. [En línea]. Available: https://miguelsandel.wordpress.com/2012/11/05/comandos-at-para-los-modem-gsm/.

[3] G. wiki, «GPRS Shield V2.0,» [En línea]. Available: http://www.geeetech.com/wiki/index.php/GPRS_Shield_V2.0.

# ANEXO C: HOJAS GUÍA DE ESTUDIANTE

# ANEXO C1
# PRÁCTICA 1: MANEJO DE PUERTOS DE ENTRADA Y SALIDA UTILIZANDO EL MÓDULO ARDUINO UNO

# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

## HOJA GUÍA

## PRÁCTICA 1

**1. TEMA:** Manejo de puertos de entrada y salida utilizando el módulo Arduino Uno.

**2. OBJETIVOS:**

2.1. Relacionar al estudiante con la programación de módulos Arduino utilizando el IDE de éste.

2.2. Implementar un circuito utilizando las placas didácticas de Arduino Uno para el manejo de puertos de entrada/salida.

**3. TRABAJO PREPARATORIO**

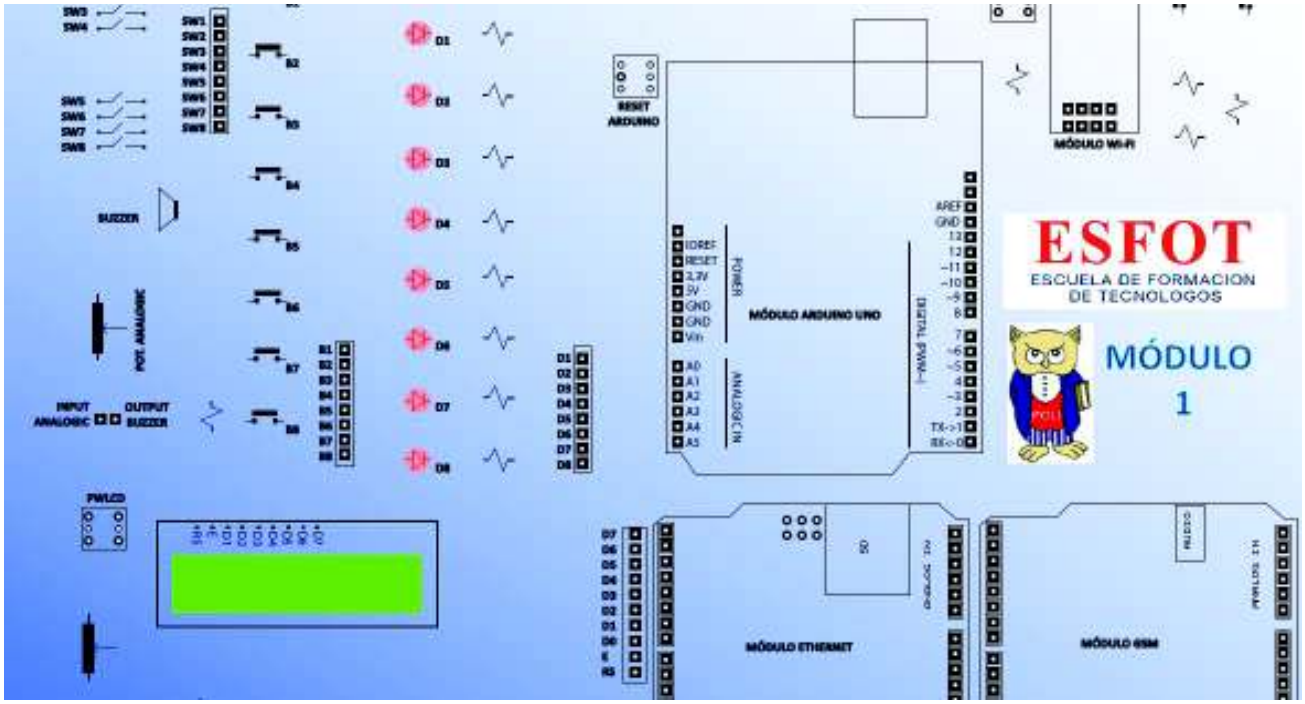
**3.1. CUESTIONARIO**

-Realizar una descripción que señale cuantos pins analógicos, digitales y PWM tiene y un dibujo de la arquitectura del Arduino Uno.

