Laboratorio de Principios de Mecatrónica

Práctica 1. Microcontroladores y Manejo de Versiones

Emilia Sofia Spinola Campos
Ingeniería Industrial e Ingeniería Mecatrónica
ITAM
Ciudad de México, México
sofispinola@gmail.com

Ricardo Edward Meadowcroft
Ing. en Computación y Licenciatura en Matemáticas Aplicadas
ITAM
Ciudad de México, México
rmeadowc@itam.m

Jorge Alejandro Ramírez Gallardo
Ingeniería Industrial e Ingeniería Mecatrónica
ITAM
Ciudad de México, México
rmrez.alejandro.g@gmail.com

***Abstract*— En el presente documento se recopila la información y experimentos realizados durante la práctica 1 del Laboratorio de Principios de Mecatrónica, donde se implementó la construcción de sistemas de directorios mediante la consola de Linux, se creó un repositorio de GitHub y se practicó el manejo de archivos entre un repositorio de GitHub y su contraparte generada en el escritorio y, finalmente, se codificaron y revisaron conceptos básicos de programación con Arduino, incluyendo el manejo de entrada y salida.**

Palabras clave*— *Ubuntu, codificación, Arduino, GitHub.

INTRODUCCIÓN

Para la primera práctica se busca, en primera instancia, hacer un reconocimiento de la utilidad que presentan los comandos de consola de Linux (concretamente, usando el sistema operativo Ubuntu) en lo referente a la manera en que se usa para manejar archivos. Por ser un sistema de libre acceso basado en la tecnología de software de GNU, Ubuntu es un sistema operativo idóneo para el desarrollo de los comandos dentro de una terminal.

En segunda instancia, también, es importante reconocer el hecho de que, en la actualidad, el desarrollo de código está familiarizado con el manejo de versiones, razón por la cual se pretende emplear el entorno de Git, particularmente para su uso con repositorios del servicio GitHub, para un acceso fácil de las versiones de nuestros códigos, las cuales deberán de guardarse en un repositorio compartido.

Finalmente, se busca una identificación de las bases de programación del microcontrolador Arduino: plataforma de desarrollo de prototipado rápido con un lenguaje basado en C, el cual nos permitirá ir familiarizándonos con el manejo y programación de un microcontrolador (en este caso, el AVR ATmega 2560).

El desarrollo y conocimiento de estos elementos introductorios permitirá al lector una mejor comprensión de la exposición del desarrollo y resultados del presente trabajo, así como su importancia en la introducción de los universitarios a los principios básicos de un sistema mecatrónico.

I. CONSTRUCCIÓN

A. *Material*

El material que se utilizó para el presente trabajo contiene los siguientes elementos.

- Una computadora con Ubuntu como sistema operativo con el software Arduino.
- AVR ATmega 2560.
- 1 Protoboard.
- 1 resistencia de 220 Ohms.
- 1 resistencia de 2.2 MOhms.
- Diodo LED rojo.
- 1 Sensor fotorresistivo.
- 1 Potenciómetro de 10kOhms.
- Alambrado suficiente para cablear.

B. *Selección del material*

En el caso del software, el sistema operativo Ubuntu y el software Arduino fueron preestablecidos por las condiciones del laboratorio, con la consideración de que, para el software de Arduino, se debe de ajustar en la ventana del código el tipo de microcontrolador que se está utilizando (AVR ATmega 2560).

Para la selección del material físico, tanto las resistencias como el potenciómetro seleccionado para el desarrollo del apartado II-C, se tomaron en consideración los elementos de construcción base que no podían ser modificables, a saber, la resistencia arrojada por el resistor fotosensible (que arrojaba una resistencia medible de 2.2 MOhms), además de que la resistencia de 220 Ohms se seleccionó para una protección adecuada del diodo LED empleado en la práctica (Figura 1).

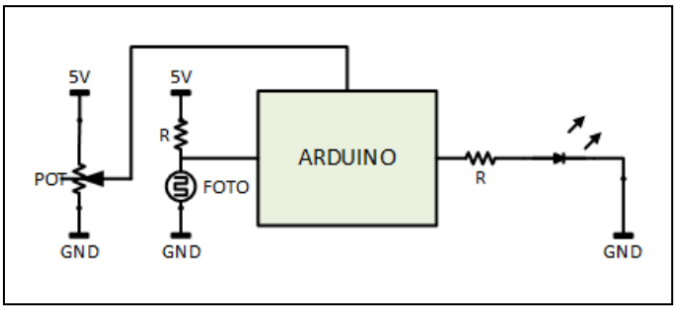

Figura 1. Conexiones básicas del apartado II-C.

