

**UNIVERSIDAD MAYOR DE SAN ANDRÉS  
FACULTAD DE INGENIERÍA  
CARRERA DE INGENIERÍA ELECTRÓNICA**



**Trabajo Dirigido**

**“Diseño y Construcción de un Aparato de Laboratorio Micro  
Computarizado para Desarrollo Experimental en Laboratorios  
de Física y Control”**

**Postulante: Mariana Bianca Lino Sánchez**

**Asesor: Ing. Marcelo Ramírez Molina**

**La Paz, Abril 2019.**



**UNIVERSIDAD MAYOR DE SAN ANDRÉS  
FACULTAD DE INGENIERIA**



**LA FACULTAD DE INGENIERIA DE LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.**

**LICENCIA DE USO**

El usuario está autorizado a:

- a) Visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
- b) Copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
- c) Copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la cita o referencia correspondiente en apego a las normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

**TODOS LOS DERECHOS RESERVADOS. EL USO NO AUTORIZADO DE LOS CONTENIDOS PUBLICADOS EN ESTE SITIO DERIVARA EN EL INICIO DE ACCIONES LEGALES CONTEMPLADAS EN LA LEY DE DERECHOS DE AUTOR.**

# Índice General

Índice General.....	1
Índice de Figuras.....	4
Índice de Tablas.....	8
<b>Parte 1.....</b>	<b>10</b>
<b>1. Resumen de la actividad laboral.....</b>	<b>10</b>
<b>1.1 Primer trabajo o empleo.....</b>	<b>10</b>
<b>1.1.1 Organización.....</b>	<b>10</b>
<b>1.1.2 Posición.....</b>	<b>10</b>
<b>1.2 Segundo trabajo o empleo.....</b>	<b>11</b>
<b>1.2.1 Organización.....</b>	<b>11</b>
<b>1.2.2 Posición.....</b>	<b>11</b>
<b>1.3 Tercer trabajo o empleo.....</b>	<b>11</b>
<b>1.3.1 Organización.....</b>	<b>11</b>
<b>1.3.2 Posición.....</b>	<b>11</b>
<b>1.4 Cuarto trabajo o empleo.....</b>	<b>11</b>
<b>1.4.1 Organización.....</b>	<b>11</b>
<b>1.4.2 Posición.....</b>	<b>11</b>
<b>1.5 Quinto trabajo o empleo.....</b>	<b>12</b>
<b>1.5.1 Organización.....</b>	<b>12</b>
<b>1.5.2 Posición.....</b>	<b>12</b>
<b>1.6 Sexto trabajo o empleo.....</b>	<b>12</b>
<b>1.6.1 Organización.....</b>	<b>12</b>
<b>1.6.2 Posición.....</b>	<b>12</b>
<b>1.7 Séptimo trabajo o empleo.....</b>	<b>12</b>
<b>1.7.1 Organización.....</b>	<b>12</b>

<b>1.7.2 Posición</b> .....	12
<b>1.8 Actual trabajo o empleo</b> .....	13
<b>1.8.1 Organización</b> .....	13
<b>1.8.2 Posición</b> .....	13
<b>Parte II</b> .....	14
<b>Capítulo 1</b> .....	14
<b>1. Introducción</b> .....	14
<b>1.1. Objetivos</b> .....	14
a.    Objetivo General.....	14
b.    Objetivos Específicos.....	14
<b>1.2. Justificación</b> .....	15
<b>1.3. Alcances y Limitaciones</b> .....	15
a.    Limitaciones.....	15
b.    Alcances.....	15
<b>2. Marco Teórico</b> .....	17
<b>2.1. Conceptos de Físicos</b> .....	17
<b>2.2. Arduino</b> .....	17
2.2.1. Generalidades.....	18
2.2.2. Entorno de Desarrollo.....	18
2.2.3. MEGA 2560.....	20
<b>2.3. LabVEW</b> .....	21
2.3.1. Generalidades.....	21
2.3.2. Sensores y Actuadores.....	22
a. Motor Paso a Paso.....	22
• Unipolares.....	22
b. ULN 2803.....	23
c. Finales de Carrera.....	24
d. Diodo Láser.....	25
e. Fotorresistor.....	26

<b>3. Desarrollo del Proyecto</b> .....	27
<b>3.1. Diseño y Funcionamiento del Aparato de Laboratorio</b> .....	27
3.1.1. Geometría del Aparato.....	27
a. Torre de Elevación.....	28
b. Plataforma móvil.....	30
c. Base Fija.....	31
<b>3.2. Diseño de Circuitos de Control y Potencia</b> .....	32
3.2.1. Diseño de Conmutadores Láser.....	32
3.2.2. Configuración Eléctrica del Motor Paso a Paso.....	35
3.2.3. Finales de Carrera.....	36
3.2.4. Conexión a Arduino.....	37
<b>3.3. Programación en Arduino</b> .....	40
3.3.1. Configuraciones Iniciales y Variables Globales.....	40
3.3.2. Configuraciones Previas al Programa General.....	41
3.3.3 Funciones.....	42
a. Función – serialA().....	42
b. Función – reset(int j).....	43
c. Función – AnglePrincipal(float ang).....	44
d. Función - sensores(int g).....	50
e. Función - ValAngulo(int a).....	55
f. Función – Lab3(int h).....	59
3.3.4 Programa Principal.....	62
a. Proceso, procedimiento para Laboratorio 1.....	63
b. Proceso, procedimiento para Laboratorio 2.....	65
c. Proceso, procedimiento para Laboratorio 3.....	66
<b>3.4. Programación en LabView</b> .....	68
3.4.1. Laboratorio 1.....	68

3.4.2. Laboratorio 2.....	71
3.4.3. Laboratorio 3.....	73
<b>3.5. Pruebas y resultados.....</b>	<b>76</b>
a. Laboratorio 1. Determinación de aceleración en un plano inclinado sin fricción.....	76
b. Laboratorio 2. Determinación del coeficiente de fricción estática.....	77
c. Laboratorio 3. Determinación del coeficiente de fricción Dinámica.....	78
<b>4. Conclusiones y Recomendaciones.....</b>	<b>80</b>
<b>4.1. Conclusiones.....</b>	<b>80</b>
<b>4.2. Recomendaciones.....</b>	<b>80</b>
<b>Parte III</b>	
<b>1. Análisis de la Actividad.....</b>	<b>82</b>
1.1 Desempeño laboral.....	82
1.2 Formación recibida en la UMSA.....	82
<b>Bibliografía.....</b>	<b>83</b>
<b>ANEXOS.....</b>	<b>84</b>

# Índice de Figuras

2.1. IDE Arduino – Plataforma de Programación.....	19
2.2. Placa – Arduino MEGA 2560.....	20
2.3. Motor Paso a Paso Unipolar.....	23
2.4. ULN 2803.....	24
2.5. Tipos de Michoswitch.....	25
2.6. Curva del Diodo Láser.....	26
2.7. Fotorresistor LDR.....	26
3.1. Prototipo Completo.....	28
3.2. Torre de Elevación.....	29
3.3. Plataforma Móvil.....	30
3.4. Base Fija.....	31
3.5. Circuito del Conmutador Láser.....	32
3.6. Circuito del Conmutador Láser en Saturación y Corte.....	33
3.7. Circuito Diseñado del Conmutador Láser.....	35
3.8. Conexión ULN 2803 a Motor Paso a Paso.....	36
3.9. Conexión de Finales de Carrera.....	37
3.10. Conexión de fotorresistores.....	38
3.11. Conexión de Finales de Carrera.....	38
3.12. Conexión de Motor paso a paso.....	39
3.13. Conexión completa - Arduino.....	39
3.14. Configuración Arduino – Variables Globales.....	40
3.15. Configuración en programación.....	40
3.16. Configuración Void Setup.....	42
3.17. Función “serialA.....	42
3.18. Diagrama de Flujo – Función “reset” .....	43

<b>3.19.</b> Promedio y Factores .....	45
<b>3.20.</b> Diagrama de Flujo – Función “angle(long x, long act)” .....	48
<b>3.21.</b> Diagrama de Flujo – Función “anglePrincipal(float ang)” .....	49
<b>3.22.</b> Variables locales “anglePrincipal(float ang)” .....	50
<b>3.23.</b> Diagrama de flujo – Función “Sensores(int g) – Primer conmutador.....	51
<b>3.24.</b> Diagrama de flujo – Función “Sensores(int g) – Segundo conmutador.....	52
<b>3.25.</b> Diagrama de flujo – Función “Sensores(int g) – Último conmutador.....	53
<b>3.26.</b> Función “Sensores(int g) – Envío de datos por el puerto Serial.....	53
<b>3.27.</b> Diagrama de flujo – Función “Sensores(int g) .....	54
<b>3.28.</b> Diagrama de flujo – Función “ValAngulo(int a)” .....	56
<b>3.29.</b> Diagrama de flujo – Función “ValAngulo(int a)” - Segunda parte.....	58
<b>3.30.</b> Diagrama de flujo – Función “Lab3(int h)” – Conmutador 5.....	59
<b>3.31.</b> Función “Lab3(int h) – Envío de datos por el puerto Serial.....	60
<b>3.32.</b> Diagrama de flujo – Función “Lab3(int h)” .....	61
<b>3.33.</b> Diagrama de flujo – Función Principal - General.....	62
<b>3.34.</b> Diagrama de flujo – Función Principal – Laboratorio 1.....	64
<b>3.35.</b> Diagrama de flujo – Función Principal – Laboratorio 2.....	65
<b>3.36.</b> Diagrama de flujo – Función Principal – Laboratorio 3.....	67
<b>3.37.</b> Diagrama de Bloques – Laboratorio 1.....	68
<b>3.38.</b> Diagrama de Bloques – Laboratorio 1.....	69
<b>3.39.</b> Panel Frontal – Laboratorio 1.....	70
<b>3.40.</b> Diagrama de Bloques – Laboratorio 2.....	71
<b>3.41.</b> Panel Frontal – Laboratorio 2.....	72
<b>3.42.</b> Diagrama de Bloques – Laboratorio 3.....	73
<b>3.43.</b> Diagrama de Bloques – Laboratorio 3.....	74
<b>3.44.</b> Panel Frontal – Laboratorio 3.....	75
<b>A.1.</b> Maqueta Completa - Construcción terminada (Vista Lateral) .....	84
<b>A.2.</b> Maqueta Completa - Construcción terminada (Sin Sensores).....	84



<b>A.3.</b> Maqueta Completa - Construcción terminada (Vista Superior Lateral).....	85
<b>A.4.</b> Maqueta Completa - Construcción terminada (Vista Superior Frontal).....	85
<b>A.5</b> Maqueta Completa – Construcción Terminada sin Sensores .....	86
<b>A.6.</b> Torre Principal – Construcción.....	86
<b>A.7.</b> Maqueta Completa - Sensores Láser Activos.....	87
<b>A.8.</b> Maqueta Completa - Sensores Láser Activos (Vista de Torre) .....	87
<b>A.9.</b> Torre Principal.....	88
<b>B.1.</b> Bosquejo – Bloque sobre una pendiente sin fricción.....	90
<b>B.2.</b> Coeficiente de fricción estática-Bloque en un plano inclinado .....	91
<b>B.3.</b> Coeficiente de fricción dinámica – Dos bloques conectados por una cuerda.....	92
<b>F.1.</b> Panel Frontal / Diagrama de Bloques.....	106
<b>F.2.</b> Paleta de Funciones y Paleta de Controles.....	107
<b>F.3.</b> Estructura WHILE.....	108
<b>F.4.</b> Estructura EVENT.....	109
<b>F.5.</b> Estructura CASE.....	109
<b>F.6.</b> Estructura FLAT SEQUENCE.....	110
<b>F.7.</b> Terminales LabView.....	112
<b>F.8.</b> Build Array.....	113
<b>F.9.</b> Insertar a Array.....	113
<b>F.10.</b> Función Transpuesta.....	113
<b>F.11.</b> Función Empaquetado.....	114
<b>F.12.</b> Función Aritmética Compuesta.....	114
<b>F.13.</b> Función Esperar Siguiente ms.....	114
<b>F.14.</b> Función Esperar ms.....	115
<b>F.15.</b> Función de Emparejamiento.....	115
<b>F.16.</b> Número a Cadena Fraccional.....	115
<b>F.17.</b> Número a cadena Decimal.....	116
<b>F.18.</b> Cadena Fraccional a Número.....	116

<b>F.19.</b> Nodo de Expresión – Grados a Radianes.....	116
<b>F.20.</b> Funciones ‘Sen’ y ‘Cos’.....	117
<b>F.21.</b> Función Ajuste de Curvas.....	117
<b>F.22.</b> Menú Propio de la Función Ajuste de Curvas.....	117
<b>F.23.</b> Configuración de puerto Serial VISA.....	118
<b>F.24.</b> Escritura en VISA.....	119
<b>F.25.</b> Lectura en VISA.....	120
<b>F.26.</b> Bytes en el Puerto.....	120
<b>F.27.</b> Cerrar Sesión VISA.....	120

# Índice de Tablas

<b>3.1.</b> Secuencia para Motor Paso a Paso.....	36
<b>3.2.</b> Pruebas, relación entre Número de pasos y ángulos.....	44
<b>3.3.</b> Promedio y Factores .....	45
<b>3.4.</b> Fórmulas de paso por intervalo.....	47
<b>3.5.</b> Fórmula de ángulo por intervalo.....	57
<b>3.6.</b> Pruebas experimentales – Aceleración en un plano inclinado sin rozamiento.....	76
<b>3.7.</b> Obtención de valores teóricos – Aceleración sobre un plano inclinado sin rozamiento.....	77
<b>3.8.</b> Errores – Aceleración sobre un plano inclinado sin rozamiento.....	77
<b>3.9.</b> Pruebas experimentales – Coeficiente de fricción estática.....	78
<b>3.10.</b> Pruebas experimentales – Coeficiente de fricción dinámica.....	78
<b>B.1.</b> Coeficientes de fricción ( $\mu_s$ estático, $\mu_k$ dinámico) .....	89
<b>F.1.</b> Tipos de datos más comúnmente utilizados en LabVIEW.....	111

# Parte 1

## Introducción

### 1. Resumen de la actividad laboral.

Inicia mi etapa laboral en el área de sistemas, dando soporte, programación y supervisión en algunas empresas de telecomunicaciones que, a su vez, logran que adquiriera conocimiento y experiencia en cableado estructurado.

En el área de sistemas de control, inicio una segunda práctica de pasantía en el área de mantenimiento y control en industria donde adquiero, además experiencia en instrumentación, automatización y seguridad industrial. Gracias a esto, consigo una posición en una empresa de automatización en el área de proyectos y ventas de equipos y servicios en automatización, instrumentación y control. Finalmente, gracias a la experiencia en seguridad industrial y especialidad en equipos eléctricos me encuentro en una posición de soporte en supervisión a nivel nacional en industria y a cargo de PSM Electricid (Riesgo eléctrico), seguridad y medio ambiente.

#### 1.1 Primer Trabajo o Empleo

##### 1.1.1. Organización

Nuevatel- Viva, es una operadora de telecomunicaciones de Bolivia. Fue fundada en 1999. En la actualidad se encuentra entre las mayores empresas del Bolivia. Tiene cobertura de telefonía celular en todo el país.

##### 1.1.2 Posición

Inicia la etapa laboral, en el área de sistemas en la empresa de telecomunicaciones Nuevatel, VIVA como pasantía para luego pasar a un puesto de consultoría temporal donde me desempeño dando soporte en sistemas a personal de la empresa además de la elaboración de una plataforma programada y lograr obtener una base de datos del software instalado en cada equipo de la compañía s nivel oficina central.

## **1.2 Segundo Trabajo o Empleo**

### **1.2.1 Organización**

Skyframe SRL. una empresa multidisciplinaria de ingeniería en tecnologías de la información y comunicación. Provee productos y servicios orientados a la infraestructura tecnológica, transmisión de datos, voz y multimedia, integración de sistemas informáticos, sistemas de monitoreo y alarma, seguridad de redes de comunicación, provisión de equipamiento y consultoría en tecnología.

### **1.2.2 Posición**

Como consultor externo, tengo a mi cargo el cableado y puesta en marcha de un sistema de conexiones generales de red estructurada. Empresa Skyframe SRL.

## **1.3 Tercer Trabajo o Empleo**

### **1.3.1 Organización**

SOLATEC Bolivia es una empresa que provee servicios de instalación de equipos de vigilancia y seguridad. Instalación de sistemas de alimentación ininterrumpida UPS y generadores.

### **1.3.2 Posición**

Como consultor externo, tengo a mi cargo el diseño y conexionado de sistemas de sistemas de seguridad y vigilancia en diversas oficinas además de la puesta en marcha.

## **1.4 Cuarto Trabajo o Empleo**

### **1.4.1 Organización**

TELCOMBOL, empresa especializada en cableado estructurado para redes de área local e instalación de fibra óptica.

### **1.4.2 Posición**

Como consultor externo, realicé el diseño de los planos del cableado estructurado y supervisión de la ejecución y puesta en marcha según requerimiento. A mi cargo tuve un equipo de cinco personas encargadas de realizar la tarea de acuerdo a planificación.

## **1.5 Quinto Trabajo o Empleo**

### **1.5.1 Organización**

PEPSI Bolivia, franquicia comprada por la argentina Quilmes Industrial, también dueña de la reconocida Cervecería Boliviana Nacional, en el año 2009 ubicada en la ciudad del El Alto, zona Río Seco.

### **1.5.2 Posición**

Pasantía en el área de Mantenimiento y control en industria. Me encargaba del mantenimiento preventivo y correctivo en línea de envasado, elaboración y servicios.

## **1.6 Sexto Trabajo o Empleo**

### **1.6.1 Organización**

Electricidad Illimani, empresa especializada en instrumentación y cableado a nivel industrial.

### **1.6.2 Posición**

Como supervisor e instrumentista para el proyecto “PTA – CBN” (Plata de tratamiento de aguas dentro de CBN), me encargué de la instalación, cableado y supervisión en el área de seguridad industrial y puesta en marcha del montaje total de la nueva planta de tratamientos de agua de la Cervecería.

## **1.7 Séptimo Trabajo o Empleo**

### **1.7.1 Organización**

Feprom Solutions SRL. , empresa de ingeniería y soluciones tecnológicas en las áreas de automatización, comunicación industrial, instrumentación, sistemas de información así como también en las áreas de transmisión, control y protección en media tensión.

### **1.7.2 Posición**

Como Ingeniero de ventas y proyectos me encargué del relevamiento de proyectos nuevos, servicios o modificaciones en diversas fábricas de acuerdo al requerimiento industrial.

Realicé oferta y venta de equipos de automatización, instrumentación y media tensión de marca SIEMENS además del relevamiento y diseño en servicios de automatización.

## **1.8 Actual Trabajo o Empleo**

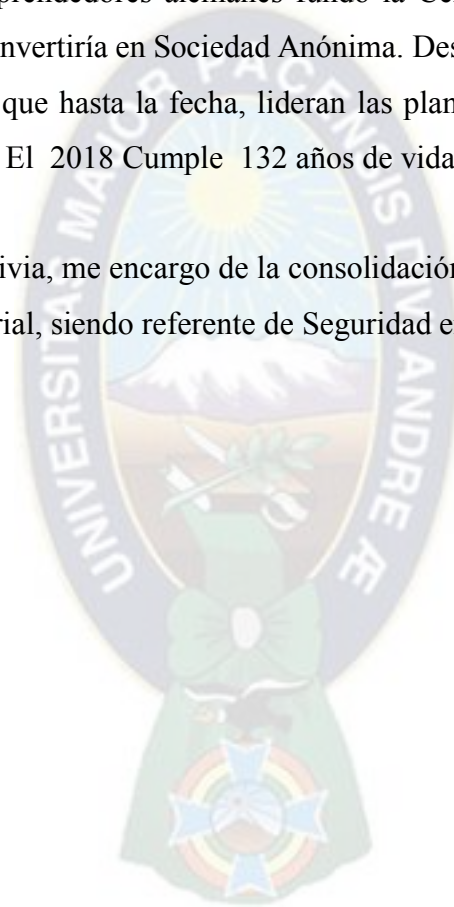
### **1.8.1 Organización**

Cervecería Boliviana Nacional.

En 1886, un grupo de emprendedores alemanes fundó la Cervecería Boliviana Nacional, empresa que en 1920, se convertiría en Sociedad Anónima. Desde 1986 Comeinza una etapa de innovación tecnológica que hasta la fecha, lideran las plantas de La Paz, Cochabamba, Huari, Santa Cruz y Tarija. El 2018 Cumple 132 años de vida.

### **1.8.2 Posición**

Como Analista HSMA Bolivia, me encargo de la consolidación de datos a nivel Bolivia en el área de seguridad industrial, siendo referente de Seguridad en trabajos eléctricos y medio ambiente.



# Parte II

## Capítulo 1

### Introducción

#### 1.1 Objetivos

##### a. Objetivo General

Diseño y construcción de un aparato de laboratorio micro computarizado que consiste en un plano inclinado reconfigurable mediante subsistemas electrónicos de control de actuadores, sensores de posición e interfaz a usuario mediante computadora para experimentación en laboratorios de física básica y en ingeniería de control de la Facultad de Ingeniería de la Universidad Mayor de San Andrés.

##### b. Objetivos Específicos

- Diseño y construcción de la mecánica del plano inclinado para determinar los coeficientes de rozamiento estático, dinámico y aceleración de una masa en un plano inclinado con la posibilidad de cambiar el tipo de superficie y así calcular diferentes coeficientes de rozamiento para poder realizar comparaciones con datos teóricos.
- Diseño y montaje de la electrónica.
- Desarrollo de programas de control de actuadores y sensores en un microcontrolador Arduino MEGA 2560.
- Implementación de interfaz de usuario en el programa LabVIEW.
- Integración de todos los subsistemas.



- Desarrollo y demostración de ejercicios experimentales.

## **1.2 Justificación**

En los laboratorios de Física Básica de la Facultad de Ingeniería se cuenta con equipos obsoletos para la determinación de los coeficientes de fricción dinámicos y estáticos, así como para el estudio de la cinemática y dinámica. Asimismo, la metodología de enseñanza no emplea tecnologías actuales. Por ello se vio la necesidad de actualizar estos equipos de laboratorio para que los estudiantes obtengan datos más precisos, experimentos más variados y un contacto con la disciplina de la electrónica para estudiantes de primer año de ingeniería.

El laboratorio de control de la Carrera de Ingeniería Electrónica cuenta con diversos equipos para la experimentación en sistemas de control. El aparato propuesto es un aporte al laboratorio y dar la posibilidad a los estudiantes llevar a cabo experimentos con un nuevo tipo de planta.

## **Alcances y Limitaciones**

### **a. Limitaciones**

- El aparato cuenta con 8 sensores tipo láser para hallar la aceleración de manera experimental.
- La distancia entre cada par de sensores es fija.
- No se guarda la información obtenida experimentalmente en una base de datos, se muestran temporalmente mientras dure el laboratorio.
- El equipo logra inclinar la plataforma hasta un máximo de 34 grados.

### **b. Alcances**

- El proyecto incluye el diseño y construcción del aparato de laboratorio micro computarizado.

- El proyecto incluye el diseño e implementación de los programas de control y adquisición de datos.
- El proyecto incluye el diseño y programación de la interfaz de usuario en computadora mediante LabView, para tres diferentes laboratorios.
- El proyecto incluye tres ejercicios experimentales demostrativos.



## Capítulo 2

### Marco Teórico

#### 2.1. Conceptos Físicos

Para el desarrollo de los algoritmos de cálculo se emplean los conceptos de:

- Leyes del Movimiento
- Fuerza
- Masa
- Primera Ley de Newton
- Segunda Ley de Newton
- Tercera Ley de Newton
- Fuerzas de Fricción

#### 2.2. Arduino

Arduino es una plataforma de código y hardware abierto u Open Source. Se puede acceder a todo aspecto del funcionamiento circuital y algorítmico de las placas.

En cuestión de costos, las placas Arduino son más accesibles comparadas con otras plataformas de microcontroladores. No necesita ningún tipo de tarjeta de programación, la misma placa se conecta vía serial a la computadora usando un cable USB y se pueden cargar los programas totalmente en vivo.

El software de Arduino funciona en los sistemas operativos Windows, Macintosh OSX y Linux; mientras que la mayoría de otros entornos para microcontroladores están únicamente limitados a Windows.

La sencilla programación y la vasta cantidad de librerías logran que se pueda trabajar con esta placa de manera más rápida con excelentes resultados

### **2.2.1. Generalidades**

Arduino es una placa de Hardware libre que incorpora un microcontrolador reprogramable y una serie de pines-hembra unidos internamente a las patillas de entrada y salida del microcontrolador en los cuales se conectan de manera sencilla diferentes sensores y actuadores.

Existen varias placas de Arduino, cada una con características diferentes como el tamaño físico, el número de pines-hembra, el modelo de microcontrolador incorporado, la cantidad de memoria utilizable, etc. Los microcontroladores incorporados en las diferentes placas de Arduino pertenecen todos a la familia AVR, una arquitectura de microcontroladores desarrollada y fabricada por ATMEL.

Los proyectos Arduino pueden ser autónomos o no. En el primer caso, una vez programado el microcontrolador, la placa no necesita estar conectada a ningún computador y puede funcionar autónomamente si dispone de alguna fuente de alimentación. En el segundo caso, la placa debe estar conectada de alguna forma permanente (por cable USB, por cable de red Ethernet, etc.) a un computador ejecutando algún software específico que permita la comunicación entre este y la placa y el intercambio de datos entre ambos dispositivos. Este software específico se lo programa mediante algún lenguaje de programación estándar y será independiente completamente del entorno de desarrollo Arduino, el cual no se necesitará más, una vez que la placa ya haya sido programada y esté en funcionamiento.

Todos los tableros Arduino son completamente de código abierto, lo que permite a los usuarios construirlos de forma independiente y eventualmente adaptarlos a sus necesidades particulares.

### **2.2.2. Entorno de Desarrollo**

Las siglas IDE vienen de Integrated Development Environment, lo que traducido significa Entorno de Desarrollo Integrado. El IDE de Arduino es una aplicación multiplataforma escrita en Java derivada del lenguaje Processing y Wiring.

El Arduino IDE viene con una biblioteca llamada Wiring, tiene la capacidad de programar en C/C++. En General, permite escribir y editar programas (Sketch), compilar el programa comprobando la existencia de errores y finalmente grabarlo en la memoria del microcontrolador de la placa Arduino.

Este IDE puede ser descargado de la página web de Arduino de manera gratuita. Este software está publicado como herramienta de código abierto, se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux.

Un programa diseñado para ejecutarse sobre un Arduino (un “sketch”) siempre se compone de cuatro secciones:

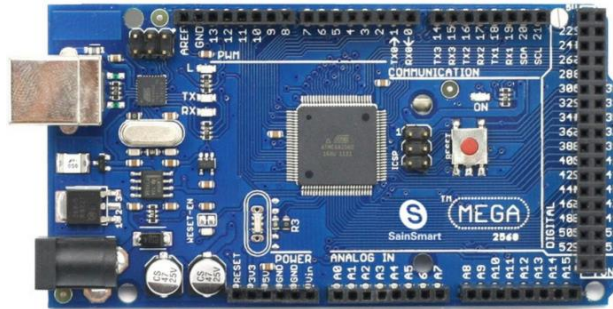
- La sección de declaraciones de variables globales, ubicada directamente al principio del sketch. Está reservada para escribir las diferentes declaraciones de variables que necesitemos o incluir librerías necesarias.
- La sección llamada “void setup()”: se ejecutan una única vez, en el momento de encender o resetear la placa Arduino. Normalmente sirven para realizar ciertas preconfiguraciones iniciales.
- La sección llamada “void loop()”: se ejecutan justo después de las de la sección “void setup()” infinitas veces hasta que la placa se apague o se resetee. El contenido de “void loop()” se ejecuta desde la primera instrucción hasta la última una y otra vez.
- La sección de Funciones: todas van después del “Void loo()”, tomando en cuenta sus respectivas variables locales o variables globales, esto de acuerdo a cada función.



**Fig. 2.1.** IDE Arduino – Plataforma de Programación  
**Fuente:** Elaboración Propia mediante Herramienta IDE de Arduino

### 2.2.3. MEGA 2560

Arduino Mega es una tarjeta de desarrollo open-source construida con un microcontrolador modelo Atmega2560 que posee pines de entradas y salidas (E/S), analógicas y digitales.



**Fig. 2.2.** Placa – Arduino MEGA 2560  
**Fuente:** Elaboración propia - Fotografía

El Arduino Mega tiene 54 pines de entradas/salidas digitales (14 de las cuales pueden ser utilizadas como salidas PWM), 16 entradas analógicas, 4 UARTs (puertos serial por hardware), cristal oscilador de 16MHz, conexión USB, jack de alimentación, conector ICSP y botón de reset. Cada uno de los 54 pines operan a 5 voltios. Cada pin puede proporcionar o recibir 20 mA, un máximo de 40 mA es el valor que no debe superarse para evitar daños permanentes en el microcontrolador.

Además, algunos pines tienen funciones especializadas:

- Serie: 0 (RX) y 1 (TX); Serie 1: 19 (RX) y 18 (TX); Serie 2: 17 (RX) y 16 (TX); Serie 3: 15 (RX) y 14 (TX). Se utiliza para recibir (RX) y transmitir datos serie (TX) TTL.
- Interrupciones externas: 2 (interrupción 0), 3 (interrupción 1), 18 (interrupción 5), 19 (interrupción 4), 20 (interrupción 3), y 21 (interrupción 2). Estos pines pueden configurarse para activar una interrupción en un nivel bajo, un flanco ascendente o descendente, o un cambio en el nivel.
- PWM: 2 a 13 y 44 a 46. proporcionan una salida PWM de 8 bits con la función `analogWrite()`.



- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Estos pines soportan la comunicación SPI utilizando la biblioteca SPI.
- LED: 13. Hay un LED incorporado conectado al pin digital 13.
- TWI: 20 (SDA) y 21 (SCL). TWI soporte de comunicación utilizando la biblioteca Wire.

Arduino Mega puede ser alimentado mediante el puerto USB o con una fuente externa de poder. La alimentación es seleccionada de manera automática.

Si se decide trabajar con una fuente externa de poder se debe utilizar un convertidor AC/DC y regular dicho voltaje en el rango operativo de la placa. De igual manera se puede alimentar el micro mediante el uso de baterías. Preferiblemente el voltaje debe estar en el rango de los 7V hasta los 12V.

Los pines de alimentación son los siguientes:

- Vin. La tensión de entrada a la placa cuando se utiliza una fuente de alimentación externa (en contraposición a 5 voltios de la conexión USB o de otra fuente de alimentación regulada). Se puede suministrar tensión a través de este pin.
- 5V. Este pin es una salida de 5 V regulada del regulador de la placa. La placa puede ser alimentada ya sea desde el conector de alimentación de CC (7 - 12 V), por el conector USB (5 V), o por el pin VIN de la placa(7-12V).
- 3V3. Un suministro de 3,3 voltios generado por el regulador de la placa. El consumo de corriente máximo es de 50 mA.
- GND. Los pines de tierra.

