Reporte de práctica 1
(Enero 31 2018)

Thomas, Manuel & Ruz, Pablo. *Principios de Mecatrónica, ITAM*

***Abstract*—El objetivo del presente reporte es realizar un análisis de la práctica de Linux/GNU. A lo largo de dos sesiones, se buscaron implementar comandos en la terminal y llevar a cabo la programación de un Arduino en diversas presentaciones.**

***Index Terms*—Linux, GNU, Arduino, Terminal, Principios de Mecatrónica**

I. Introducción

Un Arduino es una plataforma de desarrollo de computación física y de código abierto, cuyas funciones dependen en gran medida de un microcontrolador que le permite, junto con los puertos y pines, tomar información del ambiente y actuar en consecuencia, de acuerdo a lo que en ella ha sido programado. Arduino cuenta con un software compatible con Linux por medio del cual se llevan a cabo todas las programaciones de la tarjeta. Es por ello que el entendimiento básico del funcionamiento de Linux es necesario.

Linux es un sistema operativo basado en UNIX. Este último comenzó a ser desarrollado en la década de los 70 por estudiantes universitarios de Estados Unidos en lenguaje C. Existe una gran variedad de versiones de UNIX. Algunas de ellas son de libre distribución, tal como Linux, pero también las hay comerciales, como Solaris o AXP. Linux fue desarrollado por el finlandés Linus Torvalds, y a la fecha se ha convertido en uno de los principales sistemas operativos, usado principalmente en servidores gracias a su enorme versatilidad y capacidad.

Así, el objetivo principal de esta práctica consistió en aprender a utilizar la terminal de Linux y, posteriormente, llevar a cabo la programación del Arduino. Como extra, también se aprendió a utilizar Github como repositorio para los trabajos.

II. Marco teórico

La tarjeta del Arduino funciona, en gran parte, gracias al microcontrolador que tiene embebido. La lógica del microcontrolador se muestra en la *Fig.1*.

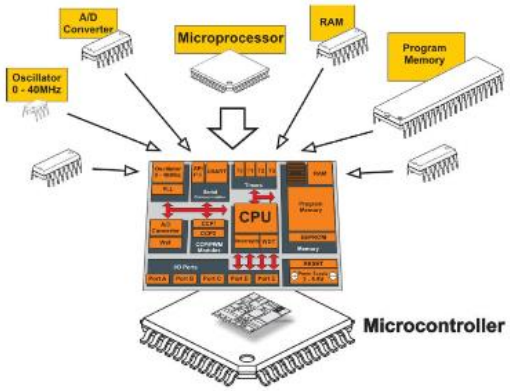


Fig. 1
Como puede observarse, el microcontrolador está constituido por varias partes, tales como un microprocesador, una memoria de programa, un oscilador, convertidor análogo digital y una RAM.

Cada una de las partes del microcontrolador permiten el funcionamiento correcto del Arduino, mismo que se muestra en la

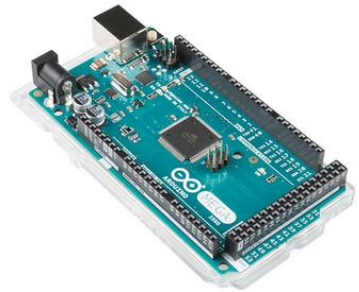


Fig. 2
Este es un Arduino Mega 2560 R3. Cuenta con puertos para funcionamiento analógico y otros para funcionamiento digital. Obsérvese el microcontrolador que se encuentra en el centro, mismo que se encuentra conectado a varios de las entradas y salidas del aparato.

El software de Arduino en Linux es de fácil uso y es capaz de ejecutar funciones por medio de código mínimo. Únicamente es necesario hacer un *setup*: método que es llamado únicamente cuando el Arduino se enciende o se reinica; y un *loop:* función que corre continua e infinitamente mientras el dispositivo esté encendido; es en este método donde debe de desarrollarse la mayor parte de la lógica de lo que el Arduino deberá ejecutar.

III. Desarrollo

PARTE 1

La práctica fue dividida en dos etapas. En la primera de ellas, se comprendió el uso de Linux y de los comandos más conocidos en su terminal. Se llevó a cabo la creación de los directorios y archivos, según se muestra en la *Fig. 3*.