II. DESARROLLO

A continuación, se presentan el desarrollo de la práctica, el cual se presenta en tres apartados diferentes: Linux, GitHub y Arduino.

A. Linux

El desarrollo de cada uno de los comandos dentro de la terminal de Linux debe realizarse conforme el siguiente procedimiento y conociendo, en primera instancia, cada uno de los comandos empleados en su ejecución. La siguiente lista contiene una recopilación de las funciones de cada comando.

Lista 1. Comandos de la consola Linux

- PWD: Significa 'print working directory' y al llamarse sin argumentos imprime a la consola la dirección completa en la que el usuario está localizado y sobre la que trabaja en el sistema de archivos.
- MKDIR: Significa 'make directory', y al escribirse el comando seguido de un nombre, crea un nuevo folder (directorio) ubicado en la dirección actual del usuario con el nombre dado.
- CD: Significa 'change directory' y se usa para cambiar el directorio en que está localizado el usuario de la consola. Llamado por si solo regresa al usuario al directorio raíz; si se llama seguido de una dirección empezando con '/' se cambia a ese directorio interpretada como dirección completa (desde raíz), mientras que sin tal signo lo interpreta como dirección relativa al directorio actual. Adicionalmente, si se escribe 'cd ..' se cambia al directorio un nivel arriba al actual, y con 'cd –' se cambia al directorio en que se estuvo anterior al actual.
- LS: Significa 'list' y se usa (sin argumentos) para enlistar todos los nombres de los archivos localizados en el directorio actual. Si se escribe después del comando una o más direcciones, se enlistará en su lugar los archivos localizados en estos. Si se escribe la bandera -l enlistará información adicional de cada archivo; con -a muestra también los archivos escondidos, y con -R muestra adicionalmente los archivos ubicados en subdirectorios dentro de los investigados.
- RMDIR: Significa 'remove directory' y si se escribe con una o más nombres de directorio, intentará eliminarnos del directorio actual, fallando si no están vacíos. Se pueden remover folders anidados, si no tienen otro contenido, llamando la secuencia anidada de folders como dirección separada por símbolos '/'.
- RM: Significa 'remove' y cuando se escribe con uno o más nombres de archivo eliminará tales archivos del directorio actual. Si se usa la bandera -r se puede escribir un nombre de un directorio, incluso no vacío, para borrarlo y también todo su contenido.
- TOUCH: Cuando se llama seguido de uno o más nombre, crea archivos nuevos sin contenido con tales nombres en el directorio actual. Si un archivo ya existe con ese nombre, actualiza la fecha de última modificación y acceso al tiempo actual sin realizar algún otro cambio.
- CP: Significa 'copy' y se usa para copiar archivos en tres modalidades: si se escriben dos nombres de archivo, se copia el contenido del primer archivo al segundo, creando un nuevo archivo en caso de no existir ya un archivo con el segundo nombre; si se escriben uno o más nombres de archivo seguido de un nombre de directorio, se copian los contenidos de los archivos a nuevos archivos creados en el directorio destino indicado bajo el mismo nombre; si se escriben dos directorios, se copiarán todos los archivos del primer directorio (usualmente usando la bandera recursiva -r) y se colocaran dentro del segundo dentro de un subdirectorio del mismo nombre (o creando el segundo directorio si no existe).
- MV: Significa 'move' y mueve archivos de una dirección a otra en múltiples modalidades: Si se escriben dos nombres de un archivo, el contenido del primer archivo se moverá al segundo, efectivamente renombrando el archivo; si se escriben uno o más archivos seguidos de un directorio, se moverán los archivos dentro del directorio dado quedando con el mismo nombre, dejando de estar presentes en el directorio actual; con dos nombres de directorio se renombrara el directorio con el nombre del primero al segundo.
- MORE: Al escribirse con el nombre de un archivo de texto existente en el directorio, despliega el contenido de texto en la consola. Como el contenido de texto puede ser mucho mayor al lo que se puede mostrar, una vez ejecutado el comando, presenta opciones adicionales para ver la información: presionar enter para mostrar línea por línea la información restante, presionar espacio para saltar línea por línea, y escribir 'b' para regresar una página.
- LESS: Realiza una acción similar a MORE pero presenta mayor funcionalidad en la visualización del contenido: por ejemplo, usando las teclas de flecha arriba y abajo se puede mover tanto arriba como abajo a nivel línea por línea, al escribir un número seguido de espacio se traslada la cantidad escrita de líneas, y al escribir '/' seguido de una palabra busca la palabra en el texto.
- Significa 'concatenate', y permite realizar, además de la concatenación de archivos, varias operaciones sobre archivos en distintas modalidades. Al escribir uno o varios nombres de archivos de texto, muestra sus contenidos en la consola; si se escribe '>' seguido de un nombre de archivo,