El Atmega2560 tiene 256 KB de memoria flash para almacenar el código (de la que se utilizan 8 KB para el cargador de arranque), 8 KB de SRAM y 4 KB de EEPROM.

## **2.3. LABVIEW**

### **2.3.1 Generalidades**

LabView es el acrónimo de Laboratory Virtual Instruments Engineering Workbench. Es un lenguaje y a la vez un entorno de programación gráfica.

Inicialmente fue desarrollado para facilitar y coleccionar datos de instrumentos de laboratorios utilizando sistemas de adquisici3n de datos, sin embargo, actualmente, LabView es utilizado tambi3n para el procesamiento de informaci3n, an3lisis de datos, control de instrumentos electr3nicos y equipos.

La habilidad de LabView de obtener informaci3n de un entorno exterior, usar esa informaci3n dentro de un programa y enviar los resultados nuevamente al mundo exterior permite interactuar y controlar eventos dentro de procesos externos y no solamente dentro de un computador usados com3nmente en la industria, pudiendo, con estos criterios desarrollar sistemas SCADA para salas de control.

### **2.3.2 Sensores y Actuadores**

#### **a. Motor Paso a Paso**

El motor paso a paso es un dispositivo electromec3nico que convierte una serie de impulsos el3ctricos en desplazamientos angulares discretos. Estos motores est3n constituidos por un rotor sobre el que van aplicados distintos imanes permanentes y por un cierto n3mero de bobinas excitadoras en su estator. Toda la conmutaci3n (o excitaci3n de las bobinas) deben ser externamente manejadas por un controlador.

Un paso es el movimiento m3nimo que puede hacer un motor de pasos. Estos motores no giran de manera continua, su giro es de acuerdo a n3mero espec3fico de pasos.

Los motores paso a paso se mueven usualmente mucho m3s lento que los motores DC, ya que hay un l3mite m3ximo para la velocidad a la que se pueden ir dando los pasos, pero ofrecen mayor torque o mayor fuerza sobre su eje.

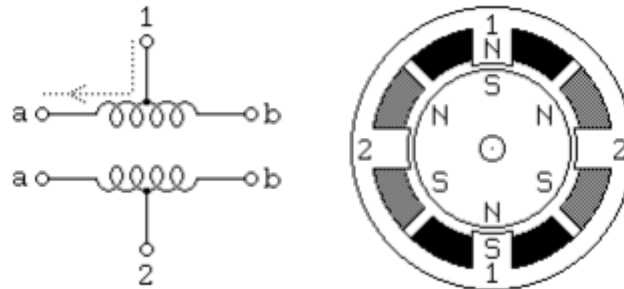
En el caso de estudio se ver3 el siguiente tipo de motor paso a paso:

- ***Unipolares***

Este tipo de motor de paso posee dos bobinas internas, cada una est3 dividida por una conexi3n central com3n por donde recibe la alimentaci3n rodeando el eje central el cual est3 imantado. Del motor salen seis cables: cuatro de ellos corresponden a los extremos de las dos bobinas y los otros dos a sus respectivas conexiones centrales comunes. (Si el modelo de



motor solo ofrece cinco cables quiere decir que las conexiones centrales se encuentran conectadas internamente una a la otra y ambas se alimentan a partir de un único cable.)



**Fig. 2.3.** Motor Paso a Paso Unipolar

**Fuente:** <http://server-die.alc.upv.es/asignaturas/lсед/2002-03/MotoresPasoapaso/Motorespasoapaso.pdf>

Existen tres secuencias posibles para este tipo de motores:

- *Secuencia Normal:* Esta es la secuencia más usada y la que generalmente recomienda el fabricante. Con esta secuencia el motor avanza un paso por vez y debido a que siempre hay al menos dos bobinas activadas, se obtiene un alto torque de paso y de retención.
- *Secuencia del tipo wave drive:* En esta secuencia se activa solo una bobina a la vez. En algunos motores esto brinda un funcionamiento más suave. Al estar solo una bobina activada, el torque de paso y retención es menor.
- *Secuencia del tipo medio paso:* En esta secuencia se activan las bobinas de tal forma de brindar un movimiento igual a la mitad del paso real. Para ello se activan primero dos bobinas y luego solo una y así sucesivamente.

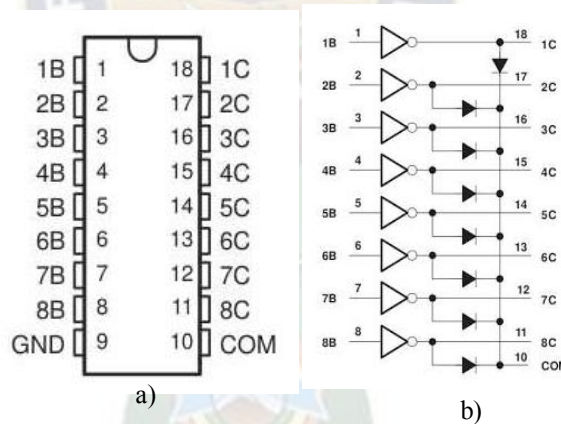
### **b. ULN 2803**

El ULN2803 es un circuito integrado que se emplea como una interface de potencia para acondicionar pulsos o señales digitales de baja intensidad de tal manera que puedan mover componentes que requieren altas corrientes o voltajes como relays, motores, series de leds, displays, etc.

Este integrado tiene en su interior 8 transistores NPN tipo Darlington con sus correspondientes diodos de supresión. Los diodos tienen como objetivo el eliminar las corrientes inversas que se producen cuando se conectan cargas inductivas, esto evita que los transistores se dañen. Cada transistor Darlington cuenta con una capacidad de carga de corriente de pico 600mA.

Algunas características importantes son las siguientes:

- **Entradas:** Soportan un voltaje máximo de 30v. Las entradas se encuentran desde el pin 1 al 8.
- **Salidas:** Se encuentran desde el pin 11 al 18. Se pueden conectar en paralelo para aumentar la corriente máxima de 500ma.
- **Voltaje de Alimentación:** Voltajes de 5, 9 y 12 voltios, pero también puede soportar hasta de 50V. Se hace por el pin 10 (COM), que más bien se trata de un punto común para todos los cátodos de los diodos de supresión. La conexión a tierra (GND) la encontramos en el pin 9.



**Fig. 2.4.** ULN 2803

a) Encapsulado, b) Esquema

**Fuente:** <http://electronica-teoriaypractica.com/uln2803/>

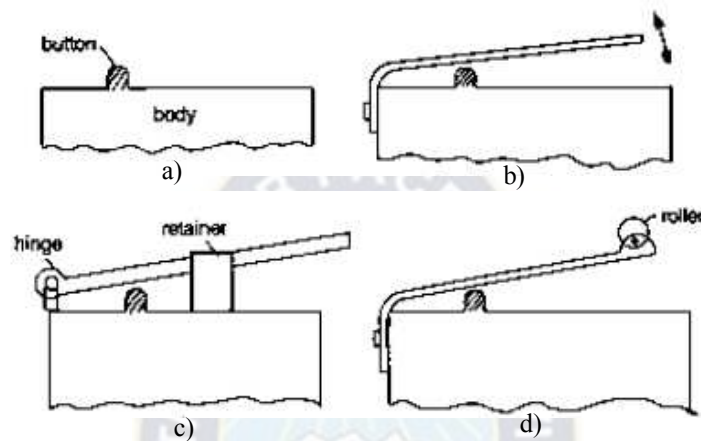
### c. Finales de Carrera

Los Finales de Carrera son un tipo de Microswitch. La marca distintiva del microswitch es su capacidad de cambio a dos posibles estados estables.

La operación del microswitch o final de carrera es presionando un pin de pequeño diámetro, esto facilita que el cambio de estado se encuentre debidamente sellado en la posición deseada.

Los finales de carrera son también llamados interruptores de posición, son interruptores que detectan la posición de un elemento móvil mediante accionamiento mecánico.

Existen varios tipos y tamaños de finales de carrera, pueden inclusive tener un torque de operación muy bajo, siendo estos accionados por presiones muy pequeñas, esto dependerá de la aplicación que se le dará al dispositivo.



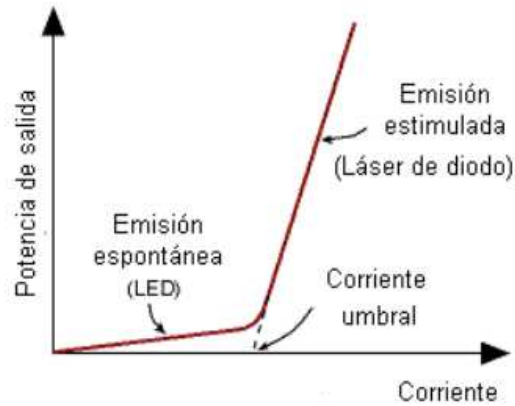
**Fig. 2.5.** Tipos de Michoswitch  
a) Botón simple , b) Palanca , c) Con retenedor, d) Palanca con rueda  
**Fuente:** Sinclair, Ian R. (2001): “*Sensors and Transducers*”

#### **d. Diodos Láser**

La palabra LÁSER significa Light Amplification by Stimulated Emission of Radiation, es decir, Luz amplificada por la emisión estimulada de radiación.

Una fuente de luz normal produce luz de muchas longitudes de onda diferentes emitidas en muchas direcciones; un láser, genera ondas luminosas con una única longitud de onda la cual se encuentra en fase unas con otras y todas en la misma dirección. Esto hace que la luz láser sea de un color muy puro y extremadamente intenso.

Un diodo láser es una forma especial de diodo emisor de luz (LED) semiconductor, emite luz coherente cuando se aplica un voltaje a sus terminales.



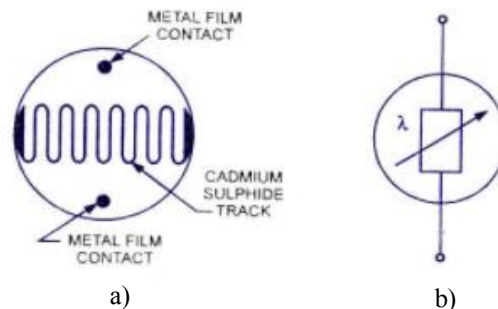
**Fig. 2.6.** Curva del Diodo Láser

Fuente: <https://www.onubaelectronica.es/diodo%20laser.htm>

### e. Fotorresistor

El fotorresistor, comúnmente llamado LDR (Light-dependent resistor - resistor dependiente de la luz) es un componente electrónico cuya resistencia varía dependiendo de la luz incidente. La celda fotoconductor, usando una cinta de sulfuro de cadmio, tiene alta resistencia en ausencia de luz y la misma baja cuando el material es iluminado.

El sulfuro de cadmio es depositado como un patrón de hilo en un aislador, como el largo de es patrón afecta la sensibilidad, usualmente se utiliza la forma de una línea en zigzag. Luego la celda es encapsulada en una resina transparente o encajonada en vidrio para proteger el sulfuro de cadmio de la contaminación atmosférica.



**Fig. 2.7.** Fotorresistor LDR

a) Estructura Básica b) Símbolo

Fuente: <https://sohailansaari.wordpress.com/2013/12/09/light-dependent-resistor-ldr-its-application/>

## Capítulo 3

### Desarrollo del Proyecto

#### 3.1 Diseño y Funcionamiento del Aparato de Laboratorio

##### 3.1.1. Geometría del Prototipo

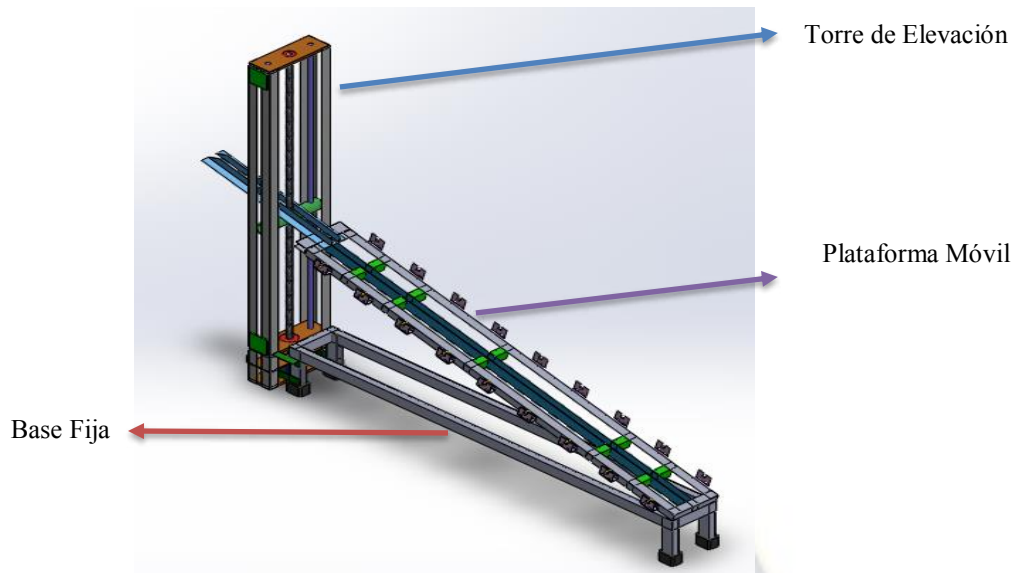
El aparato cuenta con las siguientes tres partes fundamentales.

- La torre de elevación
- La plataforma móvil
- La base fija.

El movimiento del aparato depende del motor de pasos, éste se encuentra instalado físicamente a la base de la torre de elevación. El torque del motor da inicio al giro del tornillo sin fin o barra roscada que eleva un soporte de elevación donde apoyará la plataforma móvil.

La base y la plataforma móvil se encuentran unidas mediante una bisagra que permite el desplazamiento de la plataforma móvil. La base también se encuentra fija a la torre de elevación. La interconexión de los tres elementos logra formar un único aparato.

El motor iniciará el movimiento de acuerdo al laboratorio escogido y a la decisión del usuario. La Fig. 3.1 muestra el prototipo completo donde se puede apreciar la interconexión de los elementos del aparato.



**Fig. 3.1.** Prototipo Completo  
**Fuente:** Elaboración propia mediante herramienta SolidWorks 2012

### a. Torre de Elevación

La torre se encuentra en la parte posterior del aparato y cuenta con los siguientes elementos:

- Motor de pasos.
- Barra roscada.
- Rodamiento superior y rodamiento inferior.
- Barras guías de desplazamiento.
- Soporte móvil de elevación.
- Final de carrera superior.
- Fina de carrera inferior.

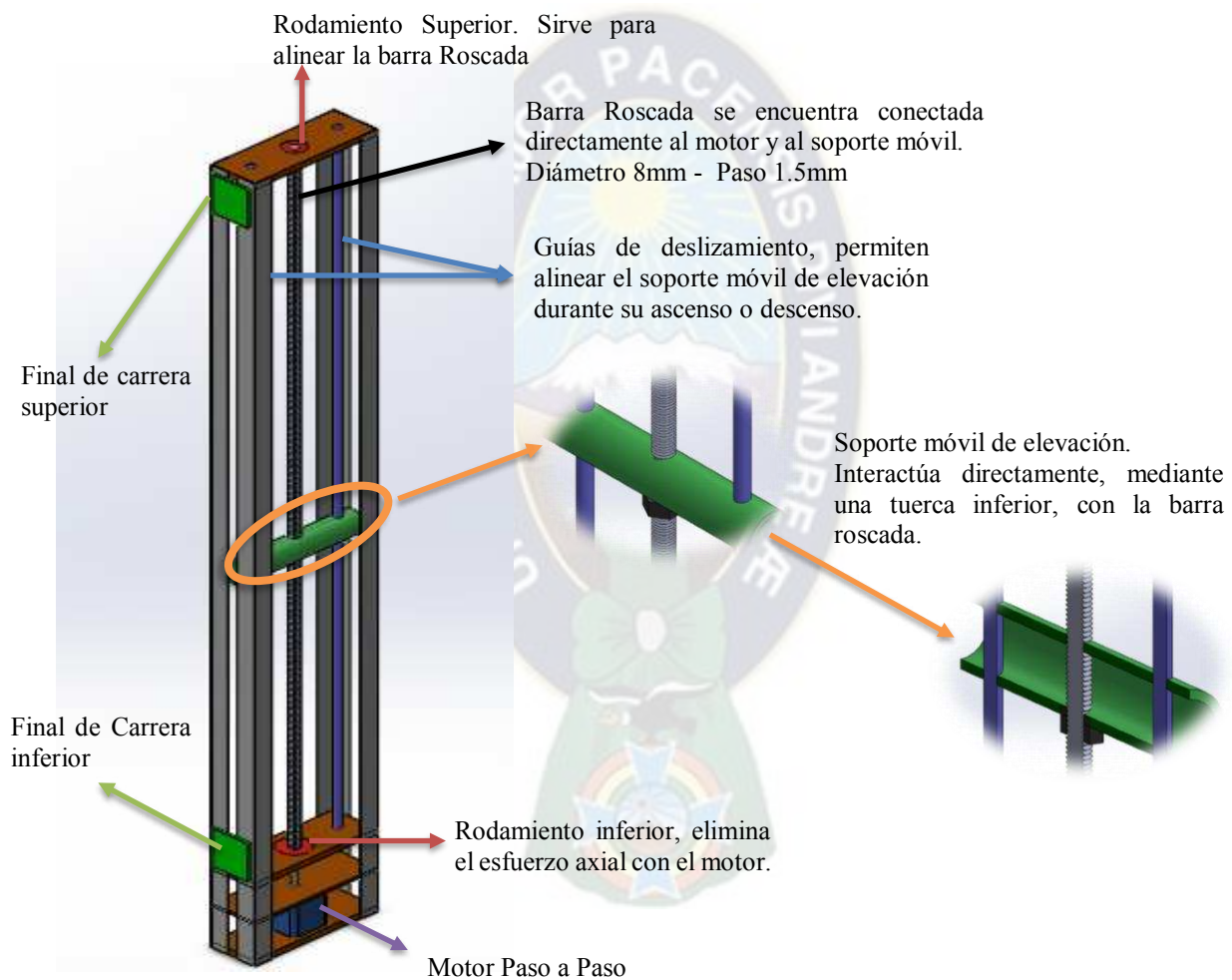
La barra roscada se encuentra conectada físicamente con el motor y con dos rodamientos, uno inferior para eliminar cualquier esfuerzo axial y uno superior para mantener la barra alineada además de suavizar el giro de la misma.

Dentro del soporte móvil de elevación se tiene adherida una tuerca acoplada a la barra roscada.



El motor paso a paso inicia el movimiento activando el giro de la barra roscada que permite el ascenso o descenso del soporte móvil de elevación. Las guías de desplazamiento se encargan de la alineación del soporte durante el movimiento.

La figura 3.2. muestra los elementos que conforman la torre de elevación.



**Fig. 3.2.** Torre de Elevación

**Fuente:** Elaboración propia mediante herramienta SolidWorks 2012

## b. Plataforma Móvil

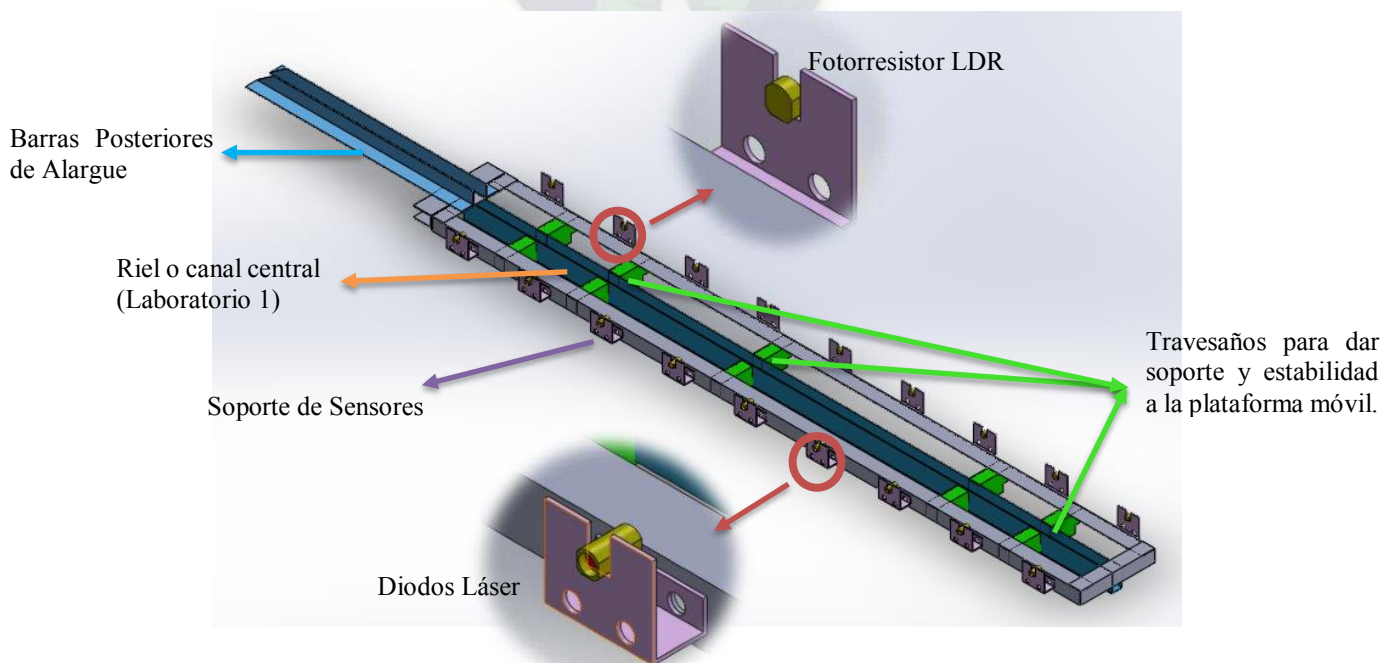
La plataforma móvil cuenta con los siguientes elementos

- Nueve diodos láser
- Nueve fotorresistores
- Riel o canal central.
- Barras posteriores de alargue
- Bases intercambiables

La plataforma móvil cuenta con “bases” intercambiables que tapan el canal principal para poder lograr el estudio de la fricción en diversas superficies.

La imagen que se tiene a continuación muestra la plataforma móvil sin la presencia de bases de fricción. El uso de la plataforma móvil de esta manera permite tener un canal central que representaría una plataforma sin fricción.

Las barras posteriores de alargue se apoyan en el soporte móvil de la torre principal, permitiendo compensar el tamaño de la plataforma móvil ante la variación del ángulo.



**Fig. 3.3.** Plataforma Móvil

**Fuente:** Elaboración propia mediante herramienta SolidWorks 2012



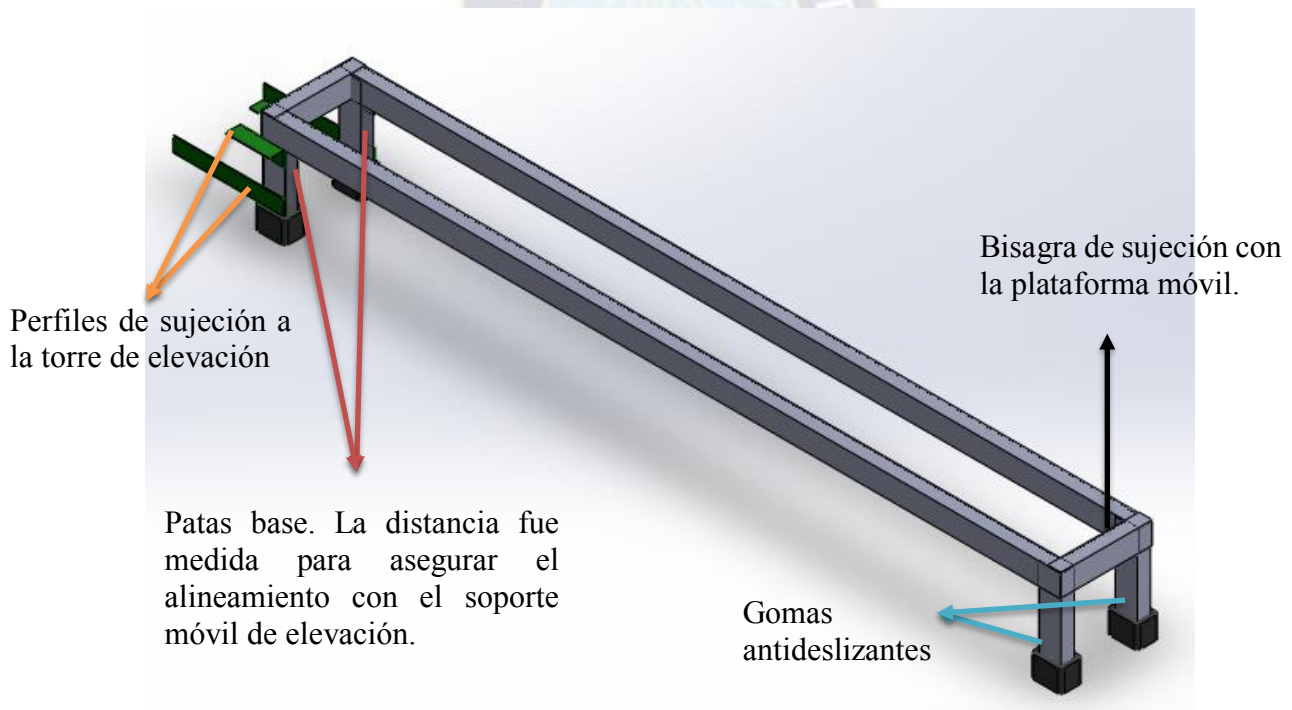
### c. Diseño de Base Fija

La base consta de los siguientes elementos importantes:

- Perfiles de sujeción a la torre de elevación.
- Gomas antideslizantes.
- Bisagra de sujeción a la plataforma móvil.

La base fija mantiene la unión entre la torre y la plataforma móvil. El largo de las patas de la base ayuda a alinear el soporte móvil y la plataforma móvil encontrando el ángulo  $0^\circ$ .

La plataforma móvil se une en la parte inferior de la base mediante una bisagra que permitirá el libre ascenso o descenso de la plataforma móvil.



**Fig. 3.4.** Base Fija

**Fuente:** Elaboración propia mediante herramienta SolidWorks 2012

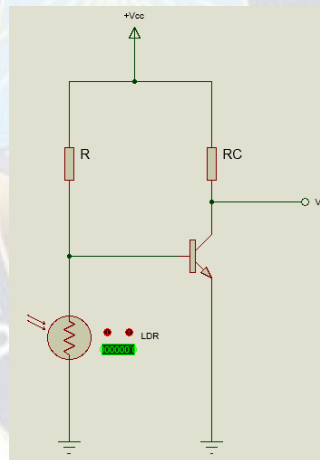
## 3.2 Diseño de Circuitos de Control y Potencia.

### 3.2.1 Diseño de Conmutadores Láser

La conmutación sin contacto no depende de ninguna forma de acción mecánica, con lo que la oxidación, atascos o desgaste no son un problema. Este tipo de conmutación tiende a ser de rápido accionamiento.

Existen varios diseños de conmutación sin contacto, uno de ellos aprovecha las propiedades de emisión de luz coherente del diodo láser y la propiedad de variación resistiva ante la incidencia de luz como es el fotorresistor o LDR.

Para diseñar el circuito se usa la propiedad de conmutación del transistor usando el circuito mostrado en la figura.

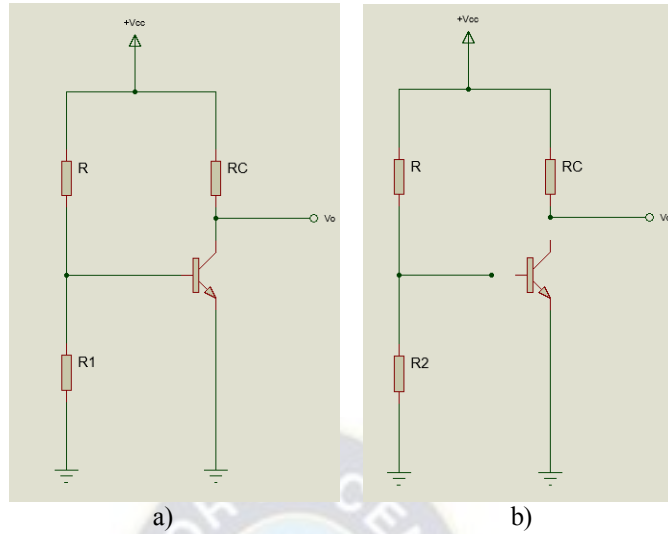


**Fig. 3.5.** Circuito del Conmutador Láser

**Fuente:** Elaboración propia mediante herramienta Proteus V8.2

El diseño del circuito es el siguiente:

Para diseñar, separamos en dos circuitos. De Saturación y de Corte:



**Fig. 3.6.** Circuito del Conmutador Láser en Saturación y Corte  
a) Circuito en corte b) Circuito en saturación  
**Fuente:** Elaboración propia mediante herramienta Proteus V8.2

Ecuaciones para el estado de corte:

$$V_{cc} = I_2 R + 0,2 \quad 3.2.1$$

$$0,2 = I_2 R_2 \quad 3.2.2$$

Ecuaciones para el estado de saturación:

$$V_{cc} = I_c R_c + 0,3 \quad 3.2.3$$

$$V_{cc} = (I_1 + I_b) R + 0,8 \quad 3.2.4$$

$$0,8 = I_1 R_1 \quad 3.2.5$$

$$I_b \geq \frac{I_c}{\beta} \quad 3.2.6$$

$R_1$  y  $R_2$ , se conocen luego de medir los valores de la resistencia del fotorresistor cuando el láser incide en este ( $R_2$ ) y cuando el haz de luz es interrumpido ( $R_1$ ). Se tienen las siguientes medidas:

$$R_1 = 100 \text{ K}\Omega$$

$$R_2 = 1 \text{ K}\Omega$$

Se resuelven las ecuaciones del circuito en estado de corte:

De la ecuación 3.2.2 se obtiene el valor de  $I_2$

$$I_2 = 0,2m[A]$$

Reemplazando el valor obtenido en la ecuación 3.2.8 se obtiene el valor de R.

$$R = \frac{(V_{cc} - 0,2)}{I_2} \quad 3.2.7$$

$$R = 24K[\Omega]$$

Se resuelven las ecuaciones del circuito en estado de saturación:

De la ecuación 3.2.5 se obtiene el valor de  $I_1$

$$I_1 = 8\mu[A]$$

Reemplazando los valores obtenidos de  $I_1$  y R en la ecuación 3.2.4 se calcula el valor de  $I_b$ .

$$I_b = \frac{(V_{cc} - 0,8)}{R} - I_1 \quad 3.2.8$$

$$I_b = 0,167m[A]$$

Se continúa el cálculo partiendo de la ecuación 3.2.6:

$$I_b \geq \frac{I_c}{\beta}$$

Para convertir en una ecuación que cumpla la condición de “Mayor o igual”, para este circuito se usará el factor ‘2’, por lo tanto:

$$I_b = \frac{2 I_c}{\beta} \quad \rightsquigarrow \quad I_c = \frac{\beta I_b}{2}$$

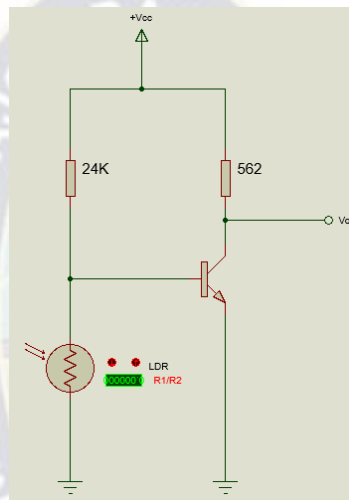
$$I_c = 8,35m[A]$$

Se reemplaza el valor de  $I_c$  en la ecuación 3.2.3 para obtener el valor de  $R_c$

$$R_c = \frac{(V_{cc} - 0,3)}{I_c} \quad 3.2.9$$

$$R_c = 562[\Omega]$$

Con los datos obtenidos ya se tiene el diseño finalizado. Se verificó el funcionamiento de cada circuito conmutador ya que se usarán nueve para la obtención de datos.