-Consultar la estructura del programa utilizado en el IDE de Arduino llamado *"SKETCH"*.

-Consultar la función de los comandos: *pinMode(pin,mode), digitalWrite(pin,value), digitalRead(pin), analogRead(pin) y analogWrite(pin,value)*.

- En el siguiente gráfico unir las conexiones solicitadas a continuación:
  - El pin 2 del Arduino a RS de la Placa Didáctica.
  - El pin 4 del Arduino a E de la Placa Didáctica.
  - El pin 7 del Arduino a D4 de la Placa Didáctica.
  - El pin 8 del Arduino a D5 de la Placa Didáctica.
  - El pin 12 del Arduino a D6 de la Placa Didáctica.
  - El pin 13 del Arduino a D7 de la Placa Didáctica.

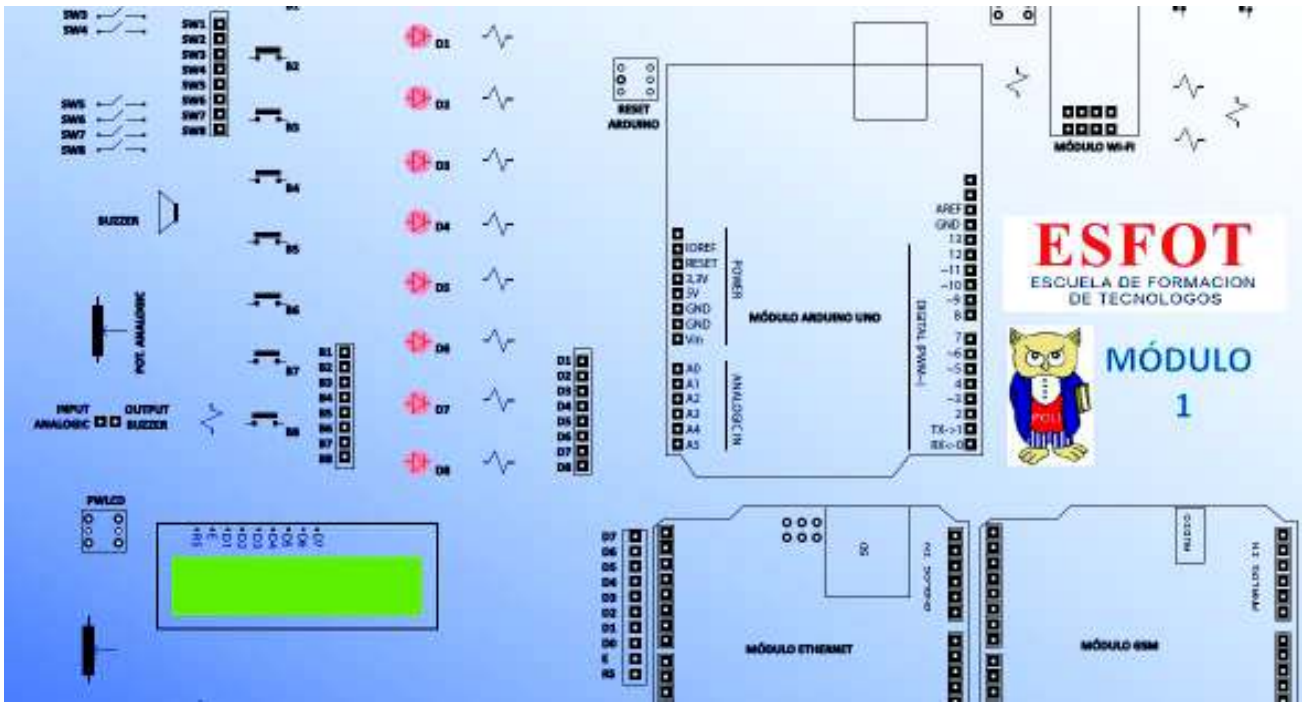



- En el siguiente gráfico unir las conexiones solicitadas a continuación:

  - El pin 11 del Arduino a D1 de la Placa Didáctica.