permitirá escribir texto en la consola hasta que se presiona ctrl+'d', después de lo cual el texto escrito es guardado con el nombre elegido, sobreescribiendo si ya existe. Si con un archivo ya existe se usa en su lugar '>>' el texto escrito será anexado al ya existente. Si se escriben uno o más nombres de archivo seguido de '>' y luego otro nombre, los contenidos de los archivos seran concatenados y guardados en un archivo nuevo con el nombre dúltimo (sobreescribiendo si ya existe); con '>>' la concatenación será anexada al archivo último si ya existe.

- HEAD: Si se escribe con el nombre de un archivo existente, mostrará las primeras 10 líneas del archivo. Si se escribe con la bandera -n seguido de un número, mostrará en su lugar la cantidad pedida de líneas iniciales. Similarmente, -c muestra la cantidad de bytes iniciales pedidos.
- TAIL: Funciona análogamente a head excepto que imprime las últimas líneas del archivo, con 10 por default, con mismas opciones de bandera. En adición, con la bandera -f se escribirá a la consola nuevas líneas conforme se vaya modificando el archivo por procesos externos.
- PS: Significa 'process status' y al escribirse por sí solo muestra todos los procesos corriendo actualmente con número de identificación, tiempo corriendo, y comando asociado.
- TOP: Muestra una lista dinámica de los procesos corriendo, actualizandose automáticamente hasta presionar 'q' para salir. Muestra enlistados varios tipos de información de los procesos incluyendo la prioridad, y el porcentaje de memoria y CPU que lo usa.
- MAN: Significa 'manual', y al escribirlo seguido de un nombre válido de comando del terminal linux, muestra una descripción de para qué y cómo se usa el comando dado en secciones estructuradas. Si se escribe antes del nombre un número muestra directamente la sección de número correspondiente.

El procedimiento para llevar a cabo la práctica es el siguiente. Por medio de los comandos de Linux, se creó la estructura de directorios correspondiente a la Práctica 1, ilustrada en la Fig. 2.

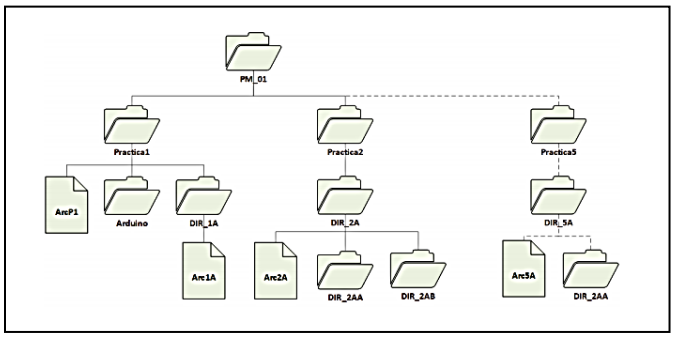

Figura 2. Estructura de directorios deseada.

Los comandos implementados deben realizarse en el orden adecuado para poder generar perfectamente la estructura deseada. Posterior a ello, se eliminó el archivo Arc2A y el directorio DIR 2AB, además de haber creado una copia de DIR 2AA y reemplazar su nombre por el de Dir X. La implementación de los comandos y su orden puede consultarse en la sección de *Resultados.*

B. GitHub

En GitHub también es esencial conocer la estructura del repositorio para poder empezar a utilizarlo. Primeramente, se debe de crear una cuenta de GitHub siguiendo el manual proporcionado por el mismo GitHub para, con esta, crear un repositorio dentro de su cuenta con el nombre de "PM_Practica_1".