**Fig. 3.7.** Circuito Diseñado del Conmutador Láser  
**Fuente:** Elaboración propia mediante herramienta Proteus V8.2

### 3.2.2 Configuración eléctrica de Motor Paso a Paso

#### *Especificaciones Generales del Motor*

- Stepper Type: 17PM-K302
- Voltaje: 24V
- Amperaje: 1A
- Grados: 1.8°

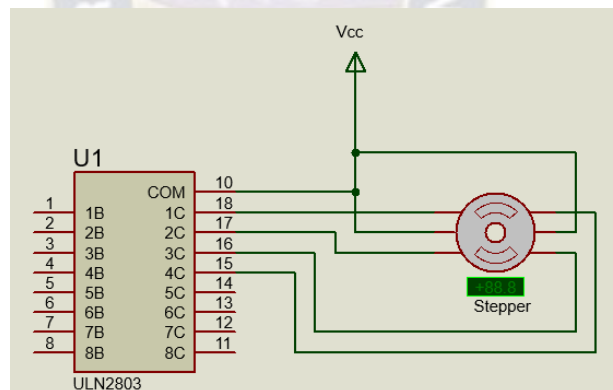
La programación del motor Paso a Paso fue realizada mediante la librería “*stepper*” de Arduino que permite controlar motores unipolares o bipolares de acuerdo a la cantidad de pines a usar. Para el prototipo se usó un motor de pasos Unipolar conectando cuatro pines al

Arduino, por lo tanto, la librería “Stepper” usa para el movimiento la secuencia mostrada en la tabla:

Pasos Unipolar	A1	A2	B1	B2
1	ON	OFF	ON	OFF
2	OFF	ON	ON	OFF
3	OFF	ON	OFF	ON
4	ON	OFF	OFF	ON

**Tabla. 3.1.** Secuencia para Motor Paso a Paso  
**Fuente:** Librería “Stepper” de Arduino

Es necesario amplificar la corriente que entrega el Arduino al motor mediante una configuración Darlington, por lo que se usó el integrado ULN 2803 como sigue:



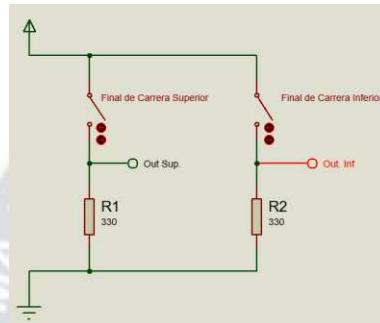
**Fig. 3.8.** Conexión ULN 2803 a Motor Paso a Paso  
**Fuente:** Elaboración Propia mediante herramienta Proteus V8.2

### 3.2.3 Finales de carrera

Como se explicó anteriormente, la torre de elevación posee soporte de elevación donde se apoya la plataforma móvil, este soporte se desplaza de forma vertical de acuerdo al tornillo sin fin activado por el motor.

Los finales de carrera superior e inferior son de protección, siendo accionados mecánicamente por un extremo del soporte de elevación. El inferior se activa cuando el soporte de elevación se encuentra en el límite inferior definido y a su vez determina el ángulo cero en esta posición. El superior se activa cuando la plataforma móvil haya llegado a su límite superior máximo deteniendo el proceso y parando el giro del motor de pasos.

La conexión electrónica es la que se muestra a continuación:



**Fig. 3.9.** Conexión de Finales de Carrera

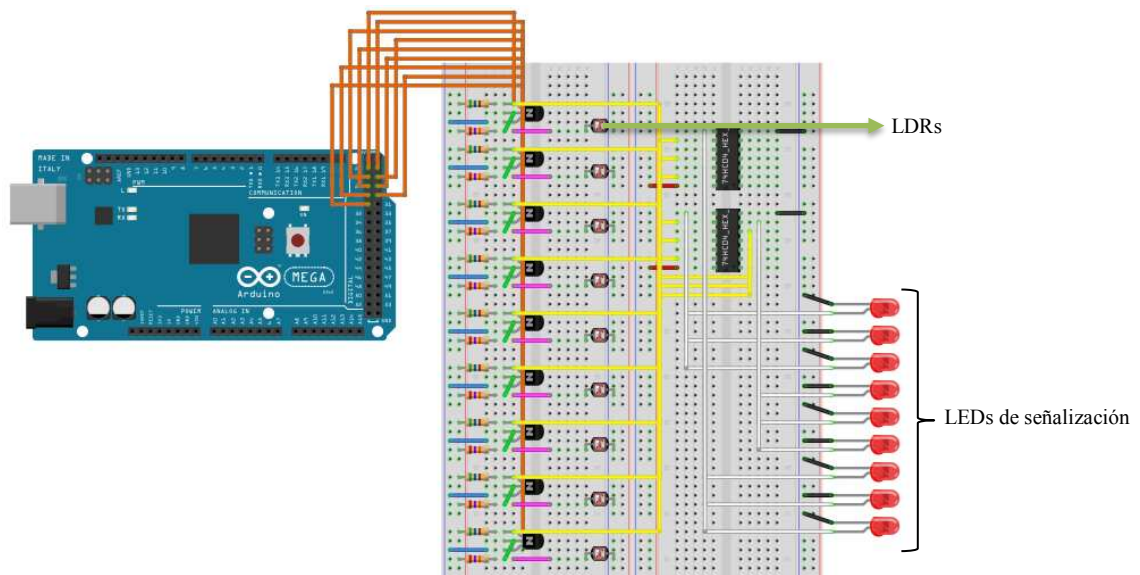
**Fuente:** Elaboración Propia mediante herramienta Proteus V8.2

### 3.2.4 Conexión con Arduino

La conexión al Arduino de los conmutadores se realizó usando los pines del puerto A (Digital Pin 22 al Digital Pin 29) y Puerto C (Digital Pin 30). Los puertos escogidos son de lectura y escritura digital exclusivamente.

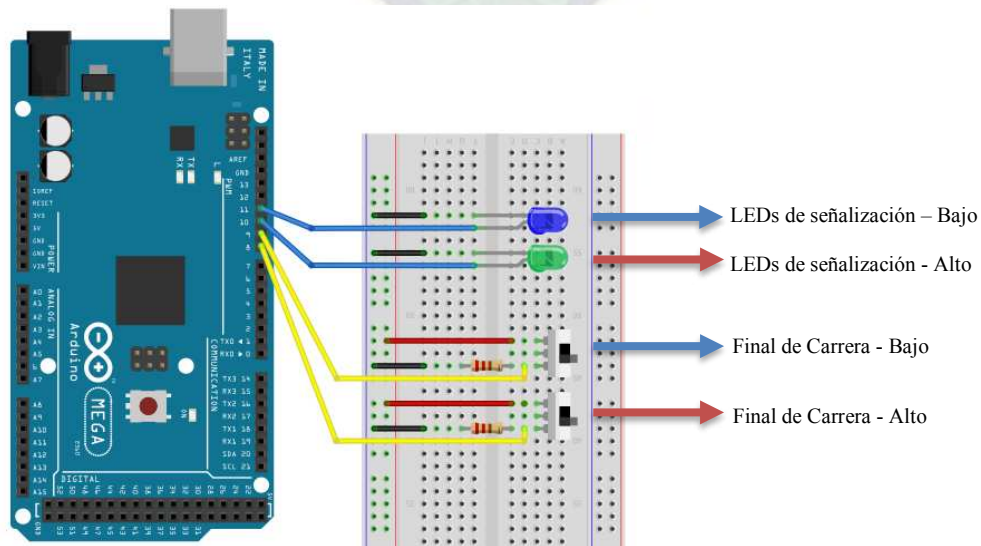
Sólo es necesaria la conexión al Arduino de los fotorresistores puesto que los diodos láser serán conectados directamente a Vcc para alimentarlos para que generen el haz de luz que incide con el LDR.





**Fig. 3.10.** Conexión de fotorresistores  
**Fuente:** Elaboración Propia mediante herramienta Fritzing V0.9.3

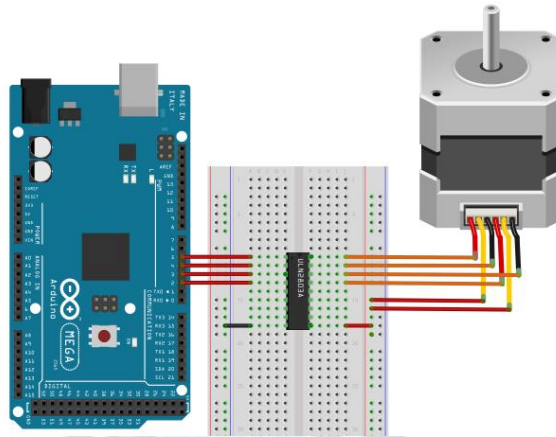
Como mencionamos anteriormente, los finales de carrera se conectan como interruptores. A su vez, mediante programación, se activan LEDs de señalización dependiendo al sensor activado. El Microswitch que indica posición baja se conectará al Pin 9 y el Microswitch, que indica posición alta se conecta al Pin 8. Los LEDs de señalización se conectan a los pines 11 y 10 respectivamente.



**Fig. 3.11.** Conexión de Finales de Carrera  
**Fuente:** Elaboración Propia mediante herramienta Fritzing V0.9.3

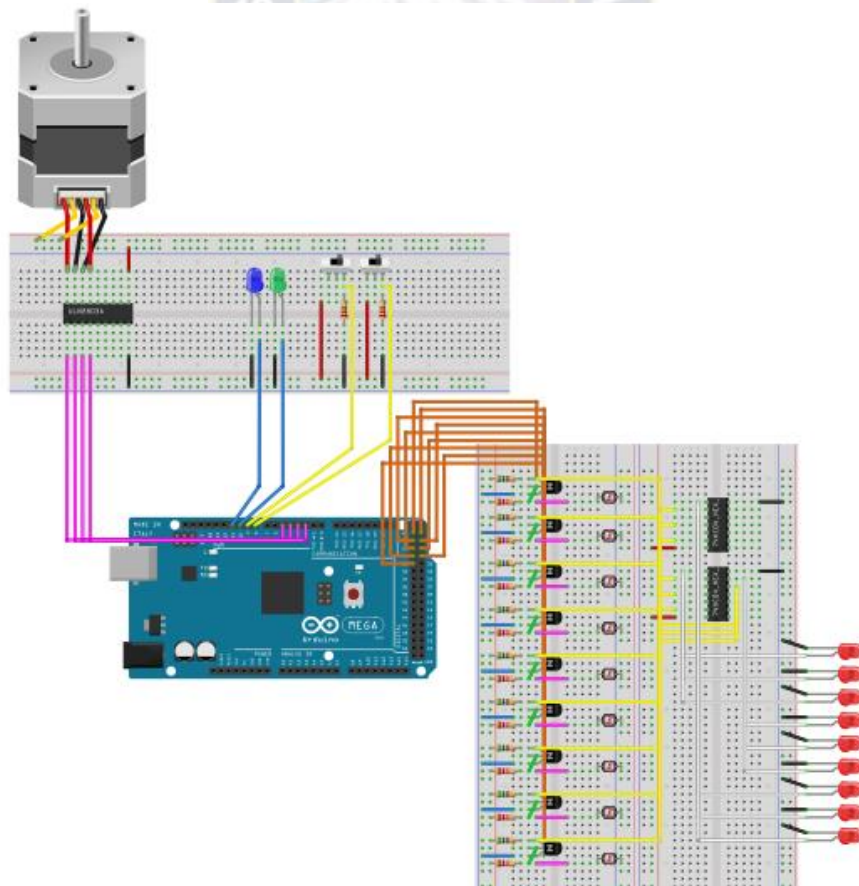


El motor de pasos se conecta, mediante el ULN 2803 (Configuración Darlington), a los pines dos, tres, cuatro y cinco del Arduino.



**Fig. 3.12.** Conexión de Motor paso a paso  
**Fuente:** Elaboración Propia mediante herramienta Fritzing V0.9.3

Finalmente se tienen todas las conexiones al Arduino como se muestra a continuación:



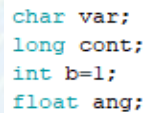
**Fig. 3.13.** Conexión completa - Arduino  
**Fuente:** Elaboración Propia mediante herramienta Fritzing V0.9.3

## 3.3 Programación en Arduino

### 3.3.1 Configuraciones iniciales y variables globales

Variables globales usadas en el programa:

- var - Variable global, definida tipo “char”. Usada para guardar el carácter leído por puerto serial. (Función “serialA(”)”).
- count – Variable global, definida tipo “long”. Usada para guardar el conteo de pasos del motor paso a paso.
- ang – Variable global, definida tipo “float”. Usada para almacenar el ángulo objetivo al cual se desea llegar.



```
char var;  
long count;  
int b=1;  
float ang;
```

**Fig. 3.14.** Configuración Arduino – Variables Globales  
**Fuente:** Elaboración Propia mediante herramienta IDE Arduino 1.6.0

Para las configuraciones iniciales del motor paso a paso es necesario tomar en cuenta lo siguientes puntos:

- Se incluye la librería Stepper.h para tener acceso a la configuración del motor paso a paso.
- Se sabe que el motor de pasos que se usa es un motor que da 200 pasos por giro, por lo tanto, se crea una variable conteniendo el valor de máximos pasos del motor usado.
- Finalmente es necesario especificar los pines que serán usados para el movimiento del motor paso a paso. En este caso se usan los pines 2, 3,4, 5 del Arduino MEGA 2560.

La programación de lo mencionado anteriormente es la siguiente:

```
#include<Stepper.h>  
int steps=200; //pasos por giro completo  
Stepper motor(steps,2,3,4,5);
```

**Fig. 3.15.** Configuración en programación  
**Fuente:** Elaboración Propia mediante herramienta IDE de Arduino 1.6.0

### 3.3.2 Configuraciones previas al programa general

Inicialmente se identifican los puertos de comunicación y la velocidad de comunicación Serial. En el caso de los puertos A y C del Mega 2560, se configura mediante el acceso directo a puertos para aprovechar la velocidad de lectura que existe.

Cada puerto es controlado por tres registros, los cuales también están definidos como variables en el lenguaje del Arduino:

- El registro DDR, determina si el pin es una entrada o una salida.
- El registro PORT controla si el pin está en nivel alto o en nivel bajo.
- El registro PIN permite leer el estado de un pin que se ha configurado con entrada usando la función `pinMode()`.

Las funciones `digitalRead()` y `digitalWrite()` son cada una cerca de una docena de líneas de código, lo cual, al ser compilado se convierte en unas cuantas instrucciones máquina. Cada instrucción de máquina necesita un ciclo de reloj de aproximadamente 16MHz, esto puede sumar mucho tiempo en aplicaciones muy dependientes del tiempo. El Registro PORT(Puerto) puede hacer el mismo trabajo en muchos menos ciclos de trabajo. Al utilizar Registros PORT(Puerto) se tiene la ventaja de que con solo una instrucción se declara el pin o puerto como entrada o salida y también como estado HIGH o LOW. Se debe especificar el puerto que se va a configurar. Siendo “0” Entrada y “1” Salida.

La lectura de los interruptores sin contacto requiere alta velocidad el momento de su lectura o activación, por lo que es importante el manejo de los puertos mediante las configuraciones anteriormente descritas.

Se configuran, a su vez y de manera regular, los pines 8,9,10 y 11 de lectura y escritura, estos pines corresponden a la lectura del estado de los finales de carrera y escritura de los LEDs de señalización que no requieren de altas velocidades de lectura.

*Pin 8     →     Final de carrera ALTO*  
*Pin 9     →     Final de carrera BAJO*  
*Pin 10    →     LED de señalización ALTO*

Finalmente, es necesario configurarla velocidad de comunicación Serial, se usó el valor típico 9600 Baudios.

```
void setup()
{
  DDRA=B00000000;//Puerto A como entradas PA0 a PA7(Lab 1 & 3)
  DDRC=B01111111;//Puerto C Pin PC7 como entrada y pines PC0 al PC6 Como salidas(No serán usadas)(Lab 1 & 3)
  pinMode(8,INPUT);
  pinMode(9,INPUT);
  pinMode(10,OUTPUT);//
  pinMode(11,OUTPUT);//
  Serial.begin(9600);
}
```

**Fig. 3.16.** Configuración Void Setup()

**Fuente:** Elaboración Propia mediante herramienta IDE de Arduino 1.6.0

### 3.3.3 Funciones

#### a. Función – serialA()

La Función prepara al puerto serial y espera a que se inicie cualquier comunicación con este puerto.

La función interna “Serial.available()” se encarga de verificar si se tienen datos en el puerto, si no hay un dato disponible la función entregará un cero.

Se crea un bucle a partir de esta función para que la nueva función creada se active sólo en caso de existir comunicación por el puerto serial.

La función “Serial.read()” permite hacer la lectura caracteres del puerto serial. En la última línea de código dentro de la función creada se procede a guardar la información leída en la variable global “var”.

```
//*****
//*****Funcion Serial*****
//*****
void serialA()
{
  while(Serial.available()==0){}
  var=Serial.read();
}
```

**Fig. 3.17.** Función “serialA()”

**Fuente:** Elaboración Propia mediante herramienta IDE de Arduino V 1.6.0

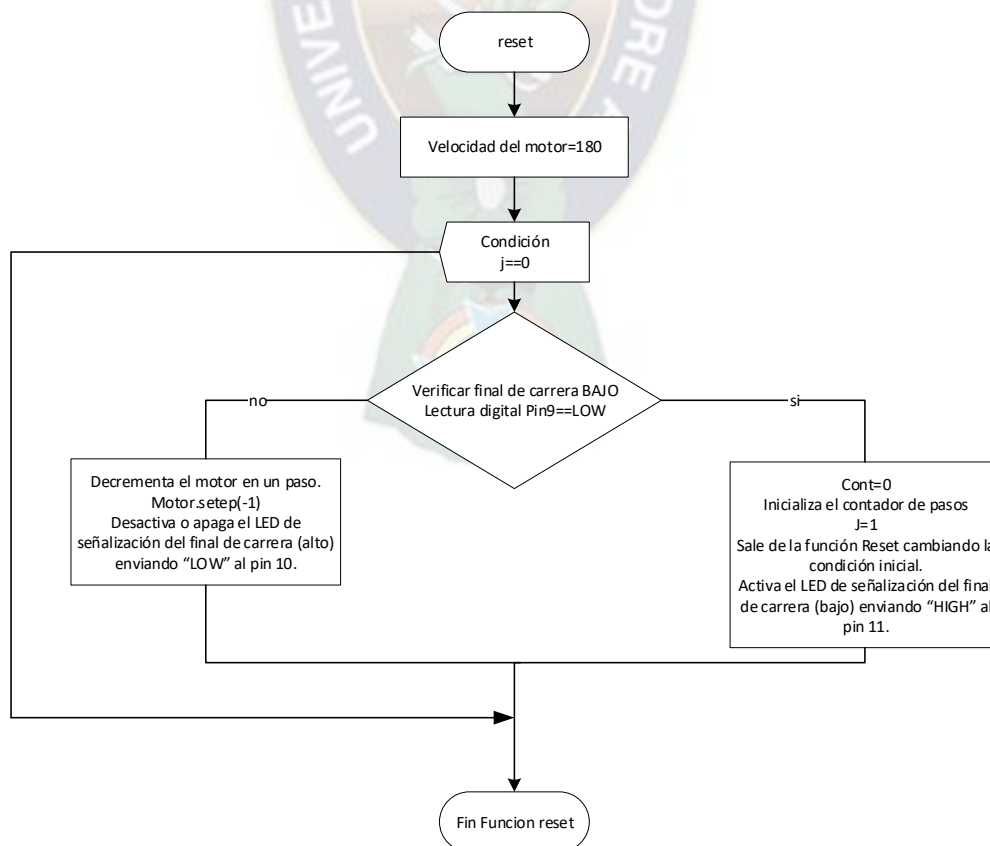
## b. Función – reset(int j)

La función “reset()” activa el motor llevando la plataforma móvil al ángulo cero “0°”, en esta posición se activa el final de carrera inferior y da inicio al contador de pasos del motor asignando, internamente, el valor de cero a la variable que representa el conteo de pasos.

El programa inicia con un bucle while cuya condición “j=0” asegura que la función continúe hasta llegar a la posición inicial de ángulo “0°”, posición en la cual la condición cambiará a “j=1” terminando la función.

Durante el proceso y dentro del bucle se consulta el estado del final de carrera inferior, si éste está activo significará que la plataforma móvil ya se encuentra en la posición inicial y en ángulo cero “0°”, por lo que saldrá del bucle y activará el LED de señalización correspondiente. Si el final de carrera no se encuentra activo entonces hará un decremento de un paso a la vez usando la función “motor.step()”, propia de la librería “stepper”, hasta que el final de carrera inferior esté activado.

Se resume lo anterior en un diagrama de flujo:



**Fig. 3.18.** Diagrama de Flujo – Función “reset”

**Fuente:** Elaboración Propia mediante herramienta Microsoft Office Visio 2016

Se define la siguiente variable local usada:

- $j$  – Variable local, definida tipo “int”. Usada para usar como condicionante del bucle while. Asegura la ejecución del bucle hasta que cambie de valor.

La forma de llamar a esta función es la siguiente:

*reset(0);*

El “cero” dentro de la función da el valor inicial a la variable local “ $j$ ”. La cual, como se vio en el diagrama de flujo, debe tener un valor de “cero” para poder ingresar al bucle while.

### c. Función – AnglePrincipal(float ang)

Cada ángulo que forme la plataforma móvil con el eje horizontal corresponderá a una cierta cantidad de pasos avanzados por el motor paso a paso. Para encontrar la relación que existe entre el ángulo con la cantidad de paso se realizaron varias pruebas. El objetivo de estas pruebas es encontrar un factor que permita la relación mencionada. Se tomaron medidas de ángulos cada 10000 pasos de motor.

A continuación, se presentan tablas comparativas de las pruebas realizadas.

N° Pasos	Prueba 1		Prueba 2		Prueba 3		Prueba 4	
	Ángulo (subida)	Ángulo (bajada)	Ángulo (subida)	Ángulo (bajada)	Ángulo (subida)	Ángulo (bajada)	Ángulo (subida)	Ángulo (bajada)
0	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
10000	4,5	4,4	4,5	4,5	4,5	4,6	4,5	4,5
20000	8,7	8,6	8,7	8,7	8,7	8,8	8,7	8,7
30000	13,2	13,1	13,2	13,2	13,2	13,3	13,2	13,2
40000	17,3	17,2	17,3	17,2	17,3	17,3	17,3	17,3
50000	21,5	21,4	21,5	21,4	21,5	21,5	21,5	21,5
60000	25,1	25,1	25,2	25,1	25,2	25,2	25,2	25,2
70000	29,0	29,0	29,0	29,0	29,0	29,1	29,0	29,0
80000	31,7	31,6	31,6	31,6	31,7	31,7	31,7	31,7
87000	34,0	34,0	34,0	34,0	34,0	34,0	34,0	34,0

**Tabla. 3.2.** Pruebas, relación entre Número de pasos y ángulos  
Fuente: Elaboración Propia mediante herramienta Goniómetro digital y analógico



Es necesario hallar los promedios de los valores experimentales medidos, esto permitirá encontrar uno o varios factores para la relación : “ángulo y cantidad de pasos”.

Se tiene la siguiente tabla:

N° Pasos	Promedio		Prom General	Factor - diferencia
	Ángulo (subida)	Ángulo (bajada)		
0	0,10	0,10	0,10	0
10000	4,50	4,60	4,65	4,6
20000	8,70	8,70	8,80	4,2
30000	13,20	13,20	13,25	4,5
40000	17,30	17,25	17,25	4,0
50000	21,50	21,45	21,40	4,2
60000	25,20	25,15	25,30	3,9
70000	29,00	29,00	28,70	3,4
80000	31,70	31,65	31,75	3,1
87000	34,00	34,30	34,20	2,3

**Tabla. 3.3.** Promedio y Factores

**Fuente:** Elaboración Propia mediante herramienta Goniómetro digital y analógico

Como se ve en la tabla anterior, los factores varían uno con respecto a otro, por lo que, para tener una precisión al encontrar el ángulo buscado con un mínimo error se usarán todos los factores. Como sigue:

0 a 10000 pasos	→	0,1 ° a 4,65°	Factor: 4,55
10000 a 20000	→	4,65 ° a 8,8°	Factor: 4,15
20000 a 30000	→	8,8 ° a 13,25°	Factor: 4,45
30000 a 40000	→	13,25 ° a 17,25°	Factor: 4
40000 a 50000	→	17,25 ° a 21,4°	Factor: 4,15
50000 a 60000	→	21,4 ° a 25,3°	Factor: 3,9
60000 a 70000	→	25,3 ° a 28,7°	Factor: 3,4
70000 a 80000	→	28,7 ° a 31,75°	Factor: 3,05
80000 a 87000	→	31,75 ° a 34,2°	Factor: 2,25

**Fig. 3.19.** Promedio y Factores

**Fuente:** Elaboración Propia mediante herramienta Goniómetro digital y analógico

Con esta información se determinan las fórmulas que permitirán encontrar los pasos que debe avanzar el motor de acuerdo a un ángulo introducido por el usuario. Para encontrar cada fórmula se parte de una regla de tres.

En el caso del primer intervalo, la deducción de la ecuación es bastante directa. El factor encontrado es 4.55 para el intervalo “0 a 10000” pasos. El ángulo deseado debe guardarse en la variable “y” y usando la regla de tres se obtiene la ecuación 3.3.1.

A partir del segundo intervalo se tiene que tomar en cuenta la variación que tiene el factor cada 10000 pasos. Por esta razón, es necesario encontrar una fórmula para cada intervalo de avance de manera independiente según los factores obtenidos y mostrados en la Tabla 3.3.

<b>Intervalo</b>	<b>Fórmula de pasos</b>	<b>N° Ec</b>
$0,1 \leq y \leq 4,65$	$x = \left( \frac{10000}{4.55} * y \right)$	3.3.1
$4,65 < y \leq 8,8$	$x = 10000 + \left( \frac{10000}{4.15} * (y - 4.65) \right)$	3.3.2
$8,8 < y \leq 13,25$	$x = 20000 + \left( \frac{10000}{4.45} * (y - 8,8) \right)$	3.3.3
$13,25 < y \leq 17,25$	$x = 30000 + \left( \frac{10000}{4} * (y - 13,25) \right)$	3.3.4
$17,25 < y \leq 21,4$	$x = 40000 + \left( \frac{10000}{4,15} * (y - 17,25) \right)$	3.3.5
$21,4 < y \leq 25,3$	$x = 50000 + \left( \frac{10000}{3,9} * (y - 21,4) \right)$	3.3.6
$25,3 < y \leq 28,7$	$x = 60000 + \left( \frac{10000}{3,4} * (y - 25,3) \right)$	3.3.7



$28,7 < y \leq 31,75$	$x = 70000 + \left( \frac{10000}{3,05} * (y - 28,7) \right)$	3.3.8
$31,75 < y \leq 34,2$	$x = 80000 + \left( \frac{10000}{2,25} * (y - 31,75) \right)$	3.3.9

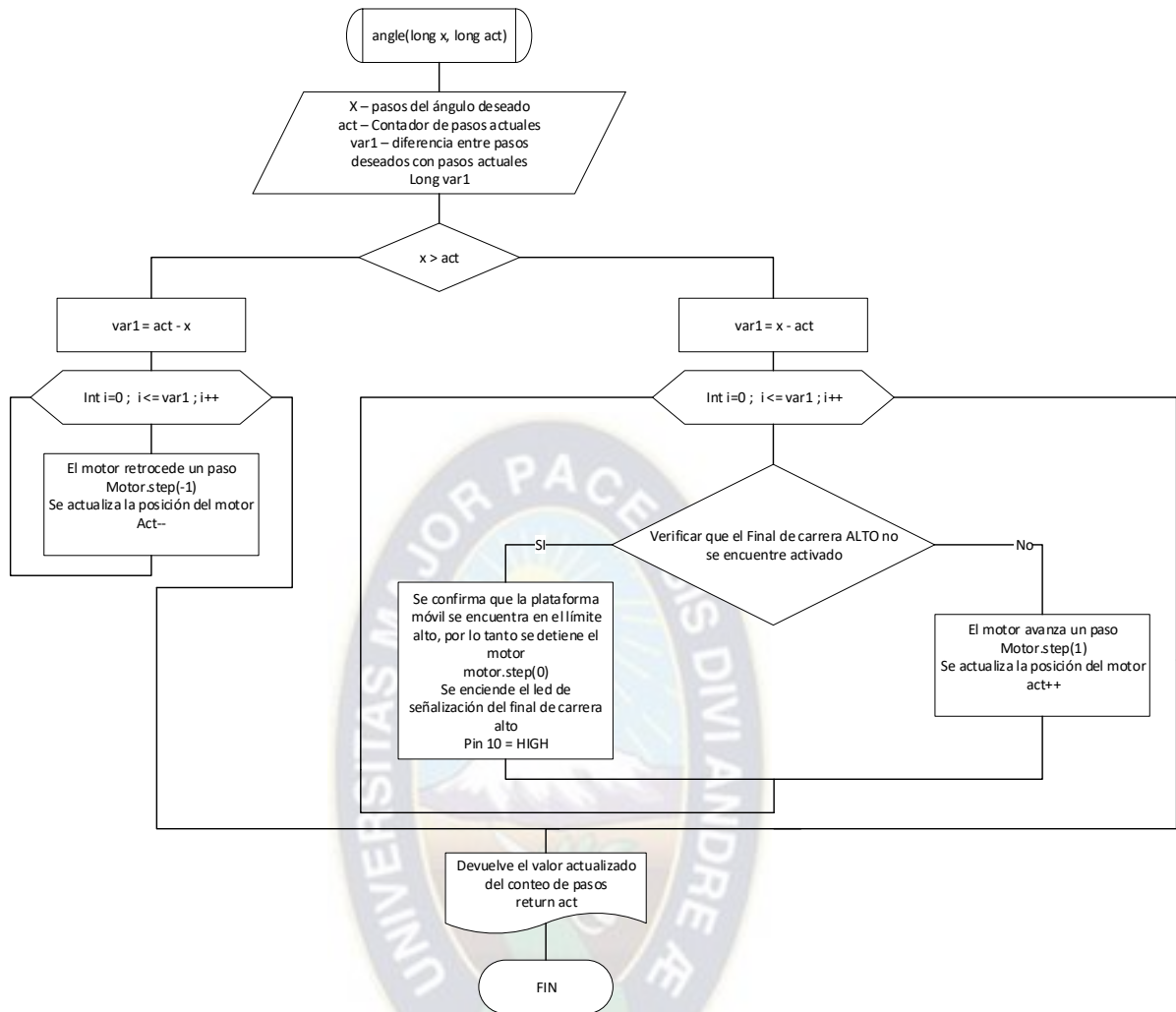
**Tabla. 3.4.** Fórmulas de paso por intervalo  
**Fuente:** Elaboración Propia Herramienta – Excel 2016

Estas fórmulas sirven cuando se inicia el conteo desde cero, o desde el punto de *reset*, sin embargo, no siempre se encontrará la plataforma móvil en este punto. Por ello, es importante conocer la posición actual de la plataforma móvil además de la posición deseada.