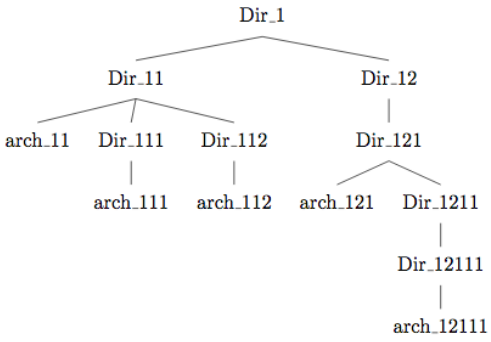


Fig. 3
Esquema de implementación para la creación de directorios y archivos por medios de la terminal de comandos de Linux

La variedad de comandos utilizada fue grande para lograr la creación del esquema superior. A continuación se enlistan los más relevantes, así como su funcionamiento básico:

- *cd*: cambio de directorio. Al ingresar este comando seguido de una dirección de carpetas o archivos dentro de la computadora, el directorio se modifica, de suerte que los nuevos comandos que se ejecuten se hagan dentro del directorio al que se redirigió.
- *mkdir*: crea un nuevo directorio en la dirección especificada. Obsérvese que en la *Fig. 3* hay varios Dirxx; todos ellos fueron creados con el presente comando.
- *touch*: este comando, al actualizar el timestamp de un archivo, es capaz de crearlo cuando éste no existe. Así, todos los archxxx fueron creados con este comando.
- *ls*: este comando despliega la lista de todos los archivos y directorios que se encuentran debajo de aquel en el que se está trabajando en ese momento. Con esto fue posible ir verificando paso por paso que la creación del árbol se estuviera llevando a cabo de la manera correcta.
- *rm* y *rm –r:* el primero de ellos elimina archivo por archivo, mientras que el segundo elimina todos los archivos que se encuentran dentro de un directorio, puesto que se hace de manera recursiva. Es importante hacer notar que el uso del segundo comando puede resultar peligroso si no se sabe bien lo que se está haciendo, puesto que es irreversible.

~~PARTE~~ 2

La parte dos de la práctica consistió en la programación del Arduino bajo diversos esquemas. El primero de ellos consistió en lograr que un led encendiera y apagara cada determinado tiempo. La figura que lo describe se muestra a continuación en la *Fig. 4*

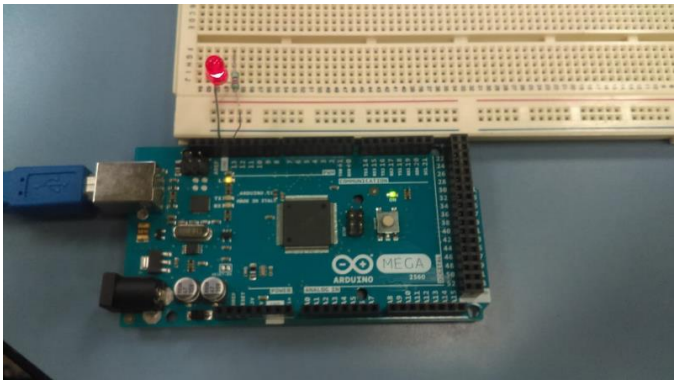


Fig. 4
Aquí se presenta la primera configuración. El led está conectado a un puerto de salida (puerto 13) y a una resistencia, que a su vez está conectada a tierra, generando así el circuito.

El código para que este led funcione es muy simple. Únicamente se tiene que indicar que el led estará conectado al puerto número 13, programar este puerto como salida y dentro del *loop* prender y apagar el led, dando un tiempo de un segundo para que se note el cambio.

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Fig. 5
Nota: aquí falta la línea de código int LED_BULLETIN = 13

En la figura de arriba, obsérvese que en el ciclo *loop*, por medio de la instrucción digitalWrite(LED_BULLETIN, HIGH) se establece que hay que prender el led y poner un voltaje alto (3V) para que éste encienda. Posteriormente, el delay(1000) indica que hay que esperar mil milisegundos (un segundo) para que se ejecute la siguiente instrucción, que es apagar el led poniendo el voltaje en 0.

La segunda configuración del circuito fue mucho más compleja y consistió en lograr que el led prendiera y apagara de acuerdo a la señal recibida por una fotorsesistencia y también que, independientemente del estado en el que se encontrara, se pudiera lograr que pasara al estado opuesto (prendido→apagado o apagado→prendido) al momento de pulsar un botón. En la *Fig. 6* se muestra el circuito que se armó para llevar a cabo esta función.