Posterior a ello, se deberán de colocar los archivos desde la PC hacia el repositorio en línea. La estructura en la que opera GitHub, en materia de gestión de archivos, es la siguiente. En GitHub existen tres estados del archivo: el archivo editable, el archivo guardado y listo para envío y el archivo guardado en el repositorio en línea. Cada vez que se desea agregar al repositorio un archivo, deben de seguirse esos tres estados, en ese específico orden, para poder subir los directorios a GitHub.

Una vez que se han anexado los directorios, debe de añadirse un archivo de texto correspondiente con el nombre del equipo y los integrantes que lo componen, creando el archivo desde la PC y anexando únicamente las claves únicas por medio de la consola.

El orden de los comandos y su sintaxis puede encontrarse dentro de la sección de *Resultados.*

C. Arduino

Lo primero que se realizó en esta sección de la práctica fue el descargar el código preestablecido en Arduino llamado Blink, para posteriormente ser cargado a la tarjeta, la cual estaba conectada en un circuito como indicaba la Figura 3 del documento de requerimientos de la práctica.

Con esto, se pudo ver que después de un tiempo determinado, el LED del circuito construido, se prendía y apagaba según las instrucciones del código. Se modificaron estos parámetros para cambiar la frecuencia de oscilación del diodo y se observaron los resultados obtenidos.

Posteriormente, con el uso de un sensor fotoresistivo, se pudo aplicar el código.

Para poder realizar el trabajo requerido, primeramente, se determinaron los valores de los Pins a utilizar y sus configuraciones, ya fueran como inputs o outputs, y especialmente, el Serial para poder hacer una carga funcional del código. A continuación, lo que se realizó fue la determinación del encendido y apagado del LED con respecto a lo sensado por la fotoresistencia. Después de unos cuantos intentos, al cargar el código, se pudo observar que, al cambiar la cantidad de luz, el LED se prendía o apagaba con respecto a lo indicado en el software.

Una vez que se pudo comprobar esto, continuamos con la siguiente parte de la práctica, la cual indicaba que se requería la

conexión de un potenciómetro al circuito. Se cargó el siguiente código a la tarjeta de Arduino:

Código 1. Sección primera de Arduino.

```
// Pin donde se conecta el led
int pinLed = 13;
// Pin analogico de entrada para el resistor fotosensible

// Variable donde se almacena el valor del LDR
int sensor = A0;
int potenciometro = A1;
int valorLDR;
int valorPot;
int perSen;
int perPot;

void setup()
{
 // Configuramos como salidas los pines donde se conectan los led
 pinMode(sensor, INPUT);
 pinMode(potenciometro, INPUT);
 pinMode(pinLed, OUTPUT);
 Serial.begin(9600);
}

void loop()
{
 // Apagar todos los leds siempre que se inicia el ciclo
 digitalWrite(pinLed, LOW);

 // Guardamos el valor leido del ADC en una variable
 // El valor leido por el ADC (voltaje) aumenta de manera directamente proporcional
 // con respecto a la luz percibida por el LDR
 valorLDR= analogRead(sensor);
 valorPot= analogRead(potenciometro);
 perSen = map(valorLDR,0,1023,0,100);
 perPot = map(valorPot,0,1023,0,100);

 // Devolver el valor leido a nuestro monitor serial en el IDE de Arduino
 Serial.println(valorLDR);

 // Encender los leds apropiados de acuerdo al valor de ADC
 if(perSen > perPot)
 {
  digitalWrite(pinLed, HIGH);
 }
 else{
  digitalWrite(pinLed, LOW);
 }
 delay(1000);
}
```

Podemos ver que el código es prácticamente el mismo para esta sección de la práctica, pero para el inciso final, se utilizó el valor de entrada recibido del potenciómetro para determinar el comportamiento del LED según la información recibida del sensor fotoresistivo. Cuando el valor otorgado por el potenciómetro usado de 1 kOhm, era el máximo, se observaba que el LED se prendía cuando el sensor era cubierto, mientras que cuando el valor dado por el potenciómetro era el mínimo, se le debía aplicar una fuente de luz al sensor para que el LED se prendiera.

III. RESULTADOS

Presentamos tres resultados diferentes, con motivo de los tres apartados de la sección II sobre el Desarrollo.

A. Linux

A continuación, se anexa la implementación y resultados de la terminal de Linux, para que, una vez que se leyó el listado de comandos y sus definiciones, pueda comprenderse el resultado aquí anexado.