Para esto se presenta una Subfunción encargada de realizar estas comparaciones mencionadas.

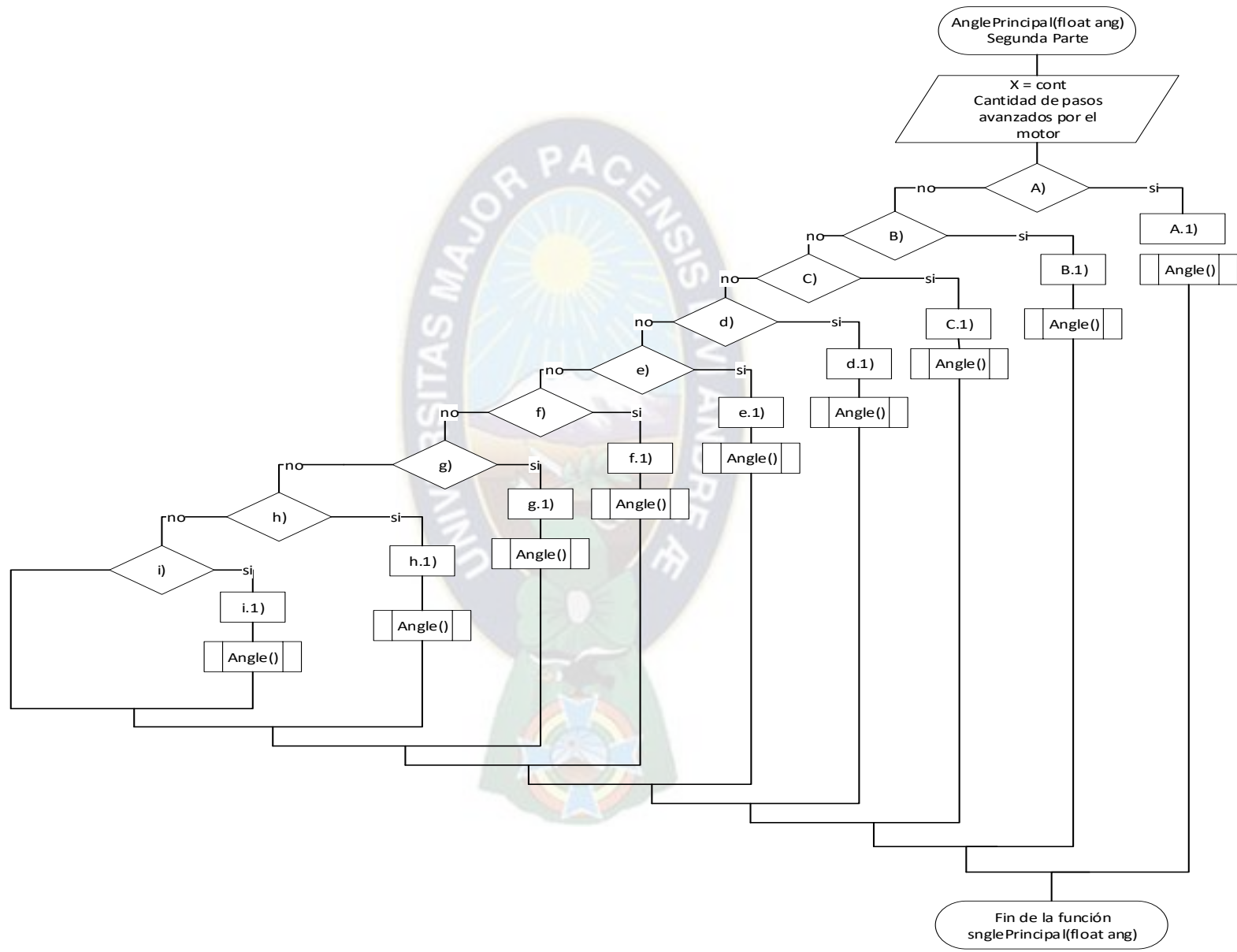
La subfunción se encarga de verificar la posición actual del motor de pasos y compararla con la posición deseada (Las posiciones de acuerdo al número de pasos). A partir de esta diferencia se tiene la cantidad de pasos restantes para llegar al punto deseado, por lo tanto, dependiendo si la cantidad de pasos actuales es mayor o menor que la cantidad de pasos deseado el motor avanzará o disminuirá en un paso hasta completar la diferencia.

Lo anteriormente explicado se plasma en un diagrama de flujo a continuación:



**Fig. 3.20.** Diagrama de Flujo – Función “angle(long x, long act)”  
**Fuente:** Elaboración Propia mediante herramienta Microsoft Office Visio 2016

Ahora se puede añadir esta subfunción a la función “anglePrincipal(float ang).  
 El diagrama de flujo que se muestra a continuación resume todo lo explicado.



**Fig. 3.21.** Diagrama de Flujo – Función “anglePrincipal(float ang)”  
**Fuente:** Elaboración Propia mediante herramienta Microsoft Office Visio 2016

La forma de llamar a esta función es la siguiente:

```
anglePrincipal(ang);
```

Siendo la variable “ang” es el valor del ángulo deseado.

#### ***d. Función - sensores(int g)***

Esta función es utilizada en el Laboratorio 1. Permite obtener la lectura digital de cada conmutador láser.

Una esfera cruzará el carril de principio a fin y activará cada conmutador desde el primero hasta el último, esto para encontrar la aceleración de la esfera en una superficie inclinada sin fricción.

Cada vez que se corta el haz de luz en cada conmutador láser se debe registrar un tiempo, usando el primero como la referencia de partida y el último para terminar el programa.

El programa está configurado para detectar cambios de estado a muy alta velocidad, por lo que es necesario añadir un detector de cambio de estado.

Se explicará la detección de los dos primeros conmutadores láser, con esto se podrá deducir fácilmente para todos los conmutadores.

Es necesario primero inicializar las siguientes variables locales:

```
float Millis0;  
float Millis1;  
  
float t0;  
float t1;  
  
int lastSensorState0 = 1;  
int lastSensorState1 = 1;  
  
int SensorState0 = 0;  
int SensorState1 = 0;
```

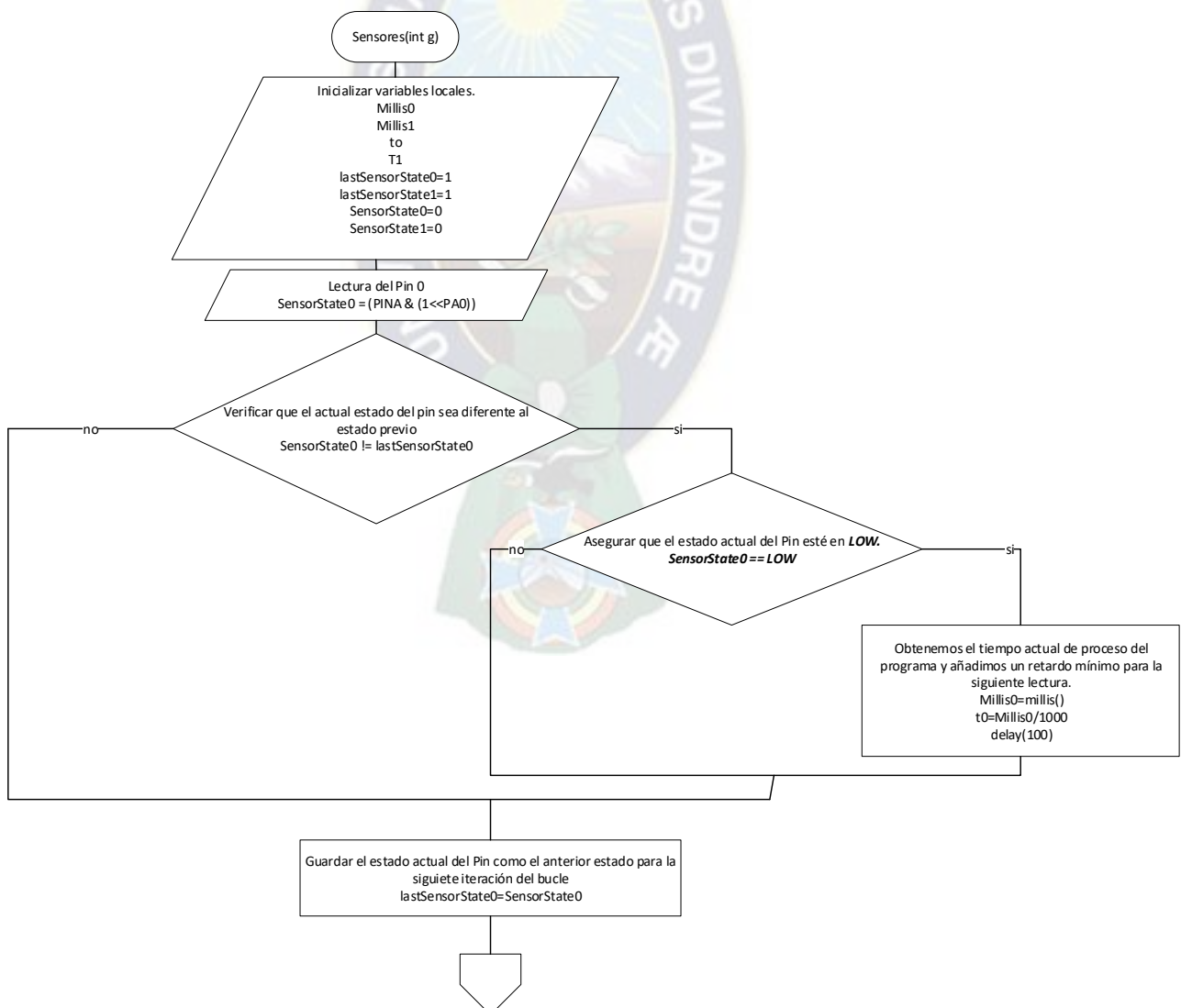
**Fig. 3.22.** Variables locales “anglePrincipal(float ang)”  
**Fuente:** Elaboración Propia mediante herramienta IDE de Arduino V1.6.8

Se inicia con la lectura del Pin en el cual está conectado el primer conmutador, este nos servirá como referencia de inicio de tiempo. La lectura se guarda en la variable “SensorState”, que guarda el estado actual del Pin.

El programa consulta si el estado actual del Pin es el mismo que el estado anterior, guardado en la variable “lastSensorState”, si es diferente, entonces aún no existe cambio de estado.

La función “Millis” devuelve el valor en milisegundos desde que el programa se ejecuta. La primera lectura de la función “millis” dará el tiempo actual. La diferencia de la siguiente lectura con la primera dará el tiempo transcurrido. Se hace este mismo procedimiento para cada conmutador.

Se explica lo anterior en el diagrama a continuación:

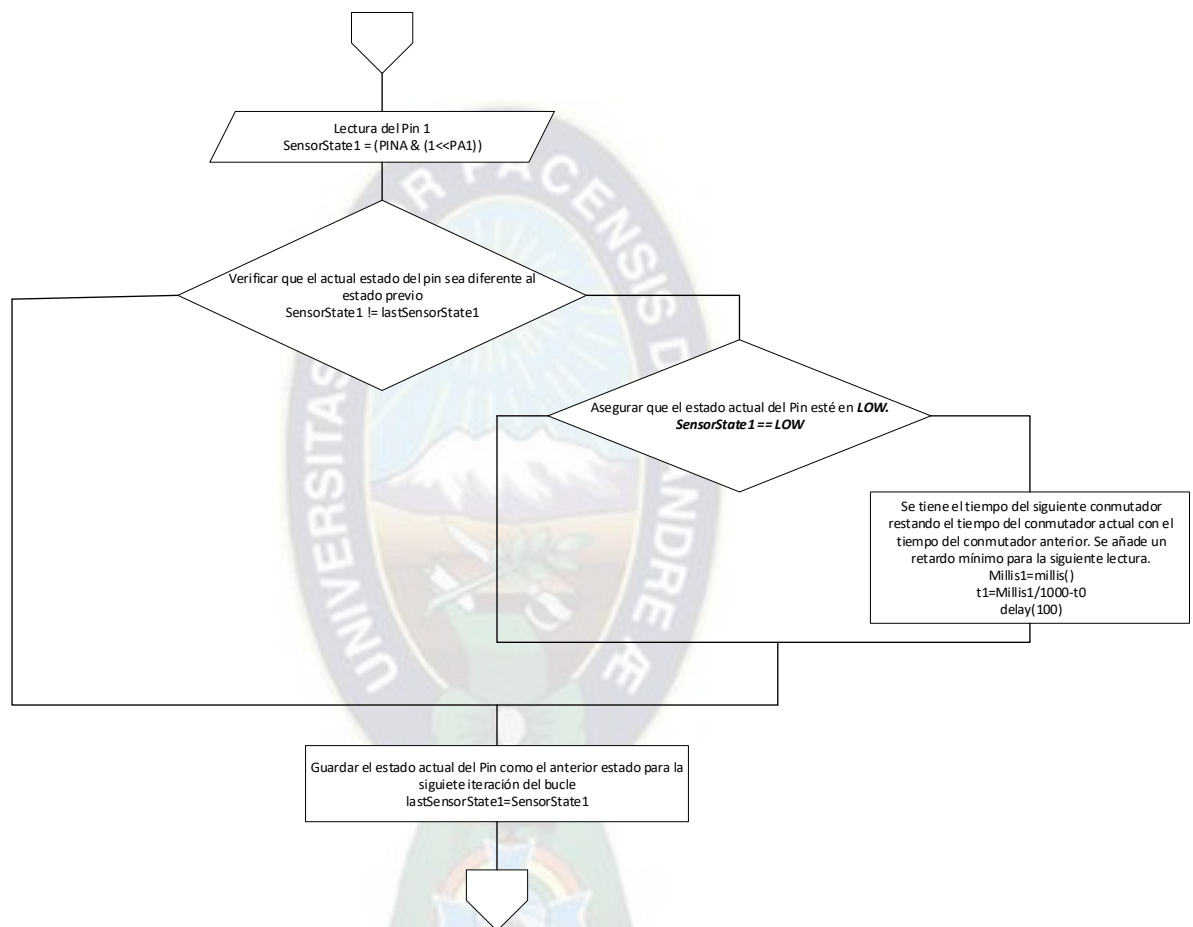


**Fig. 3.23.** Diagrama de flujo – Función “Sensores(int g) – Primer conmutador  
Fuente: Elaboración Propia mediante herramienta Microsoft office Visio 2016

Como se mencionó anteriormente, al activarse el primer conmutador se guarda el tiempo actual para dar un tiempo de inicio al recorrido.

A partir de la activación del segundo conmutador, se realiza la diferencia en los tiempos para encontrar el tiempo transcurrido desde la activación del primer conmutador.

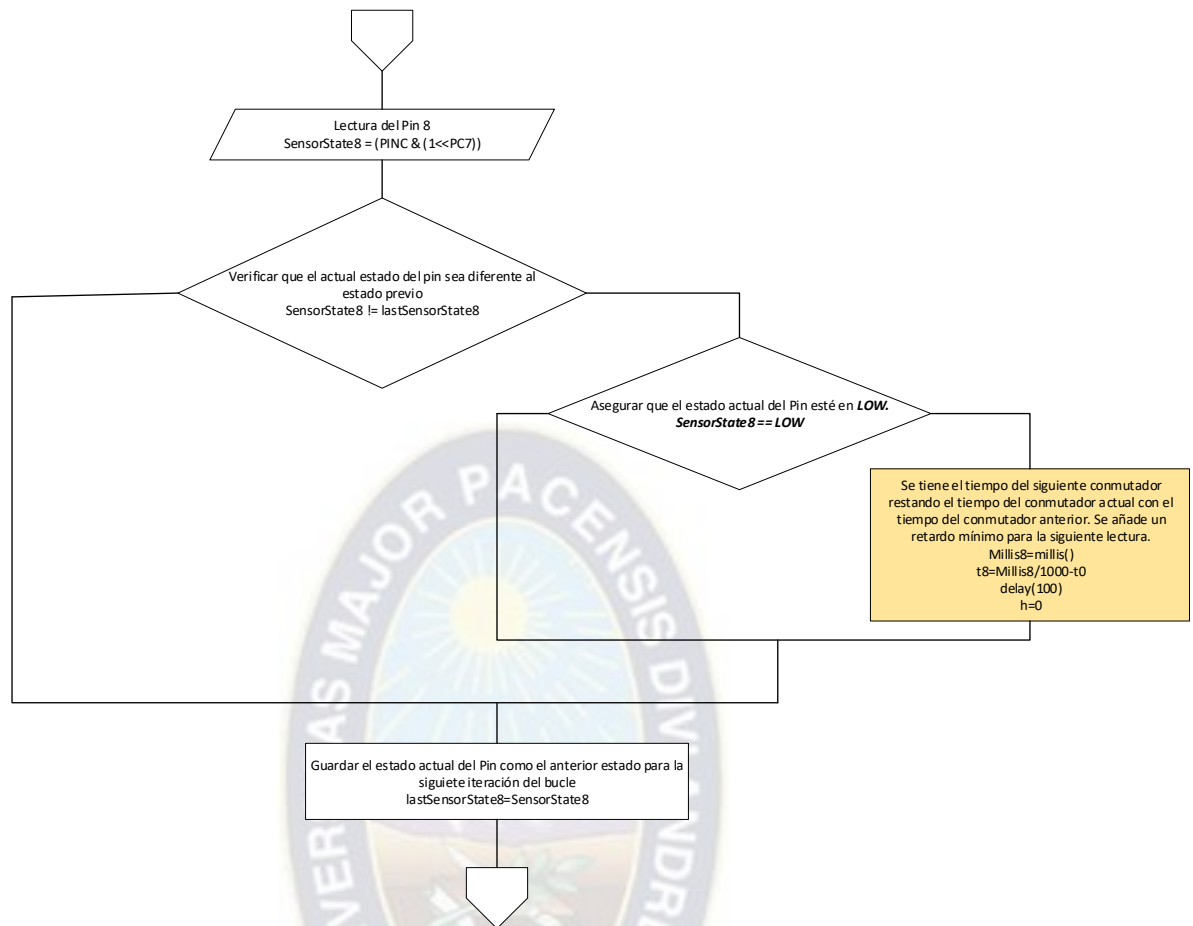
Eso se muestra en el diagrama de flujo que representa el segundo conmutador.



**Fig. 3.24.** Diagrama de flujo – Función “Sensores(int g) – Segundo conmutador  
**Fuente:** Elaboración Propia mediante herramienta Microsoft office Visio 2016

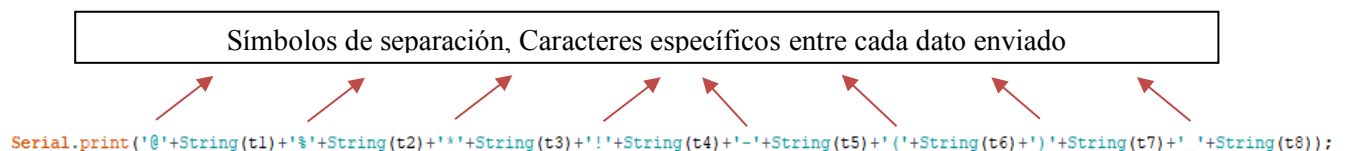
Una vez que la esfera llega al último conmutador la función debe detenerse por lo que la misma está condicionada por un bucle While interno. La variable que condiciona este bucle, es “h=1”. Por lo tanto, para salir de la función, será necesario cambiar el valor de h a “h=0”.

Esto se ve en el diagrama de flujo que representa el último conmutador:



**Fig. 3.25.** Diagrama de flujo – Función “Sensores(int g) – Último conmutador  
**Fuente:** Elaboración Propia mediante herramienta Microsoft office Visio 2016

Es importante enviar la información por el puerto serial separada con diferentes símbolos o caracteres específicos, estos símbolos permitirán la fácil lectura y orden al ser leídos por el programa Labview. Esto será explicado en la siguiente sección.



**Fig. 3.26.** Función “Sensores(int g) – Envío de datos por el puerto Serial  
**Fuente:** Elaboración Propia mediante herramienta IDE de Arduino V1.6.8

El diagrama de flujo que resume el comportamiento de todos los conmutadores se muestra en la figura 3.30.



**Fig. 3.27.** Diagrama de flujo – Función “Sensores(int g)  
**Fuente:** Elaboración Propia mediante herramienta Microsoft office Visio 2016



La forma en que se llama a esta función es la siguiente:

```
sensores(1);
```

El “uno” dentro de la función da el valor inicial a la variable local “h”. La cual, como se vio en el diagrama de flujo, debe tener un valor de “uno” para poder ingresar al bucle while.

### **e. Función - ValAngulo(int a)**

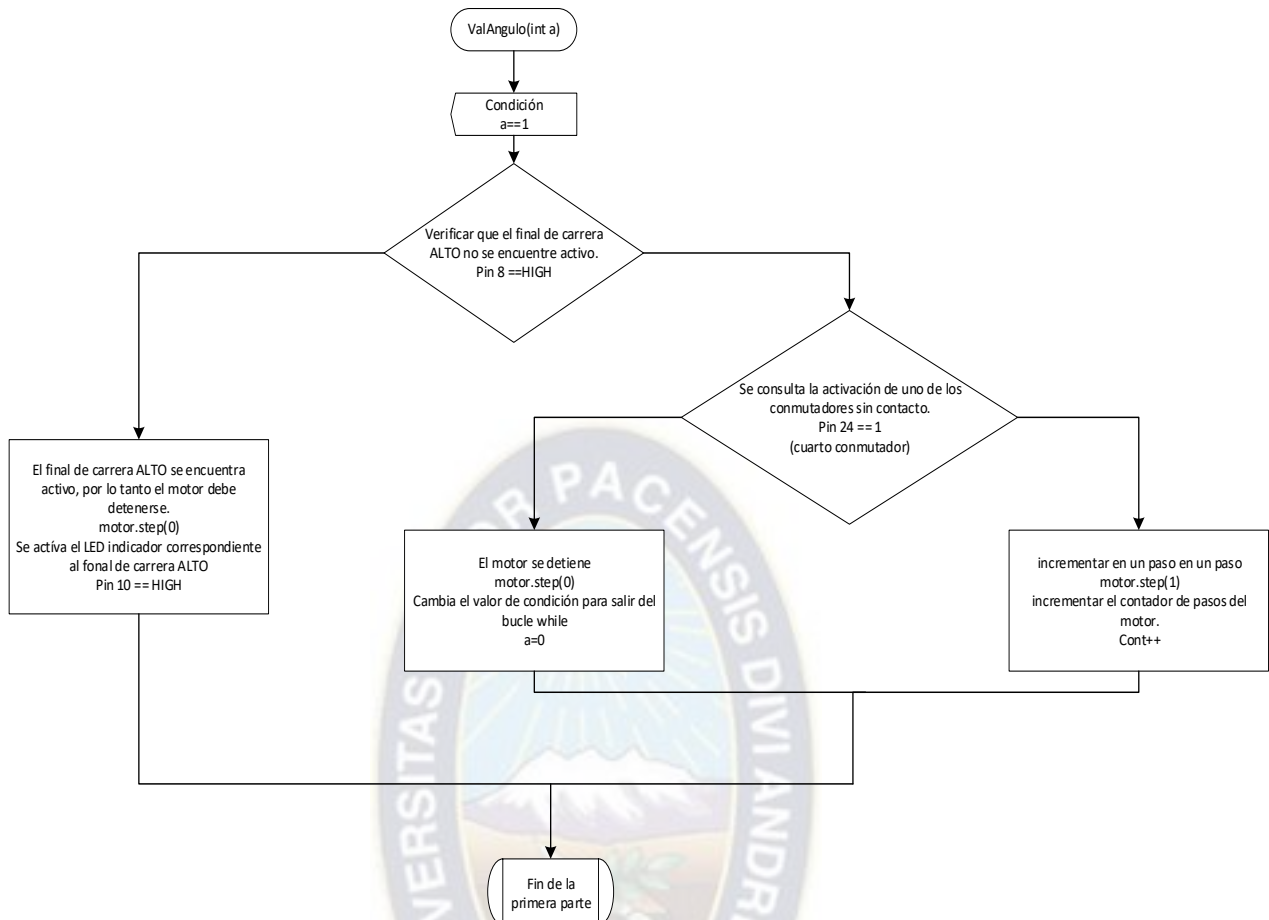
Con esta función es posible determinar el valor de un ángulo de acuerdo al número de pasos avanzados por el motor.

Se divide la función en dos partes para su apropiada explicación.

La primera parte permite el incremento de pasos del motor hasta que un conmutador sin contacto, previamente escogido, se active. Esto quiere decir que un bloque, al deslizarse debido al ángulo de inclinación, se interpondrá entre el haz de luz del láser y el LDR. Inmediatamente el motor debe detenerse.

Es necesario asegurar que la plataforma móvil no llegue a la posición superior final, por lo que se debe, primeramente, verificar que el final de carrera superior no se encuentre activo. Verificada la condición anterior, se procede al movimiento del motor, este incrementará pasos hasta que envíe señal el conmutador, ya que su haz de luz habrá sido interrumpido, por lo que se sabrá que el bloque inició el deslizamiento.

Lo descrito anteriormente se encuentra plasmado en el diagrama de flujo que sigue:



**Fig. 3.28.** Diagrama de flujo – Función “ValAngulo(int a)”  
**Fuente:** Elaboración Propia mediante herramienta Microsoft office Visio 2016

Se define la siguiente Variable local usada en esta función:

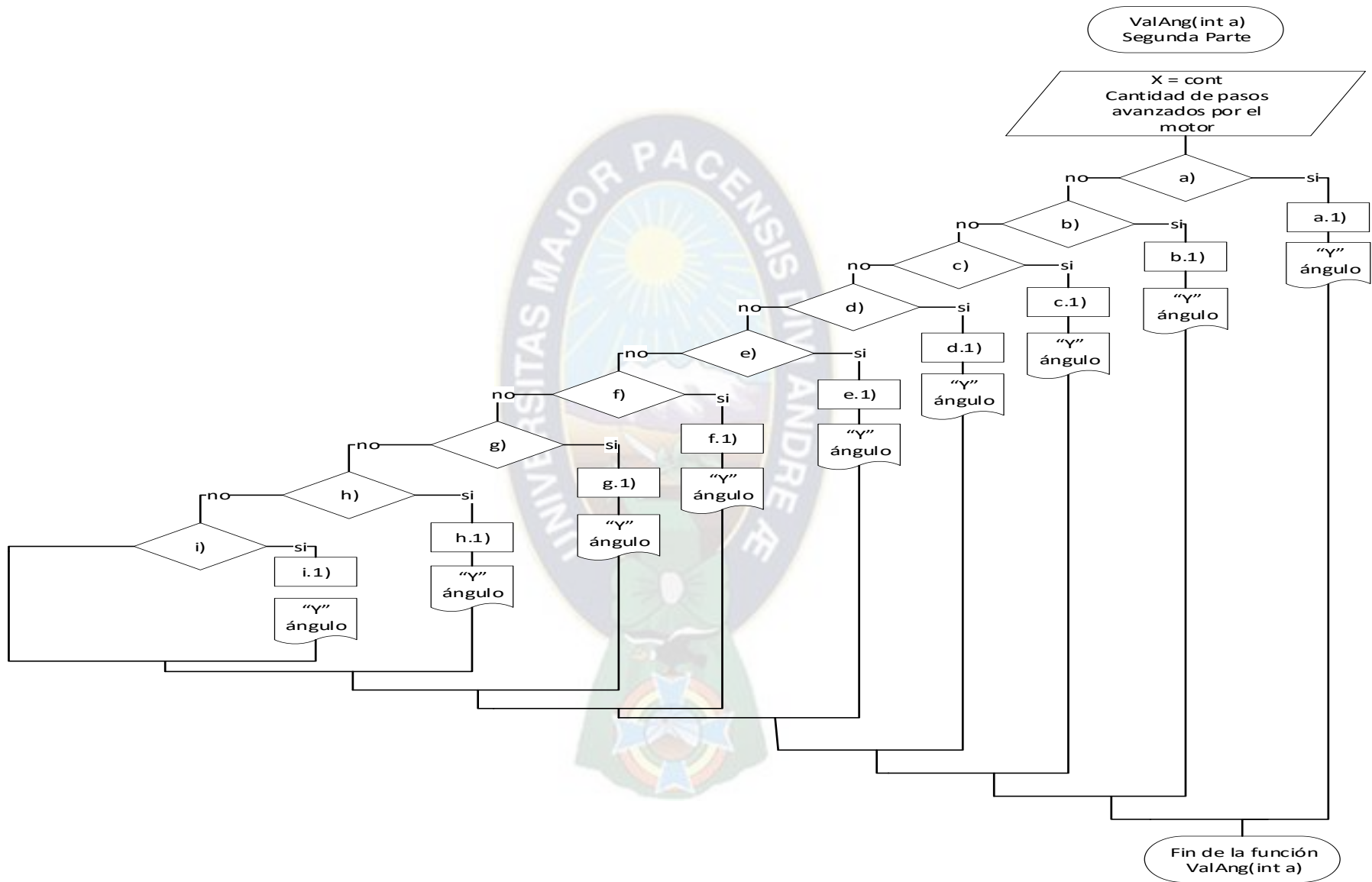
- a – Variable local, definida tipo “int”. Usada para usar como condicionante del bucle while. Asegura la ejecución del bucle hasta que cambie su valor.

La segunda parte de esta función se encarga de transformar los pasos acumulados en el ángulo crítico que se desea obtener.