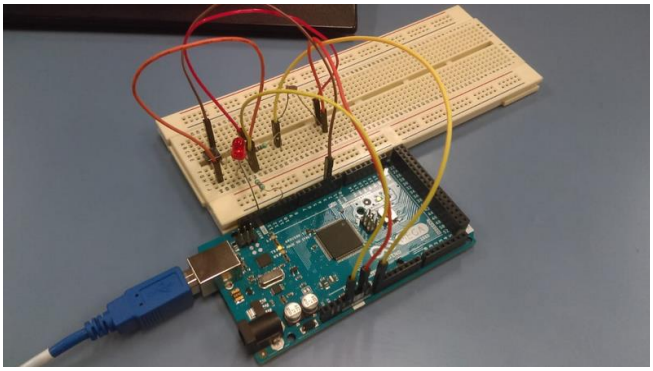


Fig. 6

El circuito implementado para la segunda parte es notablemente más complejo que el primero, al igual que el código usado para que su funcionamiento fuese el correcto. Obsérvese que hay varios cables conectados a tierra, al led y a la fotoresistencia para garantizar su funcionamiento.

El código que se usó para implementarlo consistió, a grandes rasgos, en configurar las entradas y salidas y ejecutar, si se cumplía la condición de que la cantidad de luz fuera mayor a cierto parámetro, apagar el led, y encenderlo en caso contrario. Así mismo, independientemente del estado, se programó el botón para que cambiara el led. Es importante hacer notar aquí que la fotoresistencia fue conectada a una entrada analógica de la tarjeta del Arduino, mientras que el led fue conectado a una salida digital.

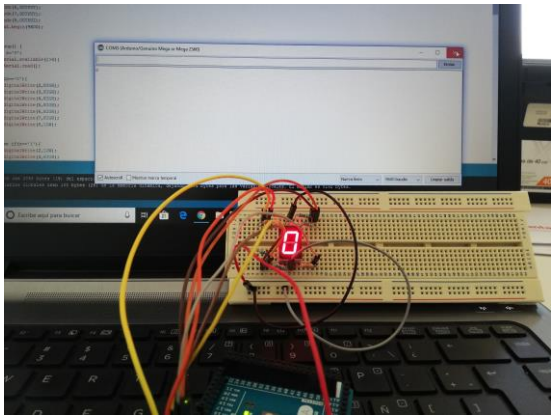
```
int r=0;

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(A0, INPUT);
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  buttonState = digitalRead(buttonPin);
  valorLDR= analogRead(A0);
  Serial.println(valorLDR);
  //digitalWrite(led,LOW);
  // Devolver el valor leido a nuestro monitor serial en el IDE de Arduino
  if(buttonState == HIGH){
    if(valorLDR < 1000){
      digitalWrite(led,HIGH);
    }
    else{
      digitalWrite(led,LOW);
    }
  }
  else{
    if(valorLDR > 1000){
      digitalWrite(led,HIGH);
    }
    else{
      digitalWrite(led,LOW);
    }
  }
  delay(200);
```

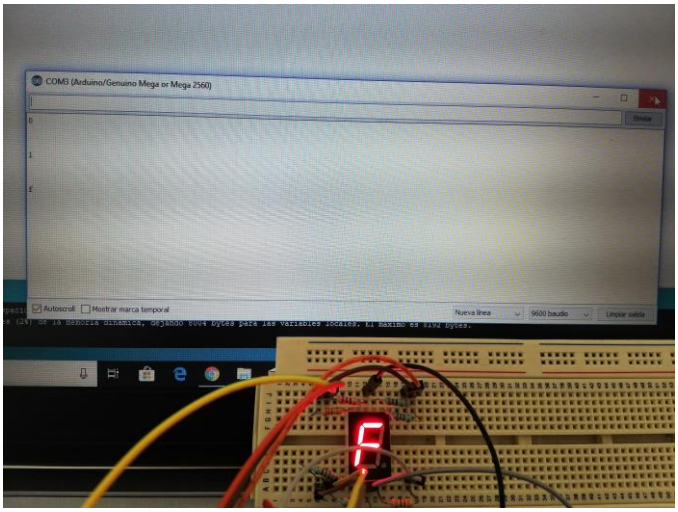
Obsérvese que en el código de arriba, el *setup* inicializa los pines (entradas) y puertos (salidas) y el loop ejecuta las instrucciones de modo que si el botón está presionado y el valor de la luz es menor a 1000, el led deberá de prenderse y, de lo contrario, apagarse. Así mismo, si el botón NO está presionado, el led deberá ejecutar las funciones de encenderse si el valor de la luz es mayor a 1000 y apagarse de lo contrario.

En la parte tres de esta práctica se necesitó implementar la siguiente tabla para hacer que los segmentos encendidos del display simularan cada uno de los caracteres.