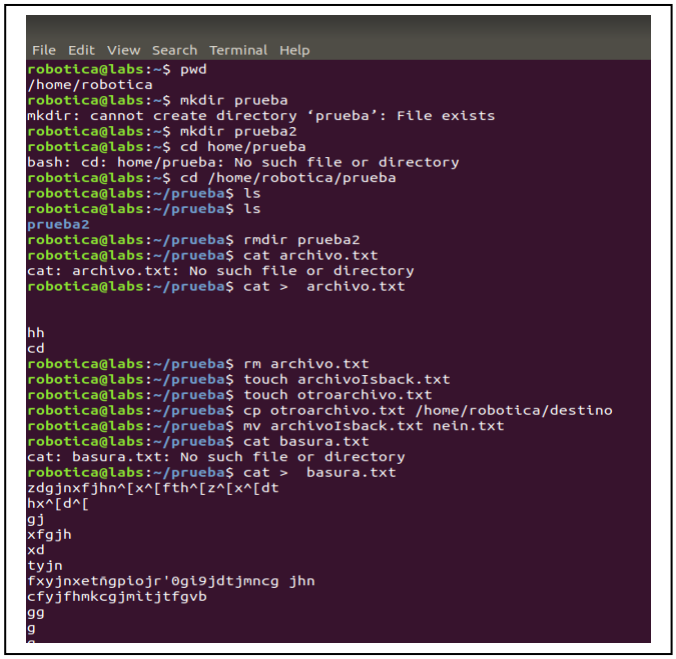

Figura 3. Comandos de Linux para implementar los directorios.

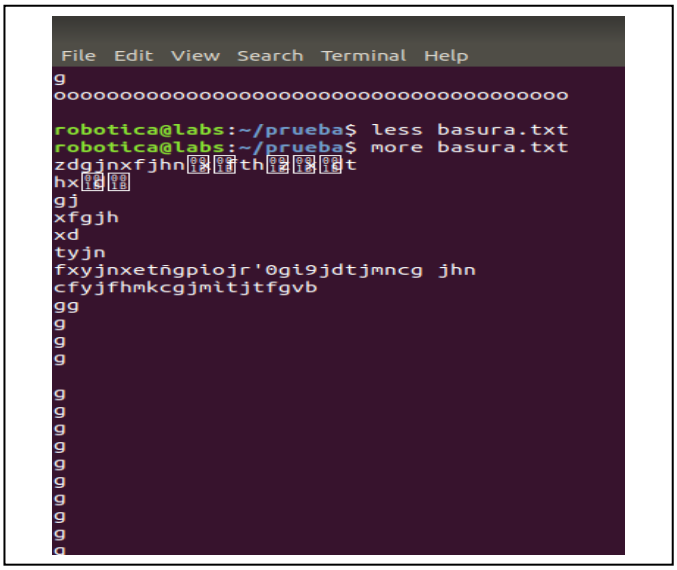

Figura 4. Comandos de Linux para implementar los directorios.

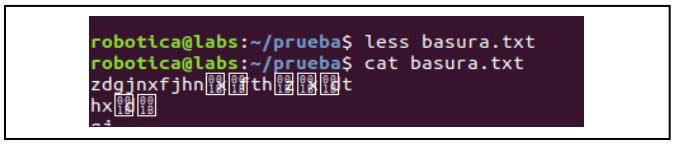

Figura 5. Comandos de Linux para implementar los archivos.

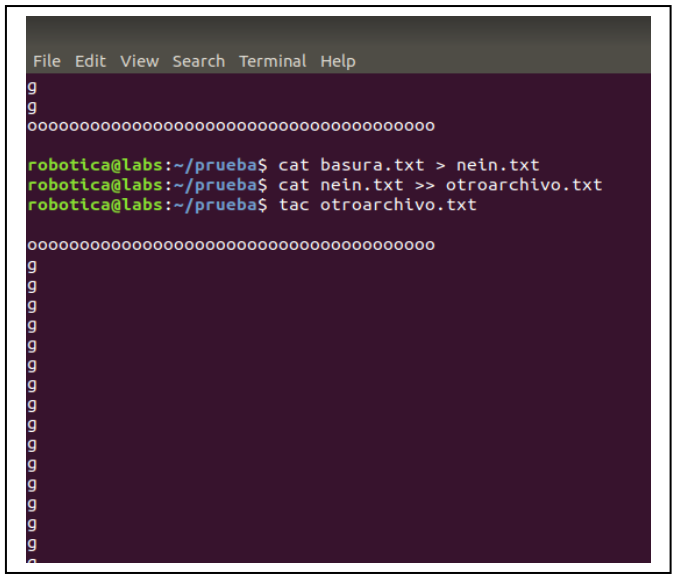

Figura 6. Comandos de Linux para implementar los archivos.