Para encontrar los ángulos a partir de la cantidad de los pasos avanzados por el motor solamente se necesita despejar todas las fórmulas mencionadas la Tabla 3.4. del apartado **Función–anglePrincipal(float g).**

<i>Intervalo</i>	<i>Fórmula de pasos</i>	<i>N° Ec</i>
$0 \leq x \leq 10000$	$y = \frac{4,55}{10000} * x$	3.3.10
$10000 < x \leq 20000$	$y = \left( \frac{10000}{4,15} * 4,65 - 10000 + x \right) * \frac{4,15}{10000}$	3.3.11
$20000 < x \leq 30000$	$y = \left( \frac{10000}{4,45} * 8,8 - 20000 + x \right) * \frac{4,45}{10000}$	3.3.12
$30000 < x \leq 40000$	$y = \left( \frac{10000}{4} * 13,25 - 30000 + x \right) * \frac{4}{10000}$	3.3.13
$40000 < x \leq 50000$	$y = \left( \frac{10000}{4,15} * 17,25 - 40000 + x \right) * \frac{4,15}{10000}$	3.3.14
$50000 < x \leq 60000$	$y = \left( \frac{10000}{3,9} * 21,4 - 50000 + x \right) * \frac{3,9}{10000}$	3.3.15
$60000 < x \leq 70000$	$y = \left( \frac{10000}{3,4} * 25,3 - 60000 + x \right) * \frac{3,4}{10000}$	3.3.16
$70000 < x \leq 80000$	$y = \left( \frac{10000}{3,05} * 28,7 - 70000 + x \right) * \frac{3,05}{10000}$	3.3.17
$80000 < x \leq 87000$	$y = \left( \frac{10000}{2,25} * 31,75 - 80000 + x \right) * \frac{2,25}{10000}$	3.3.18

**Tabla. 3.5.** Fórmula de ángulo por intervalo  
**Fuente:** Elaboración Propia Herramienta-Excel 2016

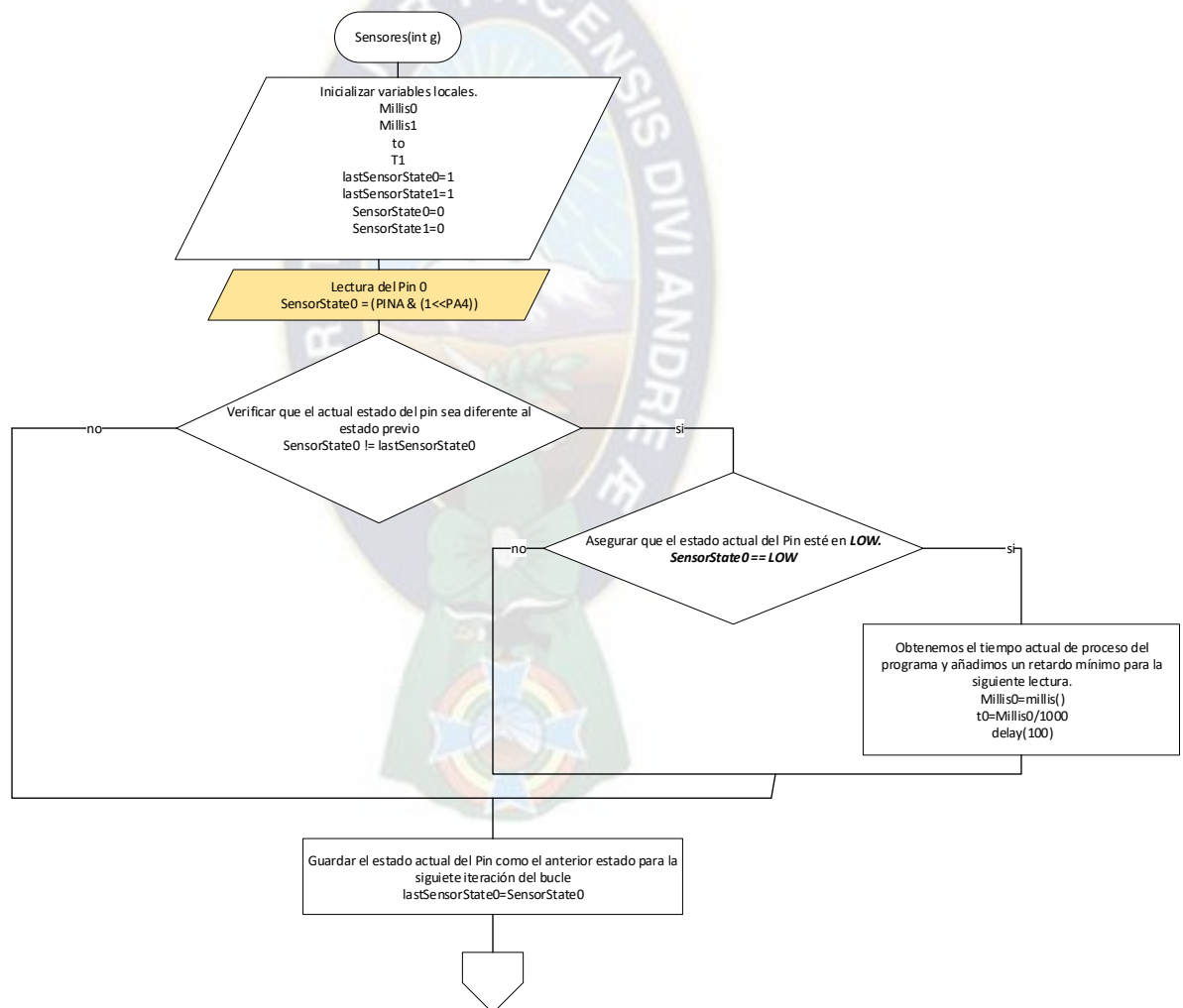


**Fig. 3.29.** Diagrama de flujo – Función “ValAngulo(int a)” - Segunda parte  
**Fuente:** Elaboración Propia mediante herramienta Microsoft office Visio 2016

## f. Función – Lab3(int h)

Esta función cumple el mismo principio que la función “*Sensores(int g)*”, las únicas diferencias son; el orden en el que se activan los conmutadores y la cantidad de conmutadores usados. Para este caso se usarán únicamente cinco conmutadores.

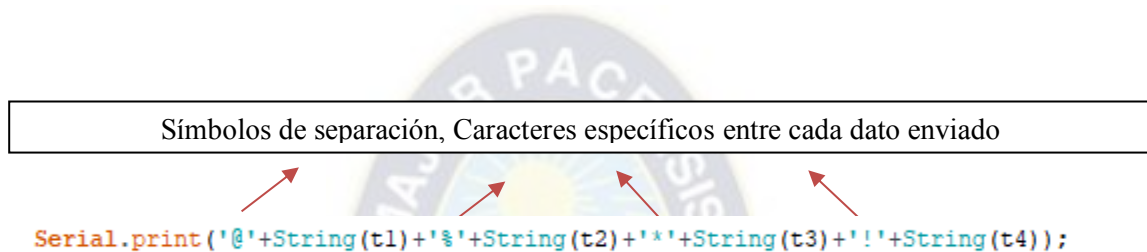
El conmutador ‘uno’, en este caso, será el último conmutador y se encargará de finalizar la función. El conmutador ‘cinco’ se encargará de dar inicio al temporizador. Al igual que en la función “*Sensores(int g)*” es necesario un detector de estados o flancos.



**Fig. 3.30.** Diagrama de flujo – Función “Lab3(int h)” – Conmutador 5  
**Fuente:** Elaboración Propia mediante herramienta Microsoft office Visio 2016

Como se comentó anteriormente, la primera lectura necesaria es la del conmutador 5 conectado al Pin 4 del puerto A. El orden de activación de los sensores usados en este caso son la única variación del programa con respecto al funcionamiento de la función “Sensores(int a)”.

Es importante enviar la información por el puerto serial separada con diferentes símbolos o caracteres específicos, estos símbolos permitirán la fácil lectura y orden al ser leídos por el programa Labview. Esto será explicado en la siguiente sección.



**Fig. 3.31.** Función “Lab3(int h) – Envío de datos por el puerto Serial  
**Fuente:** Elaboración Propia mediante herramienta IDE de Arduino V1.6.8

A continuación, se presenta el diagrama de flujo tomando en cuenta los 5 sensores utilizados.



**Fig. 3.32.** Diagrama de flujo – Función “Lab3(int h)”  
**Fuente:** Elaboración Propia mediante herramienta Microsoft office Visio 2016

La forma de llamar a esta función es la siguiente:

Lab3(1);

El “uno” dentro de la función da el valor inicial a la variable local “h”. La cual, como se vio en el diagrama de flujo, debe tener un valor de “uno” para poder ingresar al bucle while.

### 3.3.4 Programa Principal

El programa principal espera recibir instrucciones desde la interfaz de usuario mediante el puerto USB.

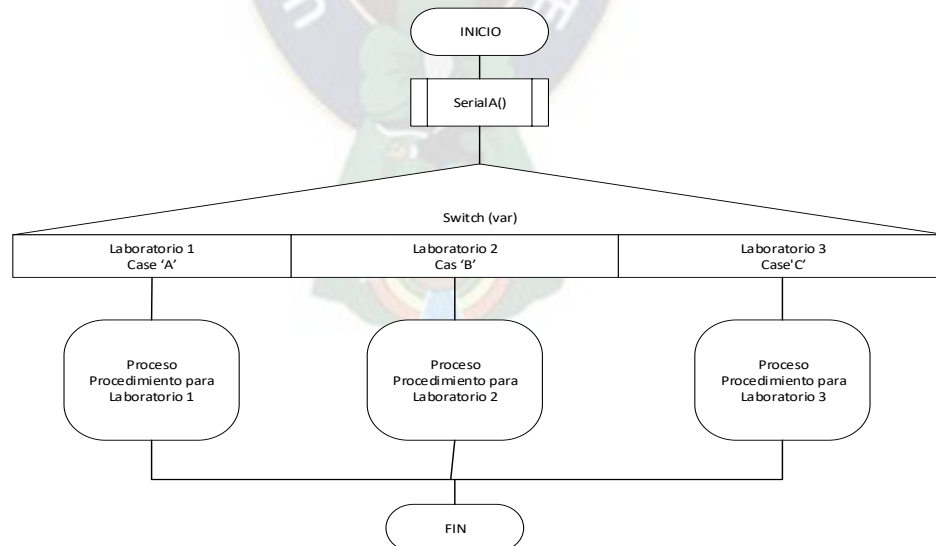
Es importante en cada caso iniciar cada laboratorio con la función “reset(0)” pues es necesario tener el contador inicializado.

Las instrucciones recibidas, leídas por el puerto serial serán caracteres que el programa principal deberá interpretar para hacer la selección respectiva.

Se toma en cuenta lo siguiente:

- A partir de la lectura serial:
  - Caracter ‘A’ – Elige el Laboratorio 1
  - Caracter ‘B’ – Elige el Laboratorio 2
  - Caracter ‘C’ – Elige el Laboratorio 3

Se resume esta primera parte en el diagrama de flujo siguiente:



**Fig. 3.33.** Diagrama de flujo – Función Principal - General  
**Fuente:** Elaboración Propia mediante herramienta Microsoft office Visio 2016



Se explicará cada proceso por partes para un claro entendimiento del programa principal.

### **a. Proceso, procedimiento para Laboratorio 1**

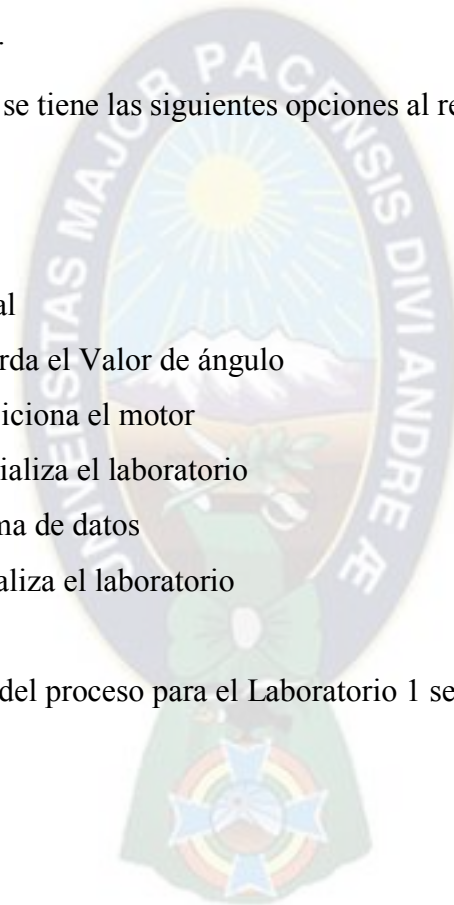
El Laboratorio 1 está diseñado para hallar la aceleración de una masa sin fricción, por eso se utiliza una esfera que posee mínimo contacto con la superficie.

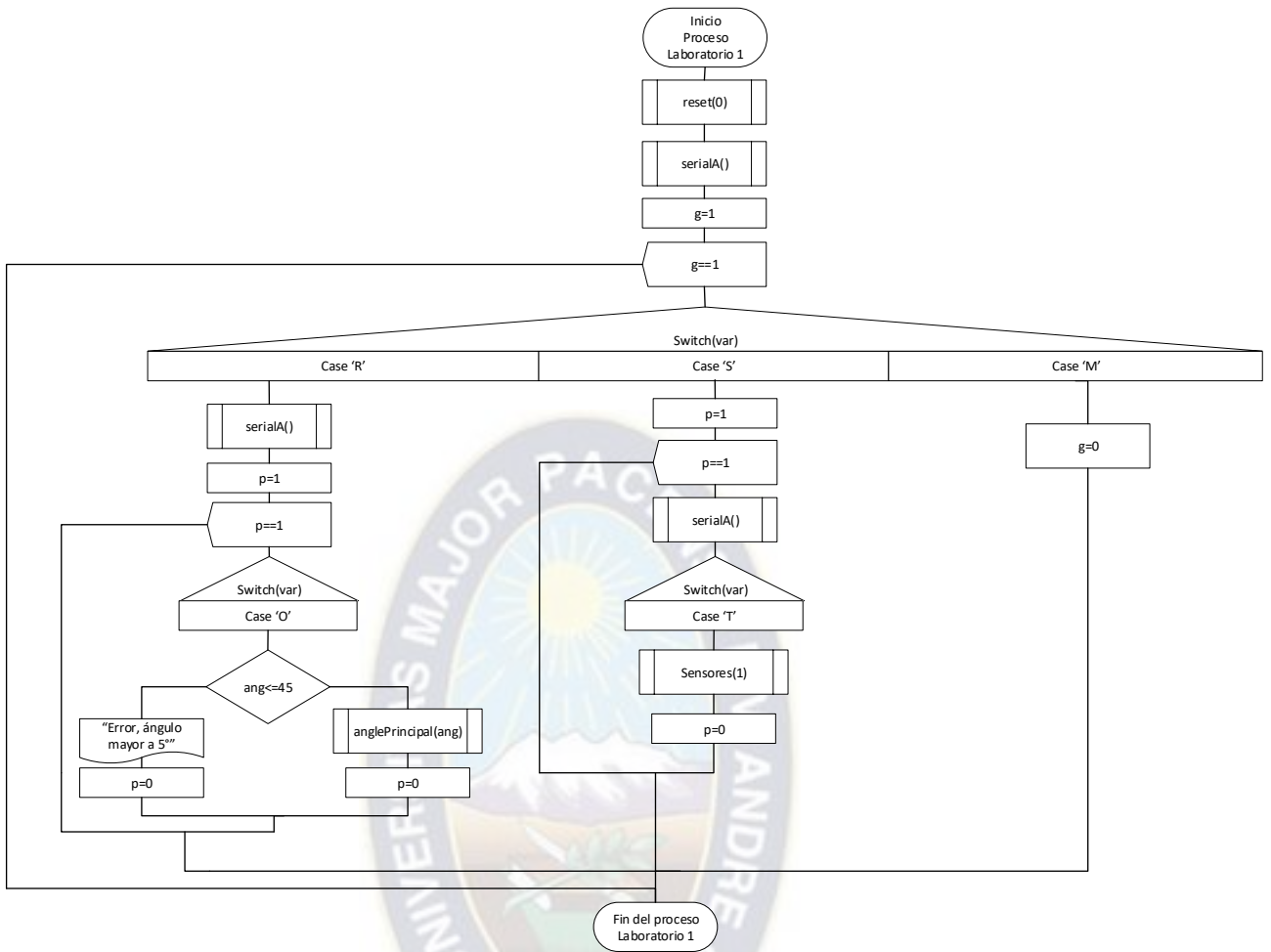
Para esto es necesaria la lectura de todos los conmutadores láser. Además, se restringe el ángulo, sólo llegará a 5°, a mayor ángulo, la esfera incrementará mucho su aceleración con posibles fallas en la lectura.

Al ingresar al laboratorio 1 se tiene las siguientes opciones al realizar la lectura por el puerto serial:

- A partir de la lectura serial
  - Caracter 'R' – Guarda el Valor de ángulo
  - Caracter 'O' – Posiciona el motor
  - Caracter 'S' – Inicializa el laboratorio
  - Caracter 'T' – Toma de datos
  - Carácter 'M' – Finaliza el laboratorio

El funcionamiento general del proceso para el Laboratorio 1 se encuentra en el siguiente diagrama de flujo:





**Fig. 3.34.** Diagrama de flujo – Función Principal – Laboratorio 1  
**Fuente:** Elaboración Propia mediante herramienta Microsoft office Visio 2016

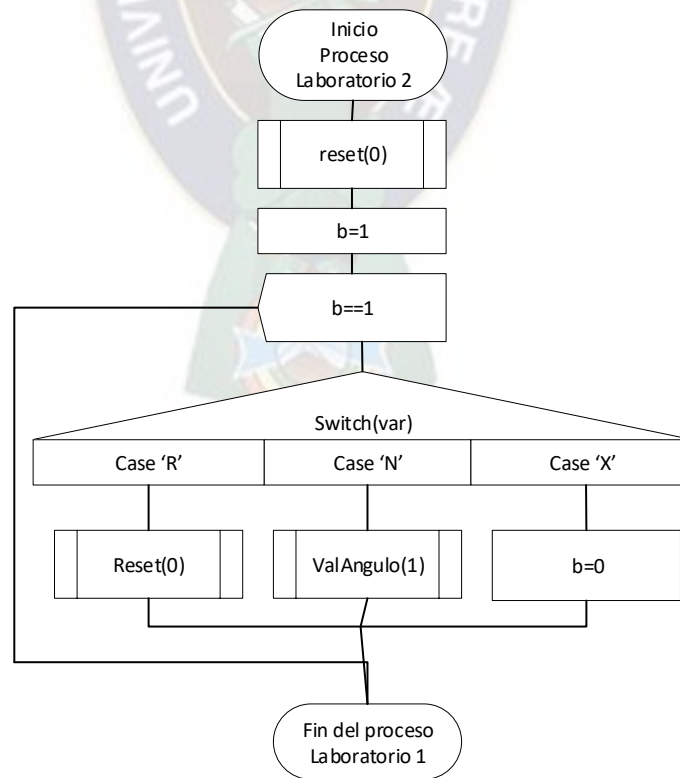
## b. Proceso, procedimiento para Laboratorio 2

El Laboratorio 2 está diseñado para hallar el coeficiente de fricción estático. Para ello se utiliza un solo conmutador láser ubicado justo después del bloque para que, cuando el bloque inicie su movimiento, el conmutador láser se active inmediatamente deteniendo, a su vez, el motor. La inclinación de la plataforma móvil tendrá el valor del ángulo crítico necesario para hallar el coeficiente de fricción estática.

Al ingresar al laboratorio 2 se tiene las siguientes opciones al realizar la lectura por el puerto serial:

- A partir de la lectura serial
  - Caracter 'N' – Iniciar programa
  - Caracter 'R' – Resetear posición del motor
  - Caracter 'X' – Finalizar el laboratorio

El funcionamiento general del proceso para el Laboratorio 1 se encuentra en el siguiente diagrama de flujo:



**Fig. 3.35.** Diagrama de flujo – Función Principal – Laboratorio 2  
**Fuente:** Elaboración Propia mediante herramienta Microsoft office Visio 2016

### c. Proceso, procedimiento para Laboratorio 3

El Laboratorio 3 está diseñado para hallar el coeficiente de fricción dinámico. Para ello se utilizan dos bloques unidos mediante un cable (de masa despreciable) y una polea. El bloque 1 se encuentra en contacto directo con la superficie de la plataforma móvil y el bloque 2 se encuentra colgando, a partir de la polea.

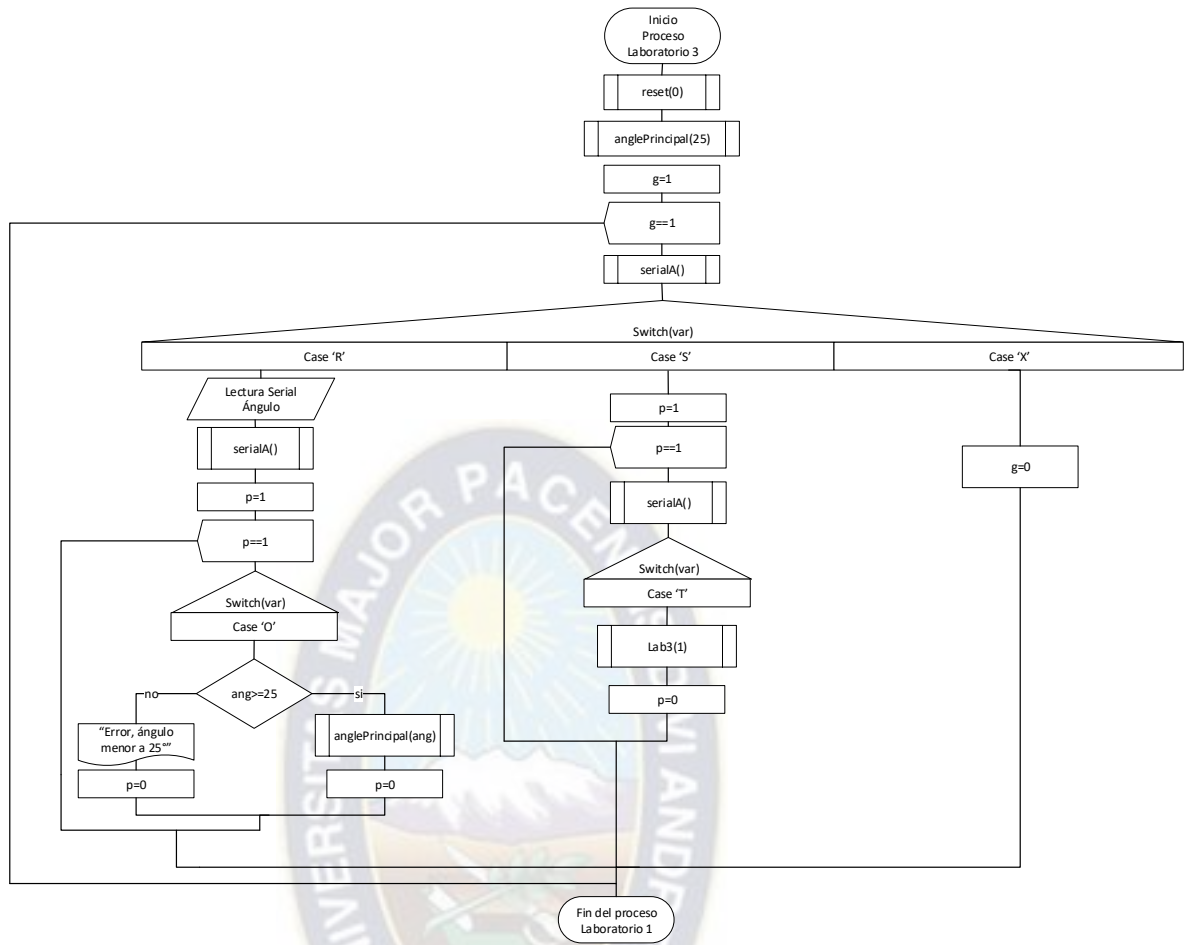
Para iniciar este laboratorio es necesario que la plataforma tenga un ángulo inicial, en este caso será de  $25^\circ$ , ya que el bloque 2 requiere del espacio suficiente para llegar a la base del prototipo.

El bloque inicializará la lectura a partir del quinto conmutador, finalizando la misma en el primer conmutador.

Al ingresar al laboratorio 2 se tiene las siguientes opciones al realizar la lectura por el puerto serial:

- A partir de la lectura serial
  - Caracter 'R' – Guardar valor de ángulo
  - Caracter 'O' – Posicionar motor
  - Caracter 'S' – Inicializar el laboratorio
  - Caracter 'T' – Toma de datos
  - Caracter 'X' – Finalizar laboratorio

El funcionamiento general del proceso para el Laboratorio 1 se encuentra en el siguiente diagrama de flujo:



**Fig. 3.36.** Diagrama de flujo – Función Principal – Laboratorio 3  
**Fuente:** Elaboración Propia mediante herramienta Microsoft office Visio 2016

### 3.4 Programación en LabView

#### 3.4.1. Laboratorio 1

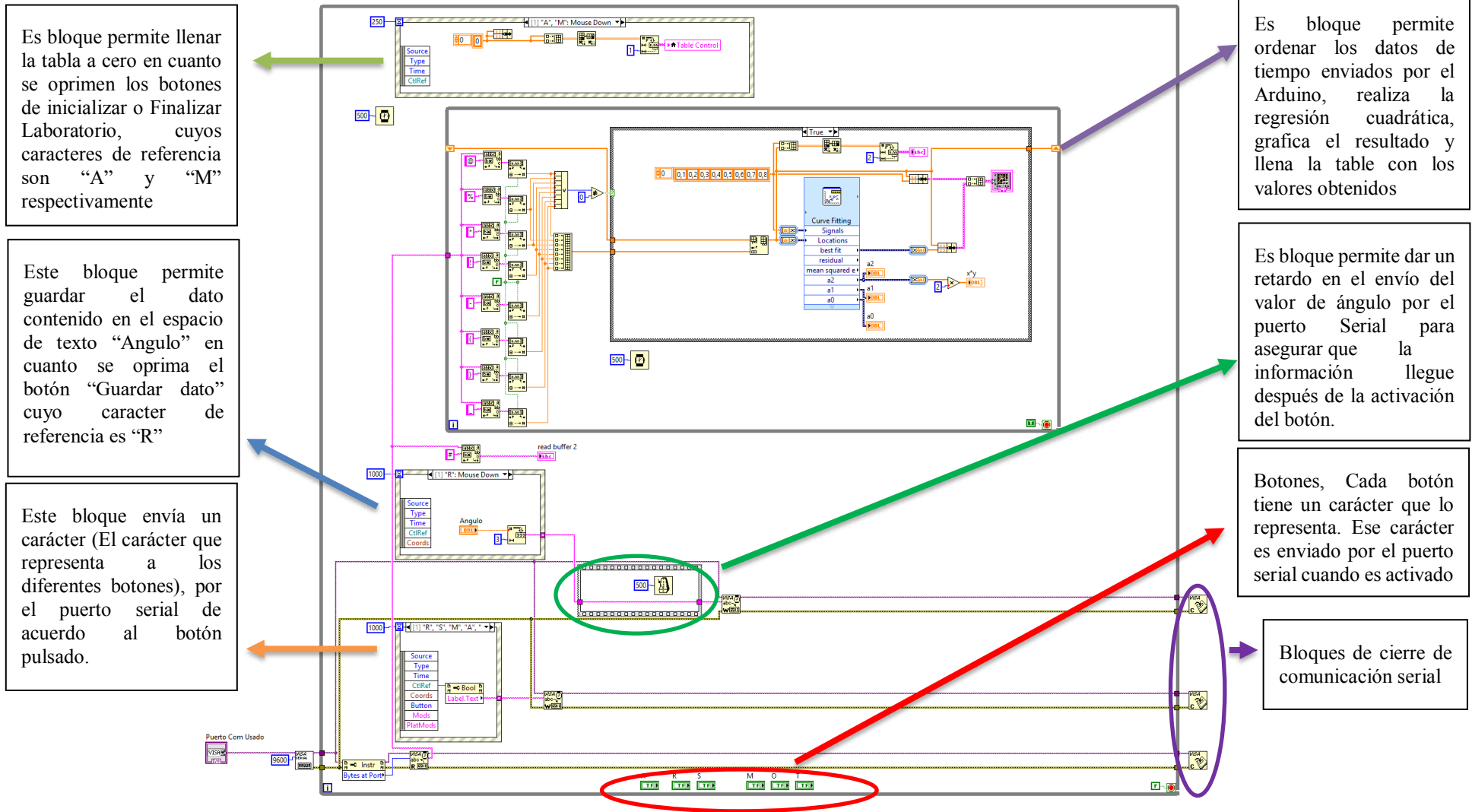
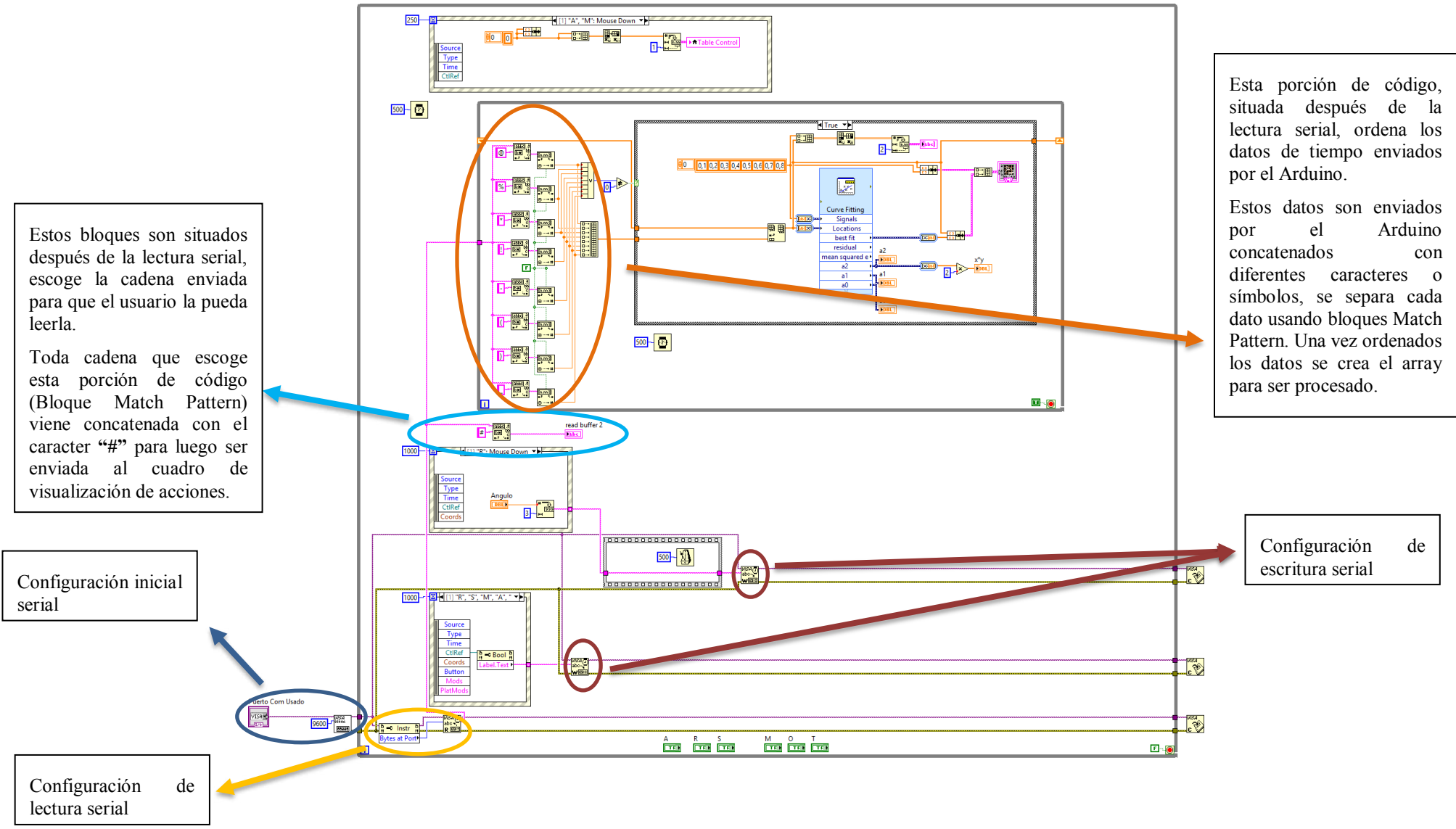
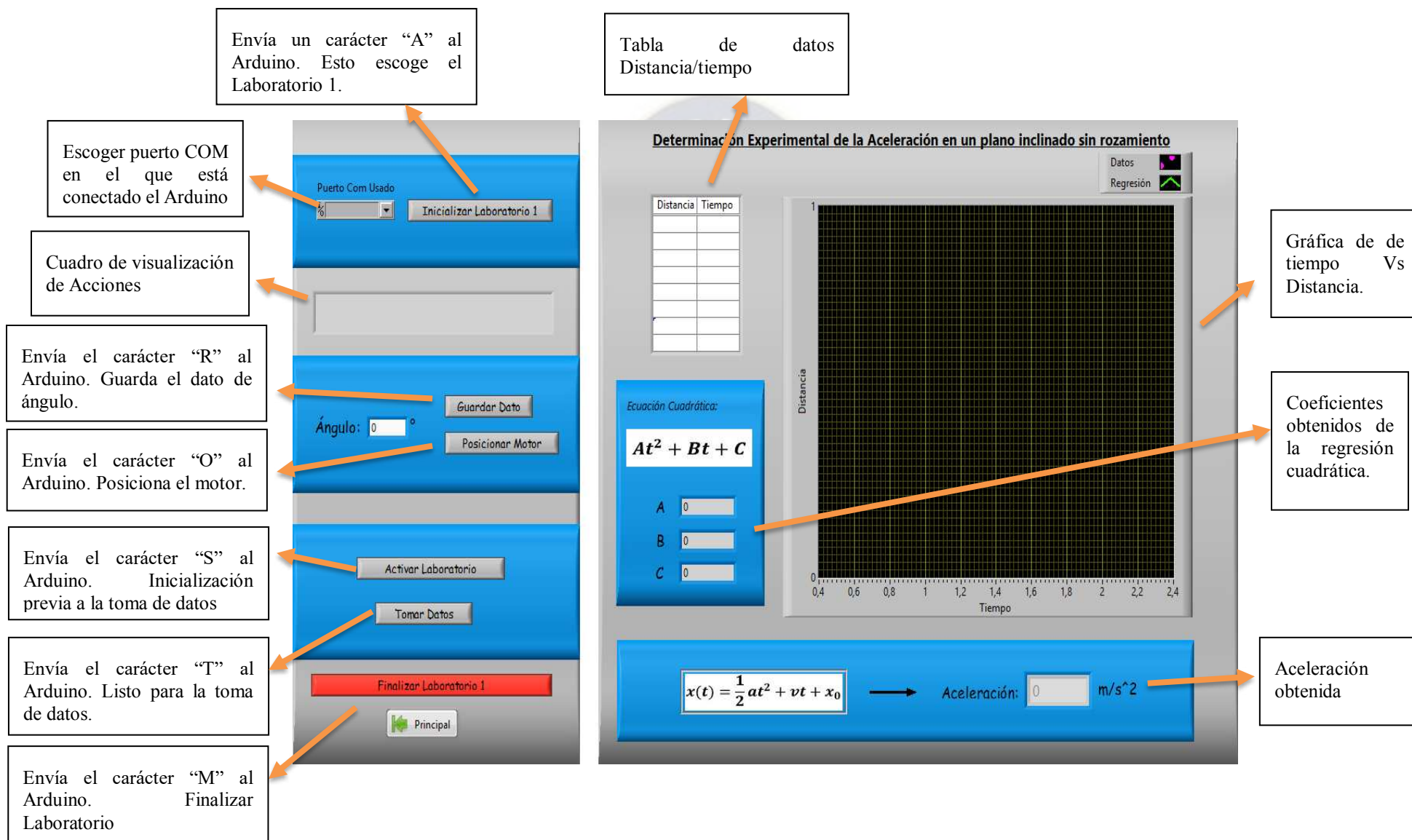