  - El pin 10 del Arduino a D2 de la Placa Didáctica.

  - El pin 9 del Arduino a D3 de la Placa Didáctica.



## 4. DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA

4.1. Elaborar un programa que permita realizar dos juegos de luces controlados por un potenciómetro.

Primer juego de luces: Cuando el potenciómetro se encuentre de la mitad hacia abajo de su capacidad, 6 leds deberán ir degradándose uno a uno hasta apagarse; una vez apagados los 6 leds, deberán encenderse gradualmente uno a uno.

Segundo juego de luces: Cuando el potenciómetro se encuentre de la mitad hacia arriba de su capacidad los 6 leds deberán encenderse ya sea de abajo hacia arriba o viceversa.

4.2. El valor del potenciómetro deberá ser indicado en el LCD de tal forma que el mínimo de su capacidad debe ser 0 y su máximo 1024.

4.3. Armar el circuito necesario utilizando las placas didácticas de Arduino Uno, grabar en el Arduino el programa desarrollado y comprobar su funcionamiento

4.4. Elaborar un programa que permita ingresar el código binario a través de 4 puertos de un dipswitch y que se presente el dígito correspondiente en un display de 7 segmentos ánodo común.

## 5. MATERIALES

- Placa didáctica Arduino Uno

# ANEXO C2
# PRÁCTICA 2: MANEJO DEL MÓDULO *SHIELD ETHERNET* UTILIZANDO MÓDULOS ARDUINO

# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

## HOJA GUÍA

## PRÁCTICA 2

**1. TEMA:** Manejo del Módulo Shield Ethernet utilizando módulos Arduino Uno.

**2. OBJETIVOS:**

2.1. Relacionar al estudiante con la programación de módulos Arduino utilizando el IDE de éste.

2.2. Implementar un circuito utilizando las placas didácticas de Arduino Uno para el manejo de módulos Shield.

2.3. Controlar el estado de los elementos pasivos como leds y buzzers a través de una interfaz gráfica

**3. TRABAJO PREPARATORIO**

**3.1. CUESTIONARIO**

-¿Qué es un módulo Shield Ethernet?

-Realice un dibujo de la arquitectura de módulo

-Consultar la librería que utiliza el módulo Shield Ethernet en el IDE de Arduino.

-Consultar cuál es la utilidad de la librería SPI en Arduino.

- Consultar cuáles son los códigos HTML para insertar: Título, encabezado y cuerpo.

**4. DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA**

4.1. Elaborar un programa que permita controlar 2 leds, un buzzer y observar la variación de un potenciómetro desde una interfaz gráfica ubicada en un servidor web.

La página web debe contener: encabezado de bienvenida, opciones para el encendido y apagado de los leds, buzzer y se pueda observar la variación de la entrada analógica.

4.2. Armar el circuito necesario utilizando las placas didácticas, grabar el programa desarrollado y comprobar su funcionamiento

**5. MATERIALES**

- Placa didáctica Arduino Uno
- Cable de Red (RJ45)

# ANEXO C3
# PRÁCTICA 3: MANEJO DEL MÓDULO WIFI.

# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

## HOJA GUÍA

## PRÁCTICA 3

**1. TEMA:** Manejo del Módulo WiFi.

### 2. OBJETIVOS

2.1. Relacionar al estudiante con la programación que proporciona el IDE de Arduino.

2.2. Implementar un circuito utilizando las placas didácticas de Arduino Uno para el manejo de un módulo WiFi.

2.3. Controlar el estado de los elementos pasivos como leds a través de una interfaz gráfica proporcionada por una página web

### 3. TRABAJO PREPARATORIO

### 3.1. CUESTIONARIO

-¿Qué es un módulo WiFi (modulo ESP8266)?