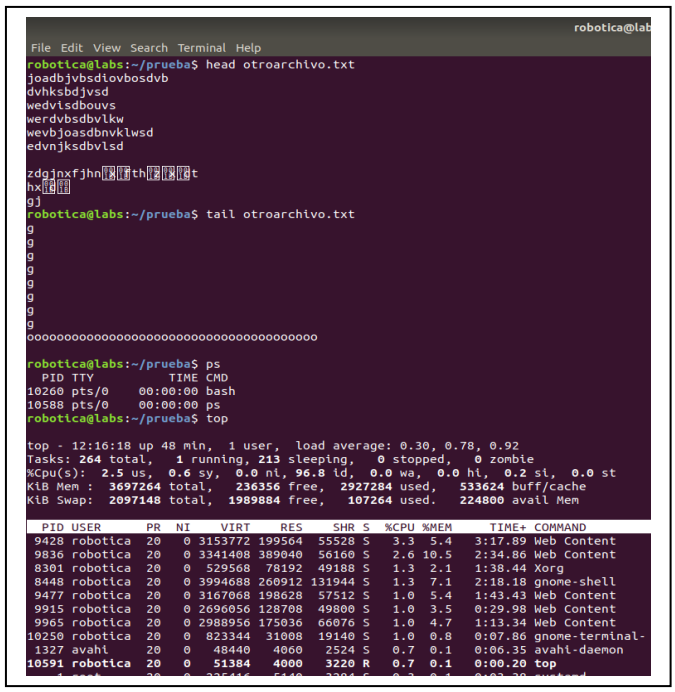

Figura 7. Comandos de Linux para implementar los archivos.

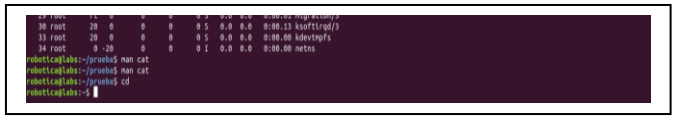

Figura 8. Comandos de Linux para eliminar los archivos.

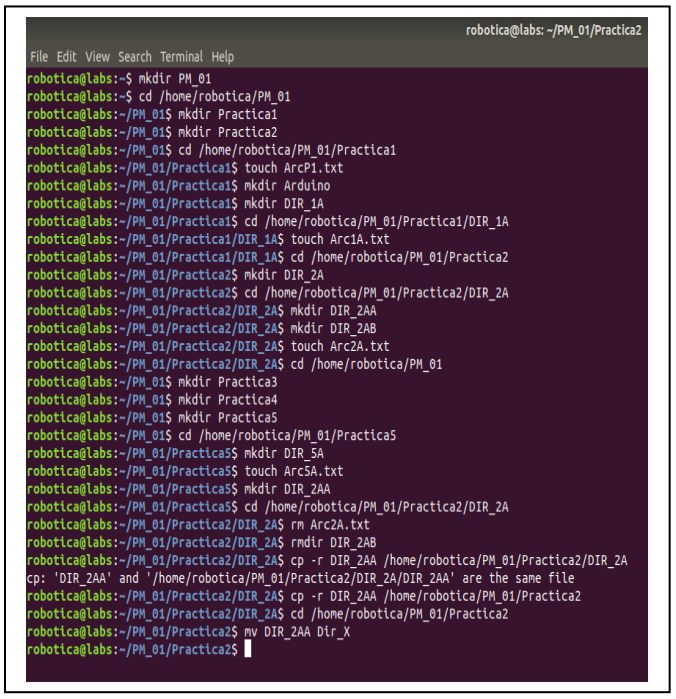

Figura 9. Últimos pasos de la práctica de Linux.

B. GitHub

Las imágenes anexadas a continuación muestran la implementación y resultado de los comandos ejecutados por medio de la consola de Linux (emulada dentro de una computadora Windows) para obtener la estructura de archivos deseada.