Fig. 3.37. Diagrama de Bloques – Laboratorio 1  
Fuente: Elaboración Propia mediante herramienta LabVIEW V15.0



**Fig. 3.38.** Diagrama de Bloques – Laboratorio 1  
**Fuente:** Elaboración Propia mediante herramienta LabVIEW V15.0





**Fig. 3.39.** Panel Frontal – Laboratorio 1  
**Fuente:** Elaboración Propia mediante herramienta LabVIEW V15.0



### 3.4.2. Laboratorio 2

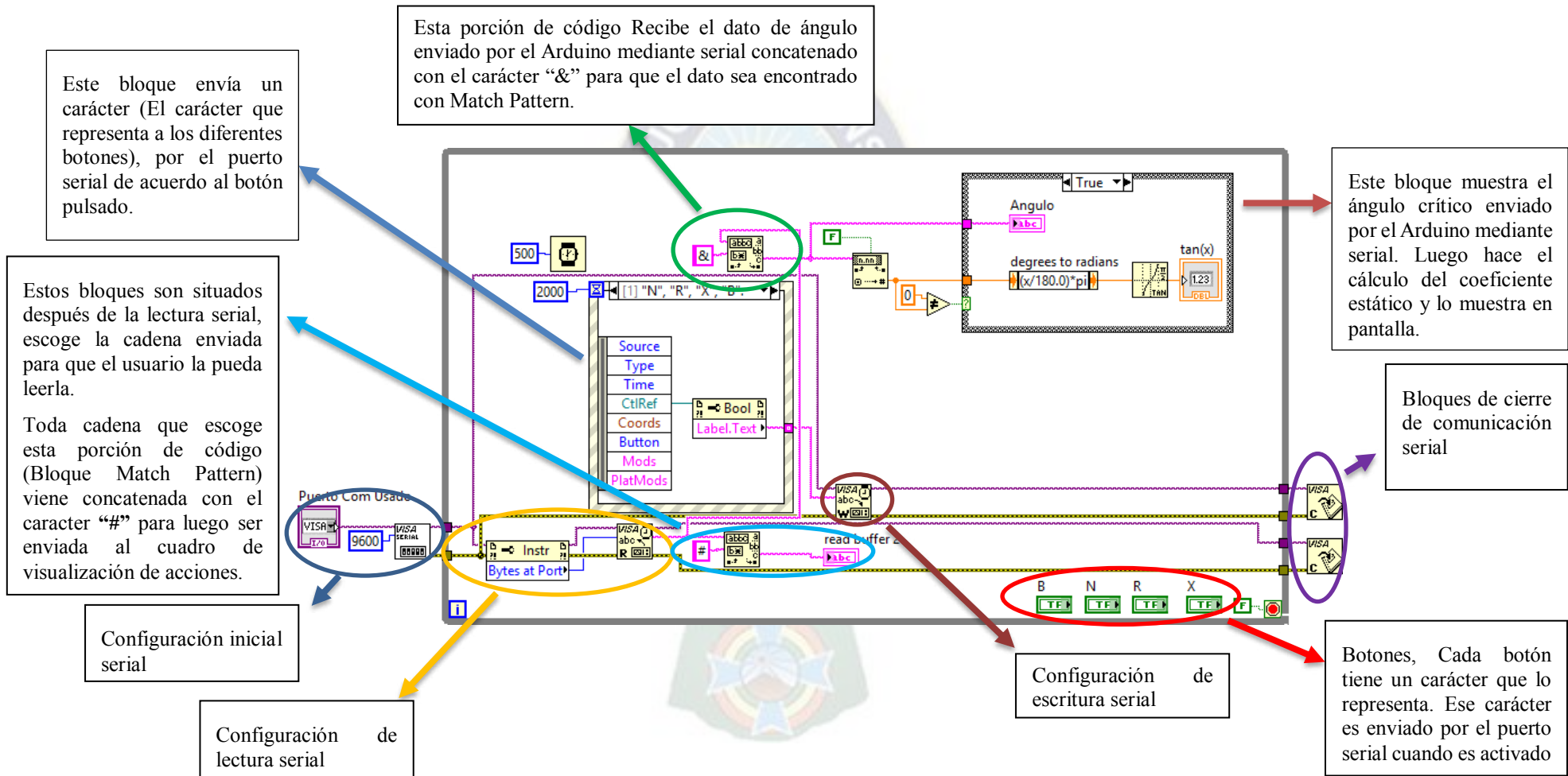


Fig. 3.40. Diagrama de Bloques – Laboratorio 2  
 Fuente: Elaboración Propia mediante herramienta LabVIEW V15.0



**Fig. 3.41.** Panel Frontal – Laboratorio 2  
**Fuente:** Elaboración Propia mediante herramienta LabVIEW V15.0

### 3.4.3. Laboratorio 3

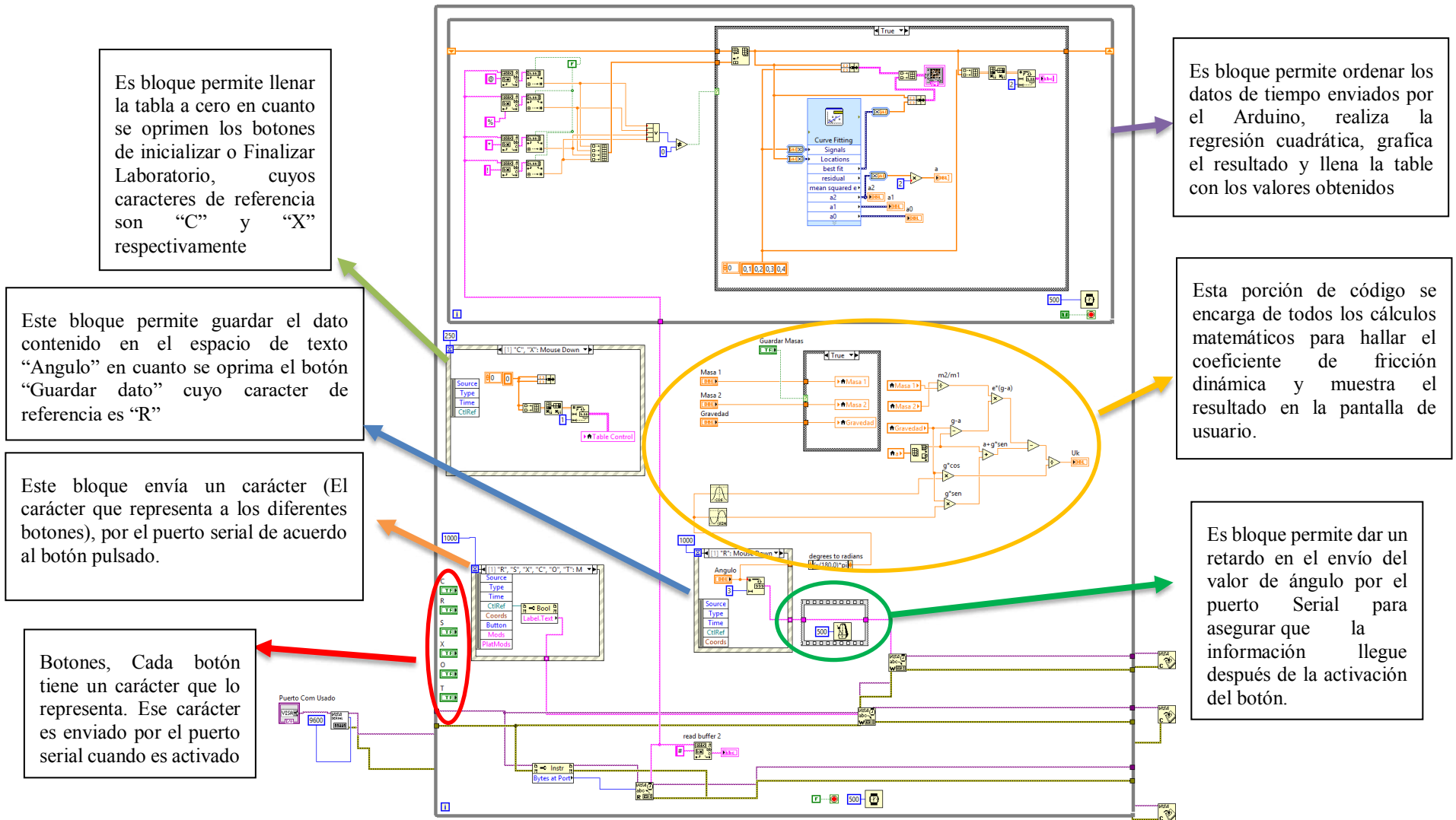


Fig. 3.42. Diagrama de Bloques – Laboratorio 3  
Fuente: Elaboración Propia mediante herramienta LabVIEW V15.0

Estos bloques son situados después de la lectura serial, escoge la cadena enviada para que el usuario la pueda leerla.

Toda cadena que escoge esta porción de código (Bloque Match Pattern) viene concatenada con el caracter “#” para luego ser enviada al cuadro de visualización de acciones.

Esta porción de código, situada después de la lectura serial, ordena los datos de tiempo enviados por el Arduino.

Estos datos son enviados por el Arduino concatenados con diferentes caracteres o símbolos, se separa cada dato usando bloques Match Pattern. Una vez ordenados los datos se crea el array para ser procesado.

Configuración de lectura serial

Configuración de escritura serial

Configuración inicial serial

Bloques de cierre de comunicación serial

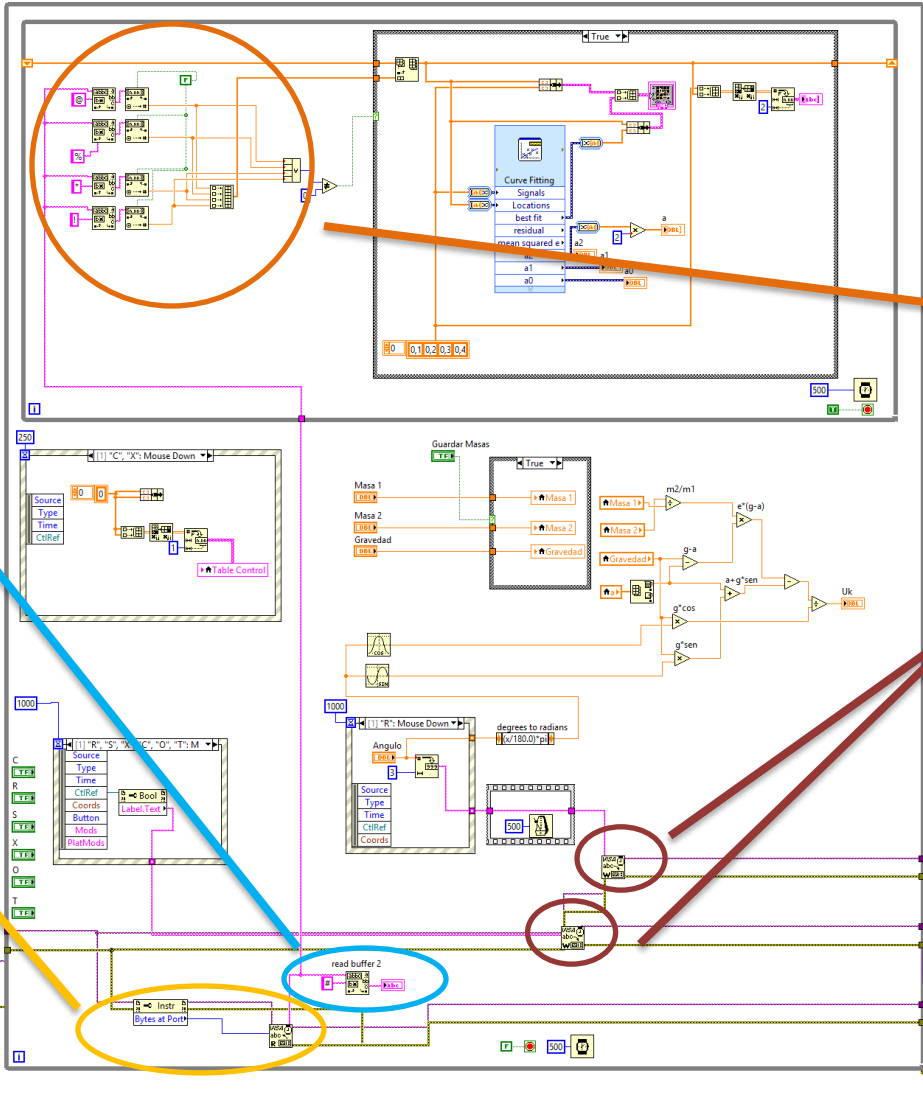
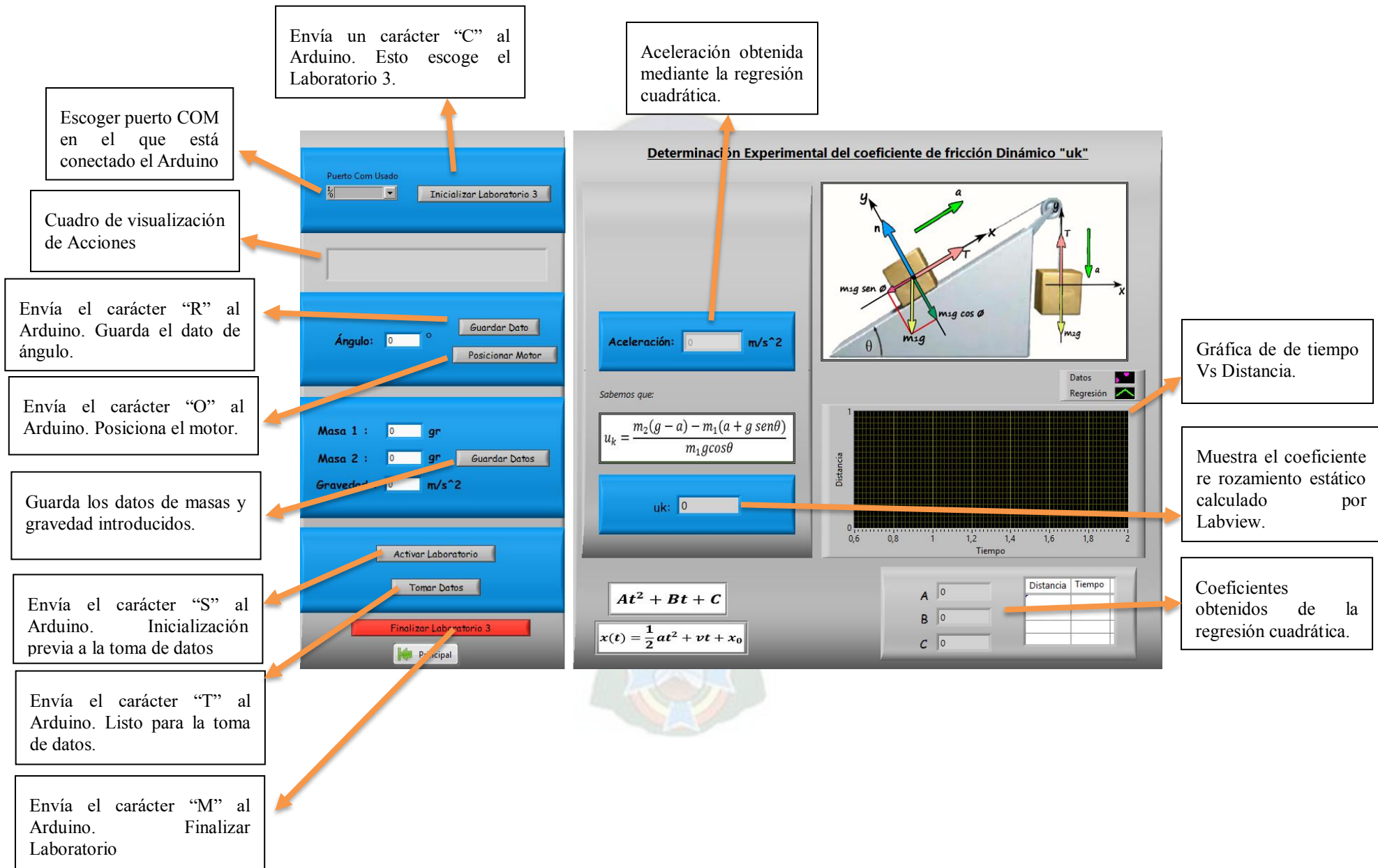


Fig. 3.43. Diagrama de Bloques – Laboratorio 3  
Fuente: Elaboración Propia mediante herramienta LabVIEW V15.0



**Fig. 3.44.** Panel Frontal – Laboratorio 3  
**Fuente:** Elaboración Propia mediante herramienta LabVIEW V15.0

### 3.5. Pruebas y Resultados

#### a. Laboratorio 1. Determinación de aceleración en un plano inclinado sin fricción.

Ángulo - 2°		Ángulo - 3°		Ángulo - 4°		Ángulo - 5°	
N°	Aceleración m/s <sup>2</sup>	N°	Aceleración m/s <sup>2</sup>	N°	Aceleración m/s <sup>2</sup>	N°	Aceleración m/s <sup>2</sup>
1	0.2897	1	0.5953	1	0.6590	1	0.7845
2	0.2890	2	0.4896	2	0.5984	2	0.8610
3	0.3699	3	0.5944	3	0.5988	3	0.7952
4	0.3655	4	0.5854	4	0.5986	4	0.8642
5	0.2982	5	0.4670	5	0.6559	5	0.7952
6	0.2449	6	0.5698	6	0.5976	6	0.7981
7	0.3942	7	0.5896	7	0.5322	7	0.7135
8	0.3150	8	0.4637	8	0.6974	8	0.7953
9	0.3666	9	0.4688	9	0.5979	9	0.7356
10	0.2988	10	0.4963	10	0.6978	10	0.7923
0.3232		0.5320		0.6234		0.7935	

**Tabla.3.6.** Pruebas experimentales – Aceleración en un plano inclinado sin rozamiento

**Fuente:** Creación propia en Excel 2016

Para tener el resultado teórico y compararlo con el experimental se sabe que:

$$a = g \operatorname{sen}(\theta)$$

Siendo:

$g$  aceleración de gravedad en La Paz – Bolivia ( $g = 9.775 \text{ m/s}^2$ )

$\theta$  Ángulo de inclinación

Ángulo Crítico	Aceleración m/s <sup>2</sup>
2°	0.3411
3°	0.5116
4°	0.6819
5°	0.8520

**Tabla.3.7.** Obtención de valores teóricos – Aceleración sobre un plano inclinado sin rozamiento

**Fuente:** Creación propia en Excel 2016

Con esta información se pueden hallar los distintos errores como se muestran en la siguiente tabla.

Ángulo	Error absoluto del conjunto $E = \bar{x} - \mu$	Error Relativo $\varepsilon = \frac{E}{\bar{x}}$	Error Relativo Porcentual $\varepsilon\% = \varepsilon * 100$
2°	0.01793	0.055469	5.55%
3°	0.02040	0.038354	3.84%
4°	0.05852	0.093875	9.39%
5°	0.05844	0.073653	7.37%

**Tabla.3.8.** Errores – Aceleración sobre un plano inclinado sin rozamiento

**Fuente:** Creación propia en Excel 2016

## b. Laboratorio 2. Determinación del coeficiente de fricción estática

Venesta			Trupan		
N°	Ángulo [°]	us	N°	Ángulo [°]	us
1	25.96	0.486869	1	14.73	0.262905
2	25.01	0.466251	2	14.28	0.254525
3	24.56	0.456729	3	14.56	0.259598
4	24.58	0.457151	4	14.23	0.253462
5	25.16	0.469440	5	14.14	0.251792
6	25.32	0.472851	6	14.58	0.259970
7	25.62	0.479270	7	15.78	0.282444



8	24.68	0.459261	8	14.86	0.265191
9	24.95	0.464977	9	14.21	0.253091
10	24.21	0.449370	10	15.54	0.277928
<b>0.466217</b>			<b>0.262091</b>		

**Tabla.3.9.** Pruebas experimentales – Coeficiente de fricción estática

**Fuente:** Creación propia en Excel 2016

Es necesario recordar valores teóricos extraídos de tablas.

En este caso se usarán los valores de tabla de Madera tanto para Venesta y Trupán.

<i>Materiales en Contacto</i>	<i>us</i>
<i>Madera sobre Madera</i>	0,25 - 0,55

Como se puede apreciar, ambos materiales ingresan dentro del rango. La madera puede tener variedad de acabados dando a la misma diferente aspereza.

### c. Laboratorio 3. Determinación del coeficiente de fricción Dinámica

<b>Venesta</b>		<b>Trupan</b>	
<b>N°</b>	<b>uk</b>	<b>N°</b>	<b>uk</b>
1	0.4745	1	0.2047
2	0.4999	2	0.2004
3	0.4660	3	0.2222
4	0.4237	4	0.2312
5	0.4880	5	0.2086
6	0.3697	6	0.2049
7	0.3700	7	0.2048
8	0.4999	8	0.2137
9	0.3370	9	0.2366
10	0.4598	10	0.2013
<b>0.438836</b>		<b>0.212820</b>	

**Tabla.4.5.** Pruebas experimentales – Coeficiente de fricción dinámica



**Fuente:** Creación propia en Excel 2016

Datos para las pruebas en Venesta

*Masa 1 = 9gr*

*Masa 2 = 20gr*

Datos para las pruebas en Trupán

*Mesa 1 = 6gr*

*Mesa 2 = 30gr*

<i>Materiales en Contacto</i>	<i>uk</i>
<i>Madera sobre Madera</i>	0,2 - 0,4

Como se puede apreciar, ambos materiales ingresan dentro del rango. La madera puede tener variedad de acabados dando a la misma diferente aspereza.

## Capítulo 4

### Conclusiones y Recomendaciones

#### 1. Conclusiones

Se logró diseñar y construir un aparato de laboratorio micro computarizado funcional programado en Arduino con una interfaz de usuario basada en LabView para la determinación de los coeficientes de rozamiento estático, dinámico y aceleración de una masa de manera experimental para su uso en el laboratorio de Física y Control de la Facultad de Ingeniería de la Universidad Mayor de San Andrés.

Se realizó el diseño y montaje de la electrónica.

Se desarrollaron los programas de control de actuadores y sensores en el microcontrolador Arduino MEGA 2560.

Se logró desarrollar la implementación de la interfaz gráfica de usuario en el programa LabVIEW.

Se logró la correcta integración de todos los subsistemas.

Se desarrollaron ejercicios experimentales y se logró una comparación con datos teóricos.

#### 2. Recomendaciones

Se debe considerar que el diseño del circuito de los detectores laser se realizó tomando en cuenta un ambiente con intensidad de luminarias de 100Watts. Si los sensores están sometidos a la intensidad lumínica del Sol no funcionan correctamente.

Mantener todos los pernos y tuercas del prototipo siempre ajustados ya que pueden existir movimientos relativos entre la torre y la base que hacen variar el ángulo de inclinación de la plataforma móvil.

Mantener lubricados los dientes de la barra roscada y las guías de deslizamiento.

Antes de iniciar los laboratorios verificar que los detectores láser se encuentren correctamente alineados.

Los sensores de posición con láser varían en la detección de acuerdo a la cantidad de luz del ambiente, por lo tanto, es aconsejable el tener un sistema programable que se ajuste automáticamente a la luz. Un sistema programado que varíe las resistencias del circuito interruptor del LDR para que se ajuste al tipo de luz que se tenga en el entorno.



# Parte III

## 1. Análisis de la actividad

### 1.1 Desempeño laboral

Creo haberme desempeñado bien y profesionalmente en el ámbito laboral. Cada etapa avanzada fue una enseñanza y posibilidad de mejora.

El pasar por varias áreas me permitió aprender más y tener una visión amplia y general en cada área, sin embargo, recomiendo a estudiantes que crezcan en una sola rama para poder especializarse desde el inicio.

### 1.2 Formación recibida en la UMSA

La formación recibida en la Facultad de Ingeniería y en la Carrera de Ingeniería Electrónica permitieron que pase por varios sectores en el ámbito laboral. Pasé tanto por las áreas de sistemas, telecomunicaciones y control.

El conocimiento adquirido en materias de instrumentación fueron una excelente base en campo, sin embargo, creo necesario más enseñanza en electrónica de potencia para la mención de Sistemas de Control.