- ¿Cuál es su distribución de pines? Realizar un dibujo.

-¿Cuáles son las librerías que utiliza este módulo?

-¿Qué rol cumple el módulo Arduino Uno con el módulo WiFi?

### 4. DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA

4.1. Elaborar un programa que permita controlar 2 leds desde una interfaz gráfica ubicada en un servidor web.

La página web debe contener: encabezado de bienvenida y opciones para el encendido y apagado de los leds.

4.2. Armar el circuito necesario utilizando las placas didácticas, grabar el programa desarrollado y comprobar su funcionamiento

### 5. MATERIALES

- Placa didáctica Arduino Uno
- Access Point (también puede usarse un celular que funcione como AP *"conexión compartida"*)

**ANEXO C4**
**PRÁCTICA 4: MANEJO DEL MÓDULO *SHIELD* GSM SIM800C.**

# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

## HOJA GUÍA

## PRÁCTICA 4

**1. TEMA:** Manejo del Módulo Shield GSM SIM800C.

**2. OBJETIVOS:**

2.1. Relacionar al estudiante con la programación que proporciona el IDE de Arduino.

2.2. Implementar un circuito utilizando las placas didácticas de Arduino Uno para el manejo de un módulo Shield GSM SIM800C.

2.3. Enviar mensajes de texto usando el Módulo Shield GSM SIM800C.

**3. TRABAJO PREPARATORIO**

**3.1. CUESTIONARIO**

-¿Qué es un módulo Shield GSM SIM800C?

- ¿Cuál es su distribución de pines? Realizar un dibujo.

-¿Cuáles son los códigos AT para disponibilidad de enviar un SMS, enviar SMS y realizar llamadas?

**4. DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA**

4.1. Elaborar un programa que permita enviar 3 SMS al presionar un pulsador 1,un pulsador 2 y un pulsador 3 respectivamente y, que realice una llamada de voz al encender un dipswitch a cualquier número celular.

**5. MATERIALES**

- Placa didáctica Arduino Uno
- MicroChip de cualquier operadora de telefonía celular

# ANEXO D: MANUAL DE MANTENIMIENTO

# ESCUELA POLITÉCNICA NACIONAL
## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

## MANUAL DE MANTENIMIENTO

### INTRODUCCIÓN

Las principales causas de fallo de los equipos eléctricos y electrónicos son: la falta de mantenimiento preventivo, desgaste natural de los elementos y errores humanos. Con el fin de maximizar el tiempo de vida útil de la placa didáctica se mencionará los errores más comunes y como resolverlos además, se da consejos que serán útiles para mantener operativos los elementos de la placa, para un funcionamiento óptimo del mismo.

### MANTENIMIENTO PREVENTIVO

La placa con el pasar del tiempo puede presentar desgaste debido a condiciones climáticas y desgaste por el uso, estas condiciones ocasionan en los elementos: corrosión, deformaciones, oxidación, rayones y entre otras, afectando en el rendimiento del equipo para lo cual se recomienda:

- Conservar las placas en un lugar libre de humedad y cubrirlas para evitar acumulación de polvo o particular volátiles.
- Limpiar una vez cada dos meses el polvo acumulado en las placas.
- Con ayuda de una aspiradora o herramienta de aire comprimido se puede retirar los residuos de cables atascados en los espadines hembra.
- En caso de oxidación, se recomienda pasar la zona afectada con un cepillo y alcohol isopropílico o *thinner;* se debe tener cuidado que las substancias no entren en contacto con la piel y ojos.

- Comprobar el estado de los espadines, en el caso de que se encuentren doblados usar una pinza y tratar de enderezarlos. Si se encuentra rota reemplazarla por otra.