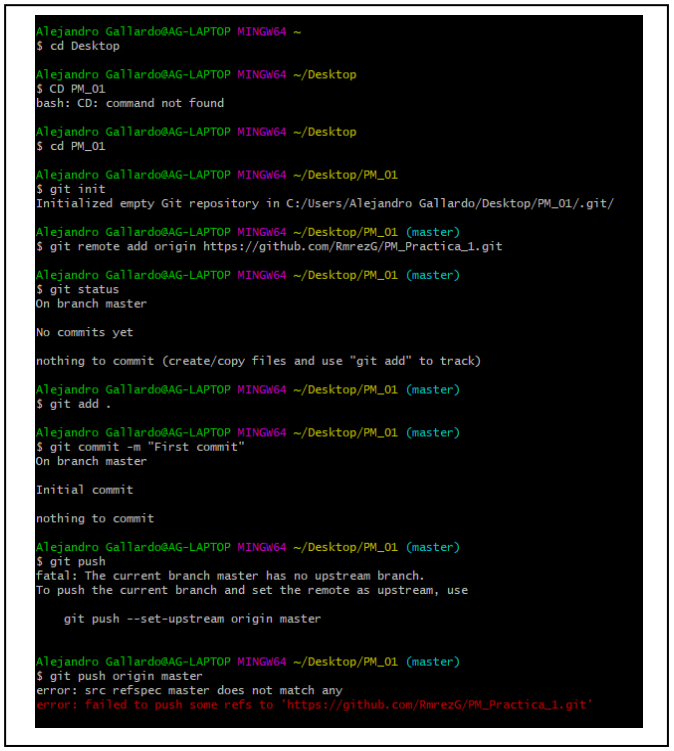

Figura 10. Enlazamiento de directorios con GitHub.

```
Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_01 (master)
$ cd ..

Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop
$ cd PM_Practica_1

Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_Practica_1
$ git init
Initialized empty Git repository in C:/Users/Alejandro Gallardo/Desktop/PM_Practica_1/.git

Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_Practica_1 (master)
$ git remote add origin egit add README.md
cho "# PM_Practica_1" >> README.md
git commit -m "first commit"
git remote add origin https://github.com/RmrezG/PM_Practica_1.git
git push -u origin master
usage: git remote add [<options>] <name> <url>

    -f, --fetch           fetch the remote branches
    --tags                import all tags and associated objects when fetching
                          or do not fetch any tag at all (--no-tags)
    -t, --track <branch>  branch(es) to track
    -m, --master <branch>
                          master branch
    --mirror[=(push|fetch)]
                          set up remote as a mirror to push to or fetch from


Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_Practica_1 (master)
$ git init
Reinitialized existing Git repository in C:/Users/Alejandro Gallardo/Desktop/PM_Practica_1

Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_Practica_1 (master)
$ git add README.md

Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_Practica_1 (master)
$ git commit -m "first commit"
[master (root-commit) 4c6833a] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md

Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_Practica_1 (master)
$ git remote add origin https://github.com/RmrezG/PM_Practica_1.git

Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_Practica_1 (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 225 bytes | 225.00 KiB/s, done.
```

Figura 11. Enlazamiento de directorios con GitHub.

```
Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_Practica_1 (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 225 bytes | 225.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/RmrezG/PM_Practica_1.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_Practica_1 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Practica1/

nothing added to commit but untracked files present (use "git add" to track)

Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_Practica_1 (master)
$ git add .

Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_Practica_1 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Practica1/ArcP1.txt


Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_Practica_1 (master)
$ git commit -m "Commit Práctica 1"
[master 2d019e2] Commit Práctica 1
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Practica1/ArcP1.txt

Alejandro Gallardo@AG-LAPTOP MINGW64 ~/Desktop/PM_Practica_1 (master)
```

Figura 12. Enlazamiento de directorios con GitHub.

```
  GNU nano 4.6
Nombre del equipo: Kung-Flu

Integrantes:
--------------------------------
Jorge Alejandro Ramírez Gallardo
Ricardo
--------------------------------

Clave única:
--------------------------------
171808
170402
```

Figura 13. Edición del texto.

C. Arduino

Por último, se buscó tener un acercamiento a la plataforma Arduino, con una introducción a la forma de programar y las funcionalidades que puede otorgar y su relevancia para la ingeniería, apoyados con distintos componentes eléctricos. El código se ubica dentro de la sección de *Desarrollo*. Los resultados e implementación del circuito se muestran a continuación.