## Bibliografía

- Serway, Raymond A.; Beichner, Robert J.(2002): “*Física para Ciencias e Ingeniería- TOMO I*” . México, D.F., McGrawHill
- Resnik, Robert; Halliday, David; Krane, Kenneth S. (1998): “*Física*”. México, CECSA
- Sears, Mark W.; Zemansky, Mark W; Young, Hugh D.; Freeman, Rogger A.(1996): “*Física Universitaria*”. México, D.F. , Addison Wesley Longman
- Warren, Jhon-David; Adams, Josh; Molle, Harald (2009): “*Arduino Robotics*”
- Garrido Pedraza, Javier : “*Fundamentos de Arduino*”
- Torrente Artero, Oscar(2013): “*Arduino, Curso Práctico de Formación*”. México D.F., Alfaomega Grupo Editor
- W. Larsen, Ronald (2011): “*LabVIEW for Engineers*”. USA, New Jersey, Pearson Education, Inc.
- Lajara Viscaino, José Rafael; Pelegrí Sebastiá, José (2007): “*LabView, Entorno gráfico de Programación*”. México D.F. , Alfaomega Grupo Editor
- Halvorsen, Hans-Petter (2012): “*Introduction to LabVIEW*”.
- National Instruments Corporation (1996): “*LabVIEW Tutorial Manual*”. USA, Austin. National Instruments
- Holguín Londoño, Germán Andrés; Pérez Londoño, Sandra Milena; Orozco Gutierrez, Álvaro Ángel (2002); “*Curso Básico LabVIEW 6i*”. Colombia, Pereira. Universidad Tecnológica de Pereira
- Sinclair, Ian R. (2001): “*Sensors and Transducers*”. Great Britain, Newnes.
- Gibilisco, Stan(2001): “*The Illustrated Dictionary of Electronics*”.USA, New York. McGraw Hill

## ANEXO A

### Reporte Fotográfico del Aparato

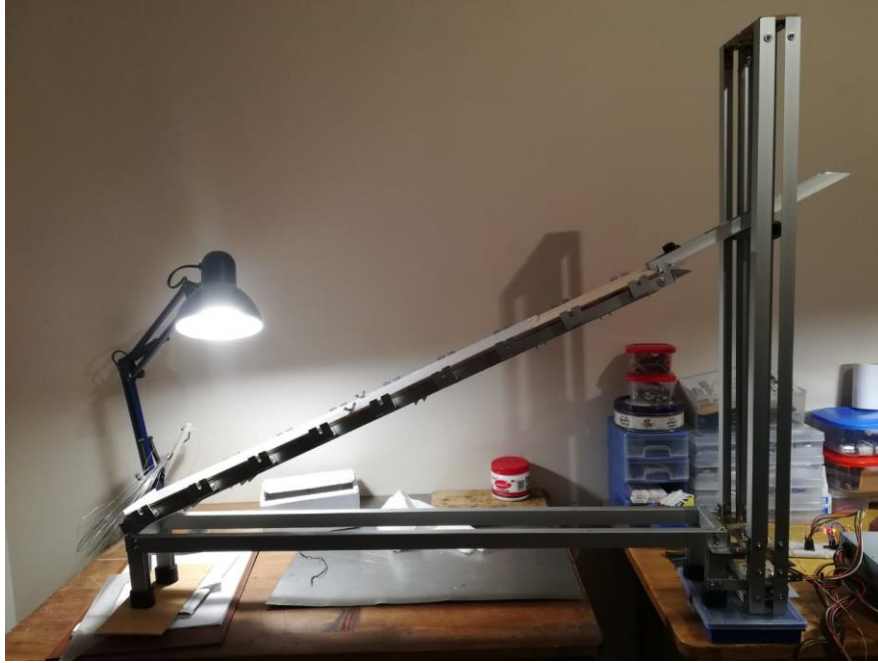
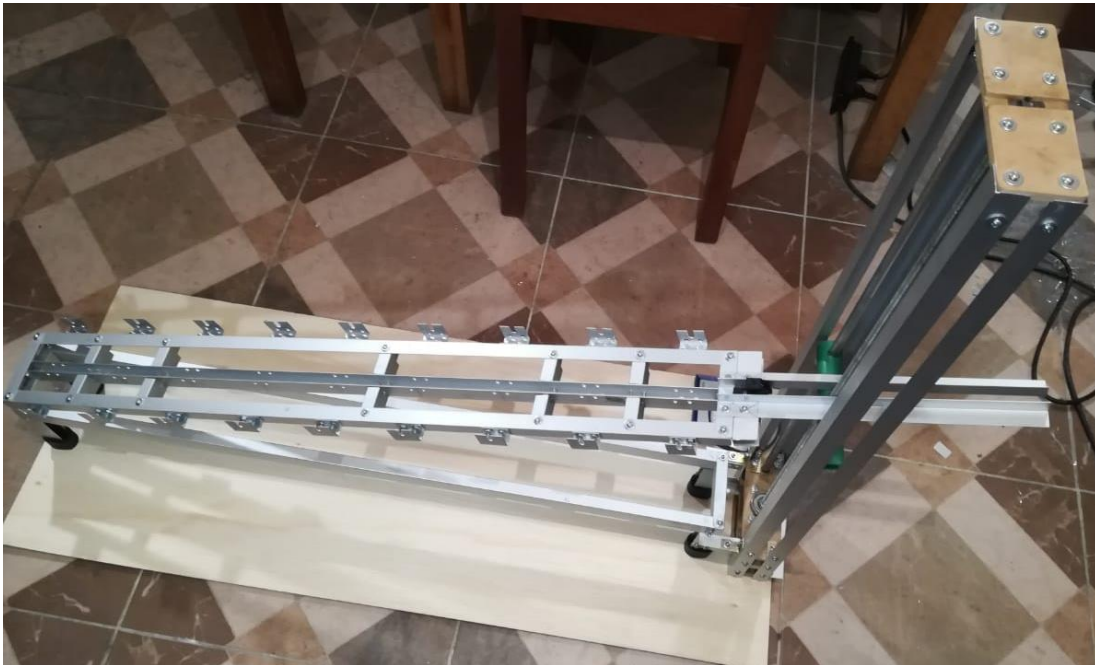


Fig. A.1. Maqueta Completa - Construcción terminada (Vista Lateral)



Fig. A.2. Maqueta Completa - Construcción terminada (Sin Sensores)



**Fig. A.3.** Maqueta Completa - Construcción terminada (Vista Superior Lateral)



**Fig. A.4.** Maqueta Completa - Construcción terminada (Vista Superior Frontal)



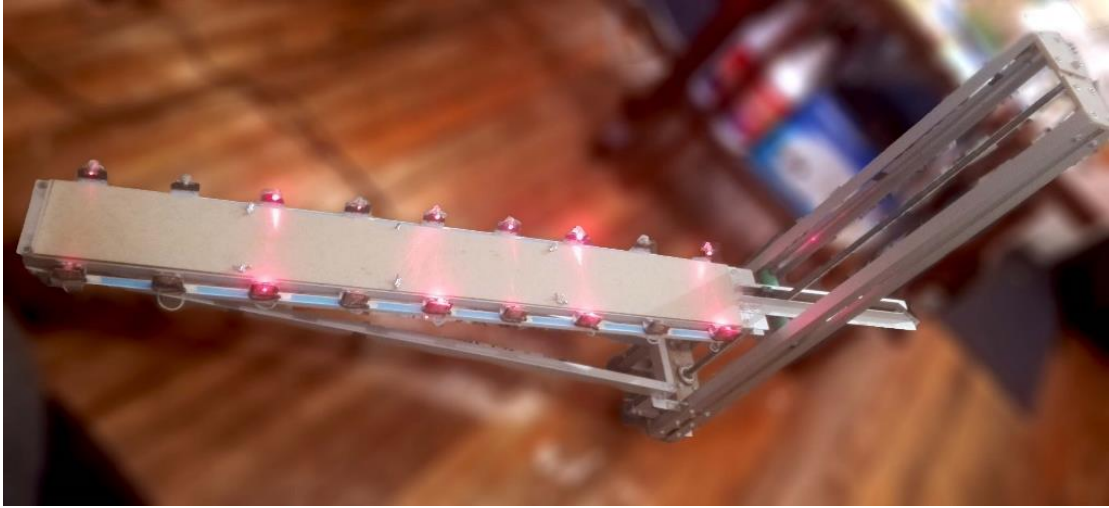


**Fig. A.5.** Maqueta Completa – Construcción Terminada sin Sensores

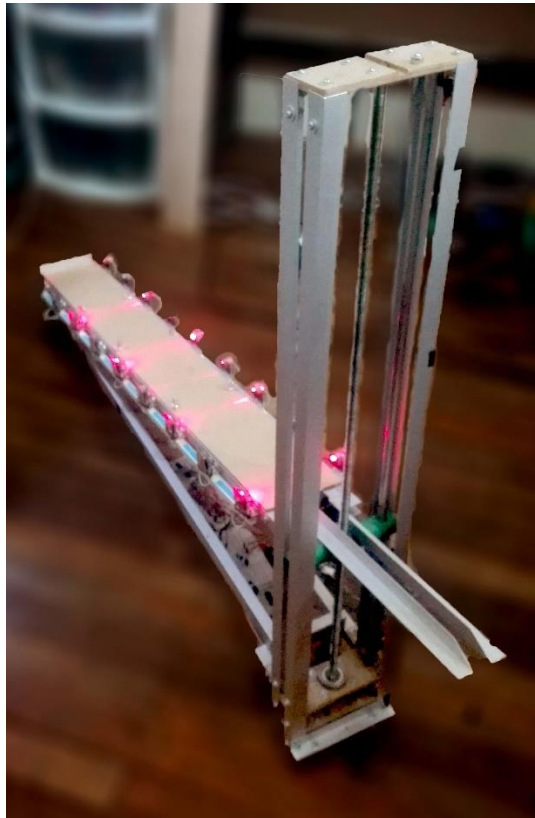


**Fig. A.6.** Torre Principal – Construcción

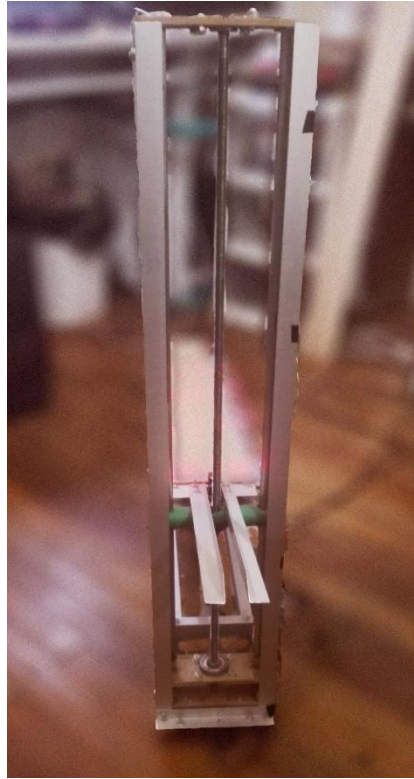




**Fig. A.7.** Maqueta Completa - Sensores Láser Activos



**Fig. A.8.** Maqueta Completa - Sensores Láser Activos (Vista de Torre)



**Fig. A.9.** Torre Principal

## ANEXO B

### Casos Experimentales

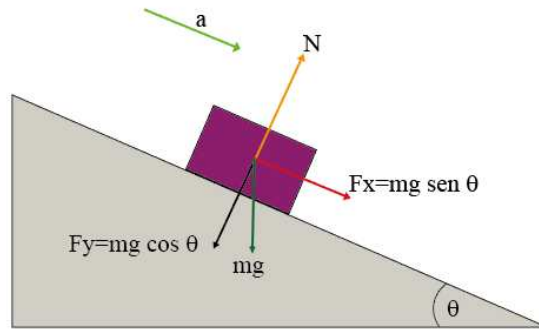
Materiales en Contacto	$\mu_s$	$\mu_k$
Acero sobre hielo	0,03	0,02
Acero sobre Teflón	0,04	0,04
Teflón sobre Teflón	0,04	0,04
Hielo sobre Hielo	0,1	0,03
Acero sobre Acero	0,15	0,09
Vidrio sobre Madera	0,2	0,25
Caucho sobre Cemento	0,2	0,25
Madera sobre Cuero	0,5	0,4
Acero sobre Latón	0,5	0,4
Madera sobre Madera	0,7	0,4
Madera sobre Piedra	0,7	0,3
Vidrio sobre Vidrio	0,9	0,4

**Tabla. B.1.** Coeficientes de fricción ( $\mu_s$  estático,  $\mu_k$  dinámico)

**Fuente:** <http://elfisicoloco.blogspot.com/2014/05/calcular-el-coeficiente-de-rozamiento.html>

#### a. Bloque sobre una pendiente sin fricción

Para determinar la aceleración de un bloque colocado sobre un plano inclinado sin fricción y ángulo  $\theta$  se usará la segunda ley de Newton.



**Fig. B.1.** Bosquejo – Bloque sobre una pendiente sin fricción  
**Fuente:** Elaboración Propia mediante Herramienta Adobe Illustrator CS7

En el bosquejo se ve que las únicas fuerzas que actúan sobre el bloque son la normal ‘N’, Ejercida por el plano inclinado, actuando perpendicular al plano, la Fuerza de gravedad que actúa verticalmente hacia abajo.

Se obtiene la siguiente ecuación en la componente ‘x’:

$$\sum F_x = mg \text{ sen } \theta = ma_x$$

Se obtiene la siguiente ecuación en la componente ‘y’, tomando en cuenta que no existe aceleración en este eje  $a_y = 0$ .

$$\sum F_y = n - mg \cos \theta = 0$$

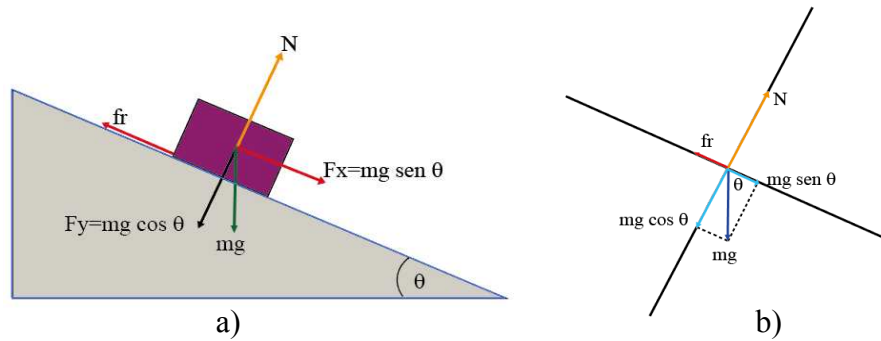
De la primera ecuación se tiene:

$$a_x = g \text{ sen } \theta$$

Lo que significa que la aceleración no depende de la masa, sólo depende del ángulo de inclinación y de la fuerza de gravedad.

## b. Coeficiente de fricción estática

Para encontrar el coeficiente de fricción estática en un plano inclinado es primero necesario hallar el ángulo crítico. El ángulo, en este caso, varía su inclinación hasta que el bloque sobre el plano inicie su deslizamiento.



**Fig. B.2.** Coeficiente de fricción estática-Bloque en un plano inclinado  
a) Bosquejo b) Diagrama de cuerpo libre

**Fuente:** Elaboración Propia mediante Herramienta Adobe Illustrator CS7

Se tienen las siguientes ecuaciones. Se toma en cuenta que no existe movimiento en ningún eje.

$$\sum F_x = mg \sin \theta - f_r = 0$$
$$\sum F_y = N - mg \cos \theta = 0 \quad \rightarrow \quad mg = \frac{N}{\cos \theta}$$
$$f_r = \mu_s N$$

Sustituyendo y despejando  $f_r$  :

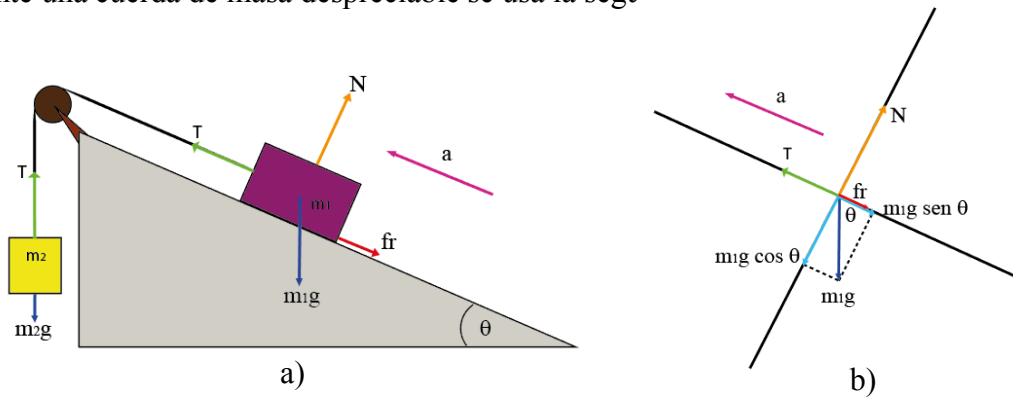
$$f_r = N \tan \theta$$

$$\mu_s = \tan \theta$$

Siendo  $\theta$  el ángulo crítico.

### c. Coeficiente de fricción cinética, bloques conectados con una cuerda

Para hallar el coeficiente de fricción cinética de un bloque conectado a un segundo bloque mediante una cuerda de masa despreciable se usa la segu



**Fig. B.3.** Coeficiente de fricción dinámica – Dos bloques conectados por una cuerda

b) Bosquejo b) Diagrama de cuerpo libre

**Fuente:** Elaboración Propia mediante Herramienta Adobe Illustrator CS7

Se tienen las siguientes ecuaciones:

$$1) \quad m_2 g - T = m_2 a$$

$$2) \quad \sum F_x = T - f_r - m_1 g \sen \theta = m_1 a_x \quad \rightarrow \quad a_x = a$$

$$3) \quad \sum F_y = N - m_1 g \cos \theta = m_1 a_y \quad \rightarrow \quad a_y = 0$$

$$4) \quad f_r = \mu_k N$$

Resolviendo las ecuaciones y despejando el coeficiente de fricción cinético  $\mu_k$  se tiene lo siguiente:

$$\mu_k = \frac{m_2(g - a) - m_1(a + g \sen \theta)}{m_1 g \cos \theta}$$

# ANEXO C

## Código Fuente IDE Arduino – Funciones

### C.1 Función serialA()

```
/**
 * Funcion Serial
 */
void serialA()
{
  while(Serial.available()==0){} // Espera comunicación por puerto Serial
  var=Serial.read(); // Guarda lo leído del puerto Serial
}
```

### C.2 Función reset(int j)

```
/**
 * FUNCIÓN RESET
 */
void reset(int j)
{
  motor.setSpeed(180); //Fija la velocidad del motor en RPM
  while(j==0)
  {
    if(digitalRead(9)==LOW) //Verifica el estado del final de carrera inferior
    {
      // Si el final de carrera inferior está activo entonces pone el contador
      // pasos a cero y cambia el valor de la condición para salir del bucle.
      cont=0;
      j=1;
      digitalWrite(11,HIGH) ; //Enciende el LED correspondiente a la posición
      inicial
    }
    else
      // En caso de no encontrarse activo el final decarrera inferior, el motor
      debe llegar
      // a la posición inicial.
      motor.step(-1); // Un paso en retraso al motor de pasos por vez
      digitalWrite(10,LOW); // Asegura que el led superior se apague (En caso de
      que la
      // plataforma se encuentre en el límite superior)
    }
    Serial.println("#Motor en estado RESET"); // Envía información por el
    puerto serial
    // El símbolo "#" al inicio
    servirá para la
    // interpretación en el Labview.
  }
}
```

### C.3 Función AnglePrincipal(float ang)

```
//*****  
//***** Angle Principal *****  
//*****  
// Permite introducir un valor de ángulo y mover el  
// motor de acuerdo al conteo de pasos.  
//*****  
  
void anglePrincipal(float ang)  
{  
    long x=0;  
  
    // Si el ángulo introducido es "cero", el programa  
    // irá al estado de "reset"  
    if(ang==0)  
        reset(0);  
  
    // Análisis por intervalos para usar los factores respectivos  
    else if(0<ang&&ang<=4.65) //Primer intervalo  
    {  
        x=10000/4.55*ang; // Fórmula 3.3.1  
        cont=angle(x,cont); // Sitúa el motor  
    }  
  
    else if((4.65<ang)&&(ang<=8.8)) // Segundo intervalo  
    {  
        x=10000+10000/4.15*(ang-4.65); // Fórmula 3.3.2  
        cont=angle(x,cont); // Sitúa el motor  
    }  
  
    else if((8.8<ang)&&(ang<=13.25)) // Tercer Intervalo  
    {  
        x=20000+(10000/4.45)*(ang-8.8); // Formula 3.3.3  
        cont=angle(x,cont); // Sitúa el motor  
    }  
  
    else if((13.25<ang)&&(ang<=17.25)) // Cuarto intervalo  
    {  
        x=30000+((10000/4)*(ang-13.25)); // Fórmula 3.3.4  
        cont=angle(x,cont); // Sitúa el motor  
    }  
  
    else if((17.25<ang)&&(ang<=21.4)) // Quinto Intervalo  
    {  
        x=40000+((10000/4.15)*(ang-17.25)); // Fórmula 3.3.5  
        cont=angle(x,cont); // Sitúa el motor  
    }  
  
    else if((21.4<ang)&&(ang<=25.3)) // Sexto Intervalo  
    {  
        x=50000+((10000/3.9)*(ang-21.4)); // Fórmula 3.3.6  
        cont=angle(x,cont); // Sitúa el motor  
    }  
  
    else if((25.3<ang)&&(ang<=28.7)) // Séptimo Intervalo  
    {  
        x=60000+((10000/3.4)*(ang-25.3)); // Fórmula 3.3.7  
        cont=angle(x,cont); // Sitúa el motor  
    }  
  
    else if((28.7<ang)&&(ang<=31.75)) // Octavo Intervalo
```



```

    {
        x=70000+((10000/3.05)*(ang-28.7)); // Fórmula 3.3.8
        cont=angle(x,cont); // Sitúa el motor
    }

    else if((31.75<ang)&&(ang<=34.5)) // Noveno Intervalo
    {
        x=80000+((10000/2.25)*(ang-31.75)); //Fórmula 3.3.9
        cont=angle(x,cont); //Sitúa el motor
    }
}

```

## C.4 Función sensores(int g)

```

//*****
//*****Sensores - Laboratorio 1*****
//*****
void sensores(int g)
{
    //*****
    //***** Definición de Variables *****
    //*****
    float Millis0; // Tiempo Actual del sensor 1
    float Millis1; // Tiempo Actual del sensor 2
    float Millis2; // Tiempo Actual del sensor 3
    float Millis3; // Tiempo Actual del sensor 4
    float Millis4; // Tiempo Actual del sensor 5
    float Millis5; // Tiempo Actual del sensor 6
    float Millis6; // Tiempo Actual del sensor 7
    float Millis7; // Tiempo Actual del sensor 8
    float Millis8; // Tiempo Actual del sensor 9

    float t0; // Tiempo en Segundos Sensor 1
    float t1; // Tiempo en Segundos Sensor 2
    float t2; // Tiempo en Segundos Sensor 3
    float t3; // Tiempo en Segundos Sensor 4
    float t4; // Tiempo en Segundos Sensor 5
    float t5; // Tiempo en Segundos Sensor 6
    float t6; // Tiempo en Segundos Sensor 7
    float t7; // Tiempo en Segundos Sensor 8
    float t8; // Tiempo en Segundos Sensor 9

    int lastSensorState0 = 1; // Último estado del sensor 1
    int lastSensorState1 = 1; // Último estado del sensor 2
    int lastSensorState2 = 1; // Último estado del sensor 3
    int lastSensorState3 = 1; // Último estado del sensor 4
    int lastSensorState4 = 1; // Último estado del sensor 5
    int lastSensorState5 = 1; // Último estado del sensor 6
    int lastSensorState6 = 1; // Último estado del sensor 7
    int lastSensorState7 = 1; // Último estado del sensor 8
    int lastSensorState8 = 1; // Último estado del sensor 9

    int SensorState0 = 0; // Estado actual del sensor 1
    int SensorState1 = 0; // Estado actual del sensor 2
    int SensorState2 = 0; // Estado actual del sensor 3
    int SensorState3 = 0; // Estado actual del sensor 4
    int SensorState4 = 0; // Estado actual del sensor 5
    int SensorState5 = 0; // Estado actual del sensor 6
    int SensorState6 = 0; // Estado actual del sensor 7
    int SensorState7 = 0; // Estado actual del sensor 8
    int SensorState8 = 0; // Estado actual del sensor 9
}

```

```

int h=1; // Condición inicial del bucle
while(h==1)
{
    //*****
    // Detetor de cambio de estado
    //*****
    // Se evitan lecturas dobles erradas
    SensorState0 = (PINA & (1<<PA0));
    // Lectura rápida, PIN 0, puerto A
    // Sensor 1

    if(SensorState0 != lastSensorState0)
    // Verificar si existe cambio de estado en la lectura
    {
        if(SensorState0 == LOW)
        //Verifica si el sensor está activado
        {
            Millis0=millis(); // Lectura del tiempo Actual/inicial
            t0=Millis0/1000; // Cálculo de tiempo en segundos
            Serial.println("#Inicio");
            // Información enviada por puerto serial
            delay(100); // Retardo de 100ms
        }
    }
    lastSensorState0=SensorState0; //Posición anterior actualizada

    SensorState1 = (PINA & (1<<PA1));
    // Lectura rápida, PIN 1, puerto A
    // Sensor 2
    if(SensorState1 != lastSensorState1)
    // Verificar si existe cambio de estado en la lectura
    {
        if(SensorState1 == 0) //Verifica Si el sensor está activado
        {
            Millis1=millis(); // Lectura del tiempo Actual
            t1=Millis1/1000-t0;
            // Cálculo de tiempo transcurrido en segundos
            delay(100); // Retardo de 100ms
        }
    }
    lastSensorState1=SensorState1; Posición anterior actualizada

    SensorState2 = (PINA & (1<<PA2));
    // Lectura rápida, PIN 2, puerto A
    // Sensor 3
    if(SensorState2 != lastSensorState2)
    // Verificar si existe cambio de estado en la lectura
    {
        if(SensorState2 == 0) //Verifica Si el sensor está activado
        {
            Millis2=millis(); // Lectura del tiempo Actual
            t2=Millis2/1000-t0;
            // Cálculo de tiempo transcurrido en segundos
            delay(100); // Retardo de 100ms
        }
    }
    lastSensorState2=SensorState2;// Posición anterior actualizada

    SensorState3 = (PINA & (1<<PA3));
    //Lectura rápida, PIN 3, puerto A
    // Sensor 4
    if(SensorState3 != lastSensorState3)

```

```

// Verificar si existe cambio de estado en la lectura
{
  if(SensorState3 == 0) //Verifica Si el sensor está activado
  {
    Millis3=millis();
    //Lectura del tiempo Actual
    t3=Millis3/1000-t0;
    //Cálculo de tiempo transcurrido en segundos
    delay(100); // Retardo de 100ms
  }
}
lastSensorState3=SensorState3;
// Posición anterior actualizada

SensorState4 = (PINA & (1<<PA4));
// Lectura rápida, PIN 4, puerto A
// Sensor 5
if(SensorState4 != lastSensorState4 )
// Verificar si existe cambio de estado en la lectura
{
  if(SensorState4 == 0) //Verifica Si el sensor está activado
  {
    Millis4=millis(); // Lectura del tiempo Actual
    t4=Millis4/1000-t0;
    // Cálculo de tiempo transcurrido en segundos
    delay(100); // Retardo de 100ms
  }
}
lastSensorState4=SensorState4;
// Posición anterior actualizada

SensorState5 = (PINA & (1<<PA5));
// Lectura rápida, PIN 5, puerto A
// Sensor 6
if( SensorState5 != lastSensorState5)
// Verificar si existe cambio de estado en la lectura
{
  if( SensorState5 == 0) //Verifica Si el sensor está activado
  {
    Millis5=millis(); // Lectura del tiempo Actual
    t5=Millis5/1000-t0;
    // Cálculo de tiempo transcurrido en segundos
    delay(100); // Retardo de 100ms
  }
}
lastSensorState5=SensorState5;
// Posición anterior actualizada

SensorState6 = (PINA & (1<<PA6));
// Lectura rápida, PIN 6, puerto A
// Sensor 7
if(SensorState6 != lastSensorState6)
// Verificar si existe cambio de estado en la lectura
{
  if(SensorState6 == 0)
  //Verifica Si el sensor está activado
  {
    Millis6=millis();// Lectura del tiempo Actual
    t6=Millis6/1000-t0;
    // Cálculo de tiempo transcurrido en segundos
    delay(100); // Retardo de 100ms
  }
}
}

```

```

lastSensorState6=SensorState6;
// Posición anterior actualizada

SensorState7 = (PINA & (1<<PA7));
// Lectura rápida, PIN 7, puerto A
// Sensor 8
if(SensorState7 != lastSensorState7)
// Verificar si existe cambio de estado en la lectura
{
  if(SensorState7 == 0) //Verifica Si el sensor está activado
  {
    Millis7=millis();// Lectura del tiempo Actual
    t7=Millis7/1000-t0;
    // Cálculo de tiempo transcurrido en segundos
    delay(100); // Retardo de 100ms
  }
}
lastSensorState7=SensorState7;
// Posición anterior actualizada

SensorState8 = (PINC & (1<<PC7));
// Lectura rápida, PIN 7, puerto C
// Sensor 9
if(SensorState8 != lastSensorState8)
// Verificar si existe cambio de estado en la lectura
{
  if(SensorState8 == 0) //Verifica Si el sensor está activado
  {
    Millis8=millis();// Lectura del tiempo Actual
    t8=Millis8/1000-t0;
    // Cálculo de tiempo transcurrido en segundos
    delay(100); // Retardo de 100ms
    h=0; // Cambio de condición para salir del bucle
  }
}
lastSensorState8=SensorState8;
// Posición anterior actualizada
}

// Enviar los datos obtenidos por el puerto serial con caracteres específicos
// para posterior lectura con LABVIEW
Serial.print('@'+String(t1)+'%'+String(t2)+'*'+String(t3)+'!'+String(t4)+'-
'+String(t5)+' ('+String(t6)+')' '+String(t7)+'_'+String(t8));
}

```

## C.5 Función ValAngulo(int a)

```

//*****
//*****Función Valor de Ángulo *****
//*****
// Obtiene el valor del ángulo en grados de acuerdo al número de pasos
// avanzados por el motor.
//*****
void ValAngulo(int a)
{
  float y=0; // Ángulo calculado

  while(a==1)
  // Condición de bucle, se cumple hasta que cambie la condición
  {

```

```

if(digitalRead(8)==HIGH)
// Verifica que el final de carrera superior no esté activado
{
    if(digitalRead(24) == 1)
    // Consulta si el conmutador sin contacto se encuentra activo
    {
        motor.step(1);
        // Mientras el bloque no bloquee la línea de vista del
        // conmutador del contacto, el motor avanza un paso.
        cont++; // Actualiza el valor de los pasos avanzados
    }
    else
    //Si el sensor está activado el bloque inició el deslizamiento.
    {
        motor.step(0); // Detener el motor de pasos
        a=0; // Cambia la condición para salir del bucle.
    }
}
else // Final de carrera superior activado
{
    motor.step(0); // Detener el motor por seguridad
    digitalWrite(10,HIGH);
    //Encender el led correspondiente a la posición final
    a=0; // Cambiar condición para salir del bucle
}
}
//*****
// Este sector se realiza el cálculo de ángulo
//*****

if((0<cont)&&(cont<=10000)) //Intervalo 0 a 10000 pasos
{
    y=(4.55/10000)*cont; // Fórmula 3.3.10
    Serial.println('&'+String(y));
    //Dato enviado por puerto serial
}
else if((10000<cont)&&(cont<=20000)) //Intervalo 10000 a 20000 pasos
{
    y=((10000/4.15)*4.65)-10000+cont)*4.15/10000; // Fórmula 3.3.11
    Serial.println('&'+String(y)); // Dato enviado por puerto serial
}
else if((20000<cont)&&(cont<=30000)) //Intervalo 20000 a 30000 pasos
{
    y=((10000/4.45)*8.8)-20000+cont)*4.45/10000; // Fórmula 3.3.12
    Serial.println('&'+String(y)); // Dato enviado por puerto serial
}
else if((30000<cont)&&(cont<=40000)) //Intervalo 30000 a 40000 pasos
{
    y=((10000/4)*13.25)-30000+cont)*4/10000; // Fórmula 3.3.13
    Serial.println('&'+String(y)); // Dato enviado por puerto serial
}
else if((40000<cont)&&(cont<=50000)) //Intervalo 40000 a 50000 pasos
{
    y=((10000/4.15)*17.25)-40000+cont)*4.15/10000; // Fórmula 3.3.14
    Serial.println('&'+String(y)); // Dato enviado por puerto serial
}
else if((50000<cont)&&(cont<=60000)) //Intervalo 50000 a 60000 pasos
{
    y=((10000/3.9)*21.4)-50000+cont)*3.9/10000; // Fórmula 3.3.15
    Serial.println('&'+String(y)); // Dato enviado por puerto serial
}
else if((60000<cont)&&(cont<=70000)) // Intervalo 60000 a 70000 pasos
{
    y=((10000/3.4)*25.3)-60000+cont)*3.4/10000; // Fórmula 3.3.16
    Serial.println('&'+String(y)); // Dato enviado por puerto serial
}