## MANTENIMIENTO CORRECTIVO

Si un elemento de placa didáctica deja de funcionar a pesar de su mantenimiento preventivo o por algún error de operación, se procede a reemplazarlo de acuerdo con el diseño mostrado en el "Capítulo 3 Resultados y Discusión". Para lo cual se tiene las siguientes recomendaciones:

Para los módulos: *Arduino* uno, *Ethernet*, *WiFi* y *GSM* se los puede reemplazar cuando se lo requiera; pero si se desea reemplazar un *Arduino uno* por uno nuevo se debe intercambiar los espadines hembra existentes, por unos espadines macho de patas largas, esto es necesario para que pueda acoplarse en la placa didáctica y de esta manera energizarla y comunicarse con los elementos.

En el caso de los elementos soldados en la placa se los puede reemplazar por nuevos si se diera el caso. Para lo cual se debe desmontar la baquelita de los marcos de madera para apreciar el circuito.

## ERRORES GENERALES Y RECOMENDACIONES

- El *LCD* presenta caracteres extraños, si esto sucede:
    - Quitar la energía y comprobar que sus espadines se encuentren bien conectado a la base.
    - Comprobar que este bien conectados los cables del LCD al Arduino.
    - Comprobar que este activado el *switch* de *Power* del LCD.
    - Comprobar que los pines del programa concuerden con los conectados y volver a grabar el programa.
    - Encenderlo.

- Si los LEDs verdes de los *display* de 7 segmentos se atenúan, no preocuparse, es normal; debido a la corriente que proporciona el Arduino da preferencia al *display* y tiende a atenuar la de los LEDs verdes debido a que requieren más corriente. Estos LEDs solo indican si esta activos los *display* y no afectan al funcionamiento.

- Si en el módulo *WiFi* no se graba el programa:

    - Primero, quitar la energía y comprobar el que *switch Reset* cerca del Arduino este presionado.
    - Comprobar que el *switch* de encendido del módulo *WiFi* este presionado.
    - Comprobar que en el IDE del Arduino este: ***Herramientas***->***Placa***-> ***Generic ESP8266 module*** y en: ***Herramientas***->***Puerto***-> ***COM*** (puerto al que se encuentra conectado la placa).
    - Finalmente antes de grabar se debe mantener presionado el botón *Reset* del módulo *WiFi* y sin soltar energizar la placa, esperar 5 segundos, pasado este tiempo debe grabar el programa.

- No se permite grabar los programas en el Arduino, si esto pasa:

    - Quitar la energía y comprobar que este activo en el IDE de Arduino: ***Herramientas***->***Placa***->**Arduino Uno** y ***Herramientas***->***Puerto***-> ***COM*** (puerto al que se encuentra conectado la placa).
    - Después verificar que los pines **0(*RX*)** y **1(*TX*)** estén desconectados solo para grabar, una vez cargado el programa se los puede utilizar sin problemas.

- El módulo GSM no envía mensajes, si esto pasa:
    - Quitar la energía y comprobar que este todo correcto y haciendo contacto.
    - Comprobar que el chip esta con saldo y funcional.
    - Comprobar en el programa, si se encuentra bien digitado el número destinatario de los mensajes.
    - Grabar el programa de nuevo.
    - Una vez energizado comprobar que: el *switch* de energía del módulo GSM este en modo Arduino y tenga la antena conectada.
    - Presionar unos 5 segundos el botón de arranque del GSM hasta que el LED del medio comience a titilar.
    - Esperar a que el parpadeo de led sea lento, aproximadamente 1 o 2 parpadeos cada segundo.

- Si del potenciómetro de contraste sale humo:
    - Quitar la energía inmediatamente hasta que enfríe.
    - Mover el potenciómetro hasta la mitad de su capacidad y volver a energizar.
    - Ajustar hasta que el LCD sea visible y no mover.
    - El potenciómetro no tiene una resistencia de regulación por lo que llevarlo al extremo de 0 ohm se crea un corto circuito que genera el calor en el potenciómetro. Esta parte está señalizada y se recomienda no moverlo.