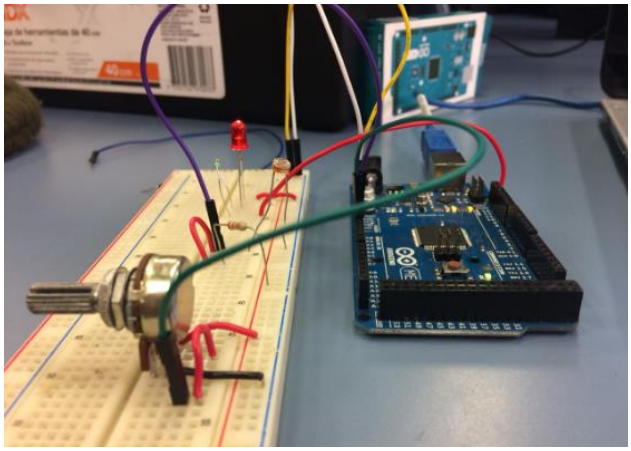

Figura 14. Implementación del apartado C - Arduino.

IV. Conclusiones

En esta práctica inicialmente se vio el manejo de la consola de Linux y los comandos usados para manejar un repositorio de GitHub, actividades que, si bien no serán usados para programar directamente el funcionamiento del sistema mecatrónico, presentan una gran ayuda para mantener en orden los archivos usados mediante comandos repetibles y almacenar el progreso desarrollado en el curso de la materia en un repositorio en línea.

De aquí, la segunda parte de la práctica, los ejercicios realizados con el Arduino, aún siendo bastante simples, muestran aspectos fundamentales en la mecatrónica. Particularmente, se puede apreciar el manejo de entradas y salidas de información en el dispositivo Arduino, y la manera en que se manejan las entradas de información de tipo analógico para interpretarse como información digital (usando en concreto el potenciómetro). Tales consideraciones, entonces,

resultan fundamentales como base para proseguir correctamente con el resto del curso, al basarse la mecatrónica en gran parte en una conjunción entre el funcionamiento mecánico y electrónica, que es permitida gracias a tal manejo de entradas y salidas.

AGRADECIMIENTOS

Queremos agradecer a nuestros profesores, el Dr. José Guadalupe y al maestro Benito Granados Rojas por el tiempo dedicado a nosotros y por su ayuda en la aplicación de los conocimientos obtenidos en clase.

REFERENCIAS

[1] EcuRed. Disponible en: https://www.ecured.cu/Pwd
[2] Ayuda Linux. Disponible en: https://ayudalinux.com/como-usar-el-comando-mkdir/
[3] Francisconi.org. Soluciones Informáticas. Disponible en: https://francisconi.org/linux/comandos/cd
[4] Rip Tutorial. Disponible en: https://riptutorial.com/es/linux/topic/5956/ls-comando
[5] Ayuda Linux. Disponible en: https://ayudalinux.com/como-usar-el-comando-mkdir/
[6] Cámbiatealinux.com. Disponible en: https://www.cambiatealinux.com/rm-borrar-ficheros-directorio
[7] Cámbiatealinux.com. Disponible en: https://www.cambiatealinux.com/touch-crear-o-actualizar-ficheros
[8] Como instalar Linux. Disponible en: https://www.comoinstalarlinux.com/comandos-linux-cp/
[9] Ayuda Linux. Disponible en: https://ayudalinux.com/como-usar-el-comando-mv/
[10] Linuxize. Disponible en: https://linuxize.com/post/less-command-in-linux/
[11] Geeks for geeks. Disponible en: https://www.geeksforgeeks.org/more-cohmmand-in-linux-with-examples/
[12] Geeks for geeks. Disponible en: https://www.geeksforgeeks.org/cat-command-in-linux-with-examples/
[13] Computer Hope. Disponible en: https://www.computerhope.com/unix/uhead.htm
[14] Computer Hope. Disponible en: https://www.computerhope.com/unix/utail.htm
[15] Geeks for geeks. Disponible en: https://www.geeksforgeeks.org/ps-command-in-linux-with-examples/
[16] Tecmint. Disponible en: https://www.tecmint.com/12-top-command-examples-in-linux/
[17] Geeks for geeks. Disponible en: https://www.geeksforgeeks.org/man-command-in-linux-with-examples/