Digit	gfedcba	abcdefg	a	b	c	d	e	f	g
0	0×3F	0×7E	on	on	on	on	on	on	off
1	0×06	0×30	off	on	on	off	off	off	off
2	0×5B	0×6D	on	on	off	on	on	off	on
3	0×4F	0×79	on	on	on	on	off	off	on
4	0×66	0×33	off	on	on	off	off	on	on
5	0×6D	0×5B	on	off	on	on	off	on	on
6	0×7D	0×5F	on	off	on	on	on	on	on
7	0×07	0×70	on	on	on	off	off	off	off
8	0×7F	0×7F	on	on	on	on	on	on	on
9	0×6F	0×7B	on	on	on	on	off	on	on
A	0×77	0×77	on	on	on	off	on	on	on
B	0×7C	0×1F	off	off	on	on	on	on	on
C	0×39	0×4E	on	off	off	on	on	on	off
D	0×5E	0×3D	off	on	on	on	on	off	on
E	0×79	0×4F	on	off	off	on	on	on	on
F	0×71	0×47	on	off	off	off	on	on	on

Mediante la entrada de datos vía keyboard el display de siete segmetos reflejaba los caracteres del uno al F en hexadecimal. Siendo esto codificado también en arduino :





IV. Resultados

Resultados (1 párrafo con la respuesta a las preguntas de la práctica)

V. Conclusiones

Pablo A. Ruz Salmones: usar el Arduino es mucho más sencillo que realizar todo el alambrado para lograr que un led encienda. Esto es, comparado con el curso de Circuitos Lógicos, la facilidad al momento de ensamblar las cosas fue mayor. Fue de mi interés especial la segunda parte de la práctica (la que incluye el botón del led), porque las aplicaciones de un circuito similar son muchas. Por ejemplo, los semáforos. Están enendidos o apagados de acuerdo con el flujo de automóviles, pero es posible que agentes de tránsito los modifiquen manualmente presionando un botón. Por el lado de Linux, la velocidad con la que se pueden ejecutar los comandos en la terminal es mucho mayor que estar "arrastrando y pegando" las cosas a partir de las carpetas. Resulta de gran utilidad para la creación de directorios de una manera más sencilla. Es cierto que hay que acostumbrarse, pero no es nada que no se pueda lograr con un poco de trabajo.

Victor M. Thomas Ruiz: El Arduino es una herramienta que facilita muchísimo la creación y el control de circuitos lógicos, la tarjea y el software permiten definir de forma clara el modo en el que algún circuito funcionará, desde secuencias lógicas y operaciones aritméticas hasta el control sobre los lapsos de tiempo que distintas funciones deben realizarse. Resulta mucho más eficiente tener una tarjeta Arduino integrada a un circuito que simplemente tener elementos y cableados en ausencia de este.

VI. Roles

Pablo A. Ruz Salmones: tanto la práctica como el reporte fueron realizados en equipo. Tanto en la parte de Linux como en la del arduino, no hubo división explícita del trabajo, lo que permitió que ambos aprendiéramos y desarrolláramos las mismas habilidades. Quizá la única diferencia fue una mayor colaboración de Manuel al terminar la práctica y una mía en el reporte, pero nada a tomar en consideración.

Manuel Thomas: No se realizó división como tal del trabajo, todo lo hicimos en equipo. Desde la sesión inicial donde trabajamos con la terminal de Linux hasta la elaboración del 'presente reporte el trabajo se hizo en conjunto, tomando decisiones entre los dos y desarrollando el trabajo de manera ordenada. Pablo, a diferencia mía desarrolló más la parte del reporte, mientras que yo completé las últimas líneas del código en Arduino para terminar la práctica.

VII. Referencias

[I] "Práctica 1. Comenzando con Arduino". Universidad de Cádiz. Disponible en: http://www.uca.es/recursos/doc/Unidades/Unidad_Innovacion/Innovacion_Docente/ANEXOS_2011_2012/22232441_310201212102.pdf

[II] R. Gómez "Introducción a Linux". Universidad de Sevilla. Disponible en: http://www.informatica.us.es/~ramon/articulos/IntroLinux.pdf

[III] R. Pratap. "Celebrating Arduino". Day 2014. Indian Institute of Technology Kanpur. Disponible en: http://students.iitk.ac.in/eclub/assets/lectures/embedded14/arduino.pdf

[IV] "What is Arduino". Arduino. Disponible en: https://www.arduino.cc/en/Guide/Introduction