```

```

    }
    else if((70000<cont)&&(cont<=80000)) // Intervalo 70000 a 80000 pasos
    {
        y=((10000/3.05)*28.7)-70000+cont)*3.05/10000; // Fórmula 3.3.17
        Serial.println('&'+String(y)); // Dato enviado por puerto serial
    }
    else if((80000<cont)&&(cont<=86000)) // Intervalo 80000 a 86000 pasos
    {
        y=((10000/2.25)*31.75)-80000+cont)*2.25/10000; // Fórmula 3.3.18
        Serial.println('&'+String(y)); // Dato enviado por puerto serial
    }
}

```

## D.6 Función Lab3(int h)

```

//*****
//*****Laboratorio 3*****
//*****
void Lab3(int h)
{
    //*****
    //Definición de variables
    //*****
    float Millis0; // Lectura de tiempo actual sensor 1
    float Millis1; // Lectura de tiempo actual sensor 2
    float Millis2; // Lectura de tiempo actual sensor 3
    float Millis3; // Lectura de tiempo actual sensor 4
    float Millis4; // Lectura de tiempo actual sensor 5
    float t0; // Tiempo en Segundos Sensor 1
    float t1; // Tiempo en Segundos Sensor 2
    float t2; // Tiempo en Segundos Sensor 3
    float t3; // Tiempo en Segundos Sensor 4
    float t4; // Tiempo en Segundos Sensor 5
    int lastSensorState0 = 1; // Último estado del Sensor 1
    int lastSensorState1 = 1; // Último estado del Sensor 2
    int lastSensorState2 = 1; // Último estado del Sensor 3
    int lastSensorState3 = 1; // Último estado del Sensor 4
    int lastSensorState4 = 1; // Último estado del Sensor 5
    int SensorState0 = 0; //Estado del Sensor 1
    int SensorState1 = 0; //Estado del Sensor 2
    int SensorState2 = 0; //Estado del Sensor 3
    int SensorState3 = 0; //Estado del Sensor 4
    int SensorState4 = 0; //Estado del Sensor 5

    while(h==1)
    {
        SensorState0 = (PINA & (1<<PA4)); // Lectura rápida, PIN 4, puerto A
        // Sensor 1
        if(SensorState0 != lastSensorState0)
            // Verificar si existe cambio de estado en la lectura
            {
                if(SensorState0 == LOW) //Verifica Si el sensor está activado
                {
                    Millis0=millis(); // Lectura del tiempo Actual
                    t0=Millis0/1000; // Cálculo de tiempo inicial en segundos
                    Serial.println("#Inicio"); // Información enviada por puerto serial
                    delay(100); //Retardo de 100ms
                }
            }
        lastSensorState0=SensorState0; // Posición anterior actualizada
    }
}

```

```

SensorState1 = (PINA & (1<<PA3)); // Lectura rápida, PIN 3, puerto A
// Sensor 2
if(SensorState1 != lastSensorState1)
// Verificar si existe cambio de estado en la lectura
{
    if(SensorState1 == 0) //Verifica Si el sensor está activado
    {
        Millis1=millis(); // Lectura del tiempo Actual
        t1=Millis1/1000-t0; // Cálculo de tiempo transcurrido en segundos
        delay(100); //Retardo de 100ms
    }
}
lastSensorState1=SensorState1; // Posición anterior actualizada

SensorState2 = (PINA & (1<<PA2)); // Lectura rápida, PIN 2, puerto A
// Sensor 3
if(SensorState2 != lastSensorState2)
// Verificar si existe cambio de estado en la lectura
{
    if(SensorState2 == 0) //Verifica Si el sensor está activado
    {
        Millis2=millis(); // Lectura del tiempo Actual
        t2=Millis2/1000-t0; // Cálculo de tiempo transcurrido en segundos
        delay(100); //Retardo de 100ms
    }
}
lastSensorState2=SensorState2; // Posición anterior actualizada

SensorState3 = (PINA & (1<<PA1)); // Lectura rápida, PIN 1, puerto A
// Sensor 4
if(SensorState3 != lastSensorState3)
// Verificar si existe cambio de estado en la lectura
{
    if(SensorState3 == 0) //Verifica Si el sensor está activado
    {
        Millis3=millis(); // Lectura del tiempo Actual
        t3=Millis3/1000-t0; // Cálculo de tiempo transcurrido en segundos
        delay(100); //Retardo de 100ms
    }
}
lastSensorState3=SensorState3; // Posición anterior actualizada

SensorState4 = (PINA & (1<<PA0)); // Lectura rápida, PIN 0, puerto A
// Sensor 5
if(SensorState4 != lastSensorState4 )
// Verificar si existe cambio de estado en la lectura
{
    if(SensorState4 == 0) //Verifica Si el sensor está activado
    {
        Millis4=millis(); // Lectura del tiempo Actual
        t4=Millis4/1000-t0; // Cálculo de tiempo transcurrido en segundos
        delay(100); //Retardo de 100ms
        h=0; // Cambio de condición para salida del bucle
    }
}
lastSensorState4=SensorState4; // Posición anterior actualizada
}
// Enviar los datos obtenidos por puerto serial con caracteres específicos para
// lectura con LABVIEW
Serial.print('@'+String(t1)+'%'+String(t2)+'*'+String(t3)+'!'+String(t4));
}

```

# ANEXO E

## Código Fuente IDE Arduino – Principal

```
//Programa Base
// Elección de Laboratorio
//
// 1.Aceleración sin Fricción
// 2.Determinación experimental de us(eficiente de fricción estático)
// 3.Determinación experimental de ud(eficiente de fricción dinámico)
//*****
// Lectura - Serial (Elección de Laboratorio)
// A - Lab 1
// B - Lab 2
// C - Lab 3
//*****
//Lectura - Serial (Laboratorio 1)
// R - Guardar valor de Ángulo
// O - Posicionar Motor
// S - Inicializar Labo
// T - Toma de datos
// M - End Lab
//*****
//Lectura - Serial (Laboratorio 2)
// N - START PROGRAM
// R - Reset
// X - End Lab
//*****
//Lectura - Serial (Laboratorio 3)
// R - Guardar valor de Ángulo
// O - Posicionar Motor
// S - Inicializar Labo
// T - Toma de datos
// X - End Lab
//*****

// FUNCION MOTOR PASO A PASO

#include<Stepper.h>
int steps=200; //pasos por giro completo
Stepper motor(steps,2,3,4,5);

//*****
char var;
long cont;
int b=1;
float ang;

//*****

//*****
void setup()
{
  DDRA=B00000000;//Puerto A como entradas PA0 a PA7(Lab 1 & 3)
  DDRC=B01111111;
  //Puerto C Pin PC7 como entrada y pines PC0 al PC6 Como salidas(No serán
  //usadas)(Lab 1 & 3)
  pinMode(8,INPUT); // Final de carrera Superior
  pinMode(9,INPUT); // Final de carrera Inferior
  pinMode(10,OUTPUT);// Led de señalización para final de carrera superior
  pinMode(11,OUTPUT);// Led de señalización para final de carrera inferior
  Serial.begin(9600); // Velocidad de transmisión serial
}
```



```

}

void loop()
{
  serialA(); // Datos puerto Serial
  switch(var) // Elección de Laboratorio
  {
    case 'A': // Laboratorio 1
    {
      Serial.println("#Laboratorio 1"); //Escritura Puerto Serial
      reset(0); // Motor en posición inicial
      Serial.println("#READY"); //Escritura puerto Serial
      int g=1; //Condición para bucle
      while(g==1) // Bucle condicionado
      {
        serialA(); // Lectura del puerto serial
        switch(var) // Escoge acción del laboratorio
        {
          case 'R': // Guardar valor de Ángulo
          {
            Serial.println("#Guardando Dato...");
            //Escritura Puerto Serial
            while(Serial.available()==0){}
            //Espera cambio en Puerto serial
            ang=Serial.parseFloat();
            // Guarda Lectura en variable como flotante
            serialA(); //Lectura de puerto serial
            int p=1; // Inicializar condición para Bucle
            while(p==1) //Bucle condicionado
            {
              switch(var)
              {
                case 'O': // Valor de ángulo leído
                {
                  if(ang<=5) // Verificar si es menor de 5°
                  {
                    anglePrincipal(ang);
                    //Ejecución de Función con el ángulo
                    //introducido
                    p=0;
                    // Cambio de condición para salir del bucle
                  }
                  else // Ángulo mayor al límite determinado
                  {
                    Serial.println("#ERROR-Introducir angulo menor igual a 5 grados");
                    p=0;
                    // Cambio de condición para salir del bucle
                  }
                  break; // Fin del caso
                }
              }
            }
          }
          break; // Fin del caso para guardar datos
        }
      }
    }
    case 'S':
    {
      Serial.println("#Inicializando...");
      //Escritura, puerto serial
      int p=1; // Inicializar condición de bucle
      while(p==1) // Bucle condicionado
      {
        serialA(); //Lectura, puerto serial
        switch(var)
        {
          case 'T':
          {
            sensores(1);
            //Ejecución de función sensores
            //Obtiene tiempos de activación de cada sensor
          }
        }
      }
    }
  }
}

```





# ANEXO F

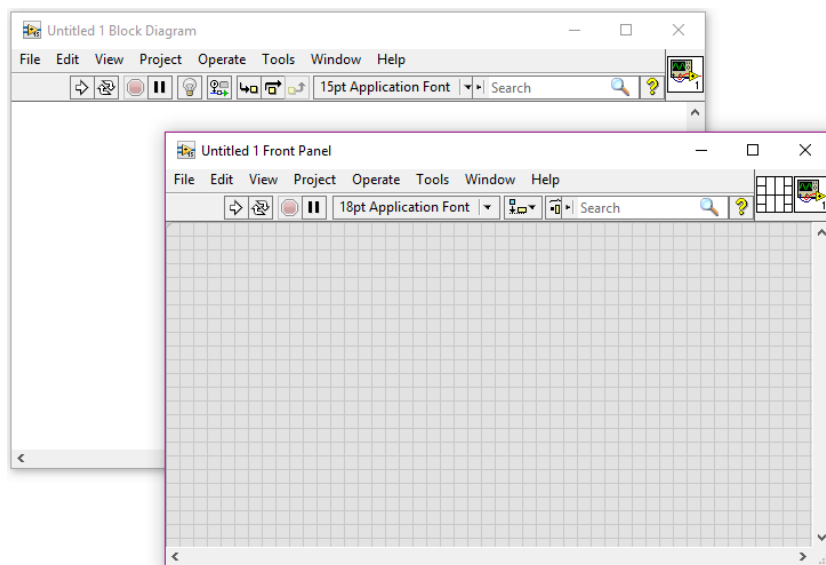
## Entorno de Programación LabVIEW

### F.1 Panel Frontal

La parte que verá y con la que interactuará el usuario. Donde se mostrarán los botones, gráficos, perillas, etc.

### F.2 Diagrama de Bloque

Donde se realizará la programación en base a bloques, funciones e incluso SubVIs todos conectados para crear un programa gráfico.



**Fig.F.1.** Panel Frontal / Diagrama de Bloques

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

El panel frontal y el diagrama de bloques están conectados a través de los terminales (Elementos que sirven como entradas o salidas de datos).

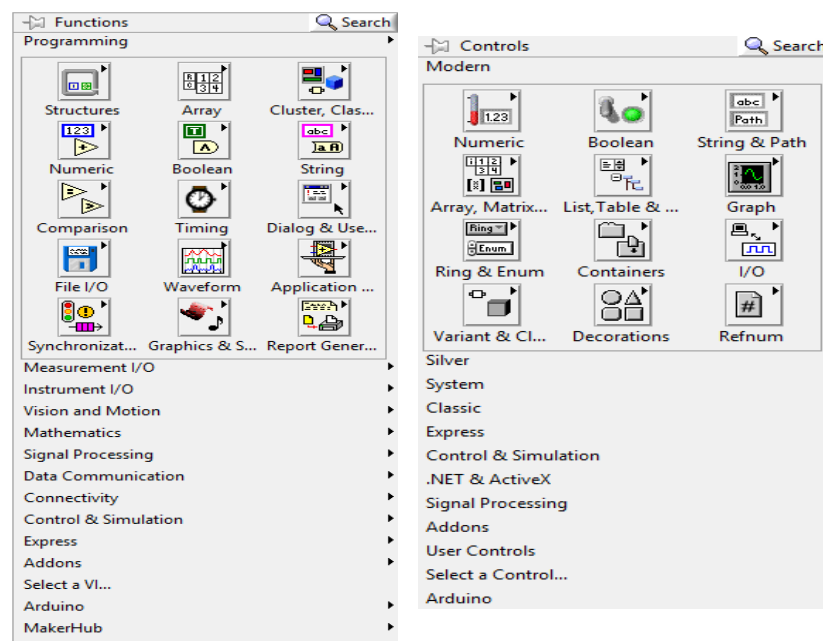
Un VI puede ser creado de manera sencilla y rápida además de que puede modificarse las veces que sean necesarias.

El lenguaje de programación que usa LabView también se llama lenguaje G que basa su ejecución en el flujo de datos (dataflow).

Un programa en Labview consiste en una serie de bloques o funciones unidas mediante cables. Cada bloque o función sólo podrá ejecutarse cuando tenga todos los datos disponibles en sus entradas conectados con los cables por donde fluyen los datos.

Tanto el panel frontal como el diagrama de bloques tienen su menú de ejecución. El menú que aparece en el panel frontal es el de Controles, donde se pueden seleccionar los terminales que servirán para interactuar con el usuario ya sean controles o indicadores. El menú que aparece en el diagrama de bloques es el de Funciones, en esta paleta se puede acceder a las diferentes funciones, subVIs y estructuras disponibles. En ambos menús se pueden apreciar submenús que se dividen dependiendo a la aplicación.

La programación en LabView se trata de la selección de objetos tanto en la paleta presente en el Panel Frontal (Controles) como en la paleta presente en el Diagrama de Bloques (Funciones) e interactuar con ambas mediante el conexionado de los objetos. El método de para usar los objetos de las paletas Funciones y Controles es “arrastrar” y “colocar”.



**Fig. F.2.** Paleta de Funciones y Paleta de Controles

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

### F.3 Estructuras.

Es necesario conocer los principales componentes que tiene LabView para su programación. Las instrucciones de control permiten a un programa ejecutar un código de programa de forma condicional o repetitivo. Estas instrucciones son estructuras que encierran en su interior el código al que afectan.

- **Estructura While**

El bucle While es una estructura de programación que causa que el código que se encuentra dentro de dicha estructura sea evaluado repetidas veces hasta que se cumpla una condición, la cual es evaluada en cada iteración.



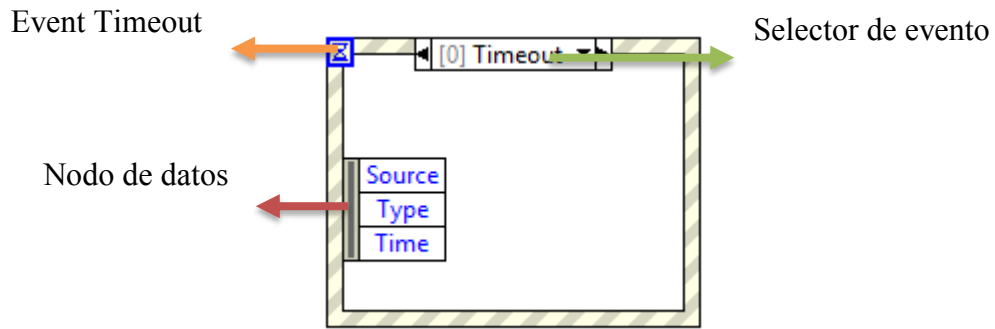
**Fig. F.3.** Estructura WHILE

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

El valor de la terminal de iteración es un número entero que irá aumentando en una unidad por cada iteración del bucle iniciando en cero, a su vez, se podrá conectar bien un valor booleano o un botón de parada a la condición de STOP para que termine el ciclo del While. Una opción importante que tiene el bucle While es el “Shift Register”. Esta herramienta añade los terminales a cada lado de la estructura, sirven para transferir el valor de una iteración de bucle a la siguiente. Los valores se pasarán a la siguiente iteración en el terminal de la derecha y se leerán en el de la izquierda.

- **Estructura EVENT**

La estructura Event tiene varios subdiagramas y un menú en la parte superior para cambiar el que se muestra. En este menú se tiene una condición que hace que el código del subdiagrama correspondiente se ejecute. Event congela el programa hasta que ocurre el evento. Es generalmente usado para manejar eventos de usuarios como el clic de un botón.



**Fig. F.4.** Estructura EVENT

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

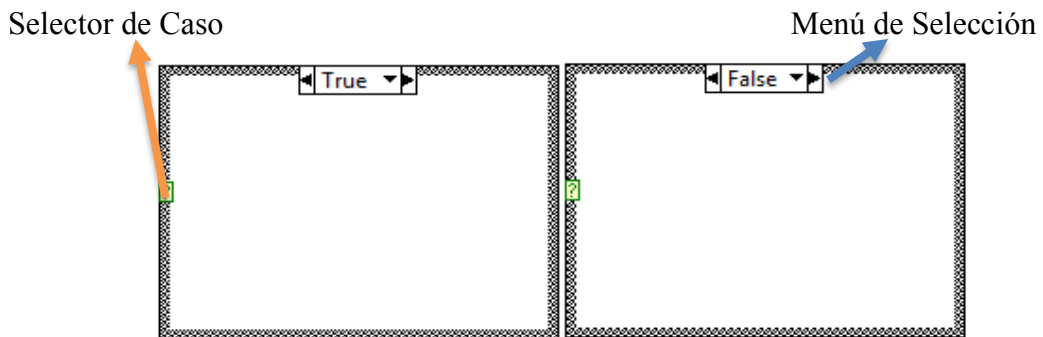
El terminal Timeout especifica la cantidad de milisegundos que debe esperar un evento antes de que se agote el tiempo de espera.

La etiqueta del selector de eventos especifica qué eventos causan que se ejecute el código que se muestra.

El Nodo de datos de eventos identifica los datos que LabVIEW devuelve cuando ocurre un evento.

- **Estructura CASE**

La utilidad de la estructura Case es ejecutar un código u otro dependiendo de una condición.



**Fig.F.5.** Estructura CASE

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

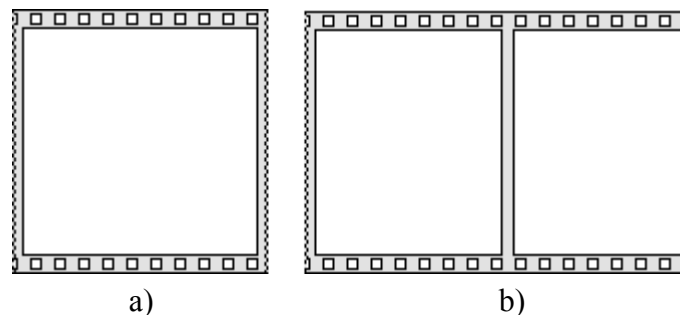
En el menú de selección se puede ver la condición para ejecutar el código del subdiagrama correspondiente. En el caso de la figura Fig. tenemos sólo dos subdiagramas, True o False.

El valor que llega al selector de caso es la condición que se evalúa para seleccionar el subdiagrama a ejecutar.

• **Estructura FLAT SEQUENCE**

Consiste en uno o más subdiagramas, o marcos, que se ejecutan secuencialmente. Se usa la estructura Flat Sequence para asegurarse de que un subdiagrama se ejecute antes o después de otro subdiagrama.

Los Frames se ven uno a continuación del siguiente, el orden de ejecución es de izquierda a derecha.



**Fig.F.6.** Estructura FLAT SEQUENCE  
a)Estructura de un Frame, b)Estructura de dos Frames

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

**F.4 Tipos de Datos**

El tipo de dato hace referencia a si son numéricos, cadenas de caracteres, etc. En diagrama de bloques, se representan por el color del terminal y del cable.

<i>Símbolo</i>	<i>Tipo de dato</i>	<i>Color de Cable</i>	<i>Rango</i>	<i>Comentario</i>
DBL	Precisión doble punto flotante numérico	Naranja	$\pm 1.7e+308$	Tipo de datos predeterminado para valores numéricos tipo Flotante



SGL	Precisión Extendida punto flotante	Naranja	$\pm 3.4e+38$	
I32	32 bits signado número entero	Azul	-2,147,483,648 a 2,147,483,647	Tipo de datos predeterminado para valores numéricos enteros
I16	Entero - Word	Azul	-32768 a +3267	
I8	Entero – Byte	Azul	-128 a +127	
TF	Booleano	Verde	Falso ó Verdadero	
[DBL]	Valor numérico, precisión doble Matricial, Array.			Los corchetes indican array o matriz. El color indica tipo de datos o elementos de la matriz. Los Alambres se muestran en las matrices portadoras con líneas gruesas
abc	String o cadena de caracteres, o caracteres	Magenta		
	Forma de Onda	Café		Sostiene la hora de inicio, el paso del tiempo, y datos de una forma de onda.

**Tabla.F.1.** Tipos de datos más comúnmente utilizados en LabVIEW

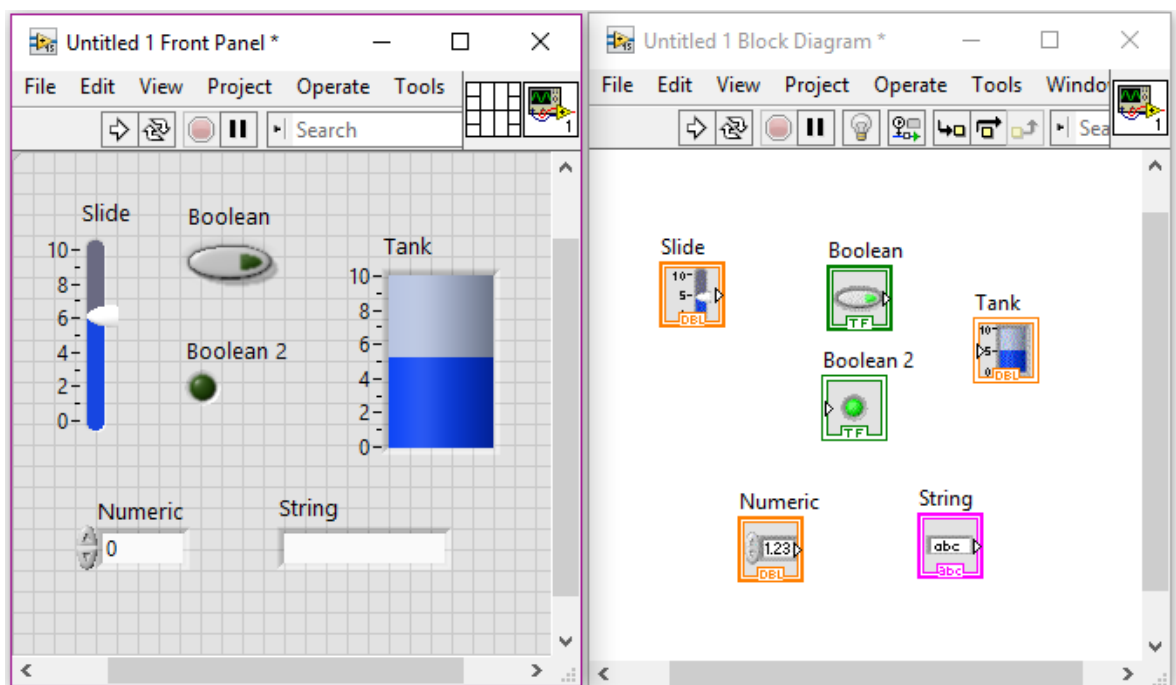
**Fuente:** Holguín , Germán Andrés, 2002 : “Curso LabView 6i”

W. Larsen, Ronald, 2011: “LabView for Engineers”

## F.5 Tipos de Terminales

Todos los controles e indicadores que sean ubicados en el panel frontal de Labview generarán un en el diagrama de bloques.

Los terminales son objetos del diagrama de bloques que representan un control o un indicador en el panel frontal. Toman el color respectivo de la variable que manejan. Por medio de ellos se obtienen los datos de los controles y se envían datos a los indicadores. Un terminal de control se diferencia de uno indicador en que los primeros poseen un borde doble mientras que los segundos uno sencillo.



**Fig.F.7.** Terminales LabView

**Fuente:**Elaboración propia mediante herramienta National Instrument LabView V 15.0

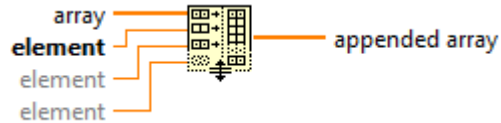
## F.6 Trabajando con Datos

- *Funciones Arrays*

Las matrices son muy potentes para usar en LabVIEW. En todas las aplicaciones probablemente se usarían matrices unidimensionales y matrices bidimensionales. LabVIEW tiene muchas funciones integradas para manipular matrices.

- **Construcción de Array**

Concatena matrices múltiples o agrega elementos a una matriz n-dimensional.

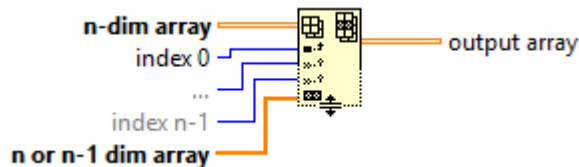


**Fig.F.8.** Build Array

**Fuente:**Elaboración propia mediante herramienta National Instrument LabView V 15.0

- **Insertar a Array**

Inserta un elemento o subcampo en una matriz de dimensión ‘n’ en el punto que especifique en el índice. Cuando conecta una matriz a esta función, la función cambia de tamaño automáticamente para mostrar las entradas de índice para cada dimensión en la matriz.



**Fig. F.9.** Insertar a Array

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

- **Transponer Array**

Reorganiza los elementos de la matriz de modo que la matriz 2D [i, j] se transpone a la matriz [j, i].

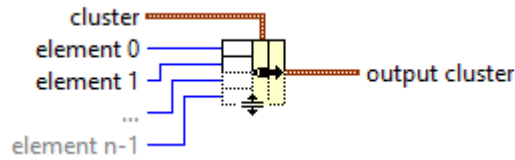


**Fig.F.10.** Función Transpuesta

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

- ***Empaquetado***

Ensambla un clúster de elementos individuales. Un clúster es una agrupación de tipos de datos potencialmente diferentes. Los clústeres se utilizan para recopilar datos relacionados que requieren una variedad de tipos de datos.

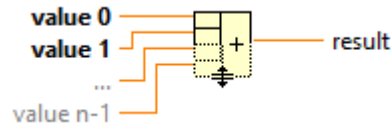


**Fig.F.11.** Función Empaquetado

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

- ***Aritmética Compuesta***

Realiza operaciones aritméticas en una o más entradas numéricas, de matriz, clúster o booleanas.

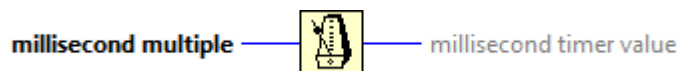


**Fig.F.12.** Función Aritmética Compuesta

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

- ***Función Esperar al siguiente Mili Segundo***

Espera hasta que el valor del temporizador de milisegundos se convierte en un múltiplo del múltiplo de milisegundos especificado. Esta función se usa para sincronizar actividades. Se puede llamar a esta función en un bucle para controlar la tasa de ejecución del bucle.



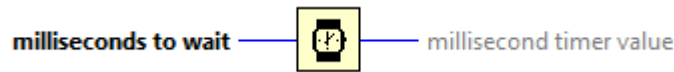
**Fig. F.13.** Función Esperar Siguiente ms

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

- **Esperar ms**

Espera el número especificado de milisegundos y devuelve el valor del temporizador de milisegundos.

Esta función realiza llamadas asíncronas al sistema, pero los nodos funcionan de forma síncrona. Por lo tanto, no completa la ejecución hasta que haya transcurrido el tiempo especificado.

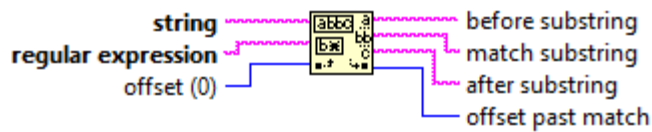


**Fig. F.14.** Función Esperar ms

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

- **Patrón de Emparejamiento**

Busca una expresión regular en una cadena de caracteres que comienza en el offset. Si la función encuentra una coincidencia, divide la cadena en tres subcadenas. “Antes de la Subcadena” , “Subcadena Emparejada” y “Después de la Subcadena”

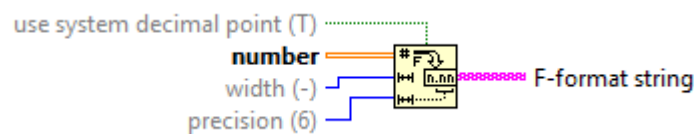


**Fig.F.15.** Función de Emparejamiento

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

- **Función de Número a Cadena Fraccional**

Convierte el número de formato en notación fraccional a una cadena de punto flotante. El número que ingresa puede ser un número escalar, una matriz, un grupo de números, etc.

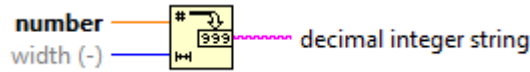


**Fig. F.16.** Número a Cadena Fraccional

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

• ***Función de Número a Cadena Decimal***

Convierte el número en una cadena de dígitos decimales de ancho mínimo o mayor, si es necesario.

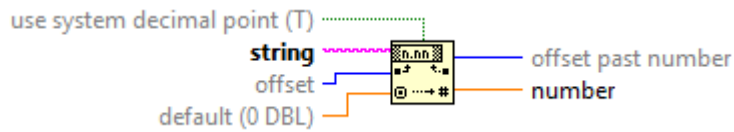


**Fig. F.17.** Número a cadena Decimal

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

• ***Función de Cadena Fraccional a Número***

Interpreta los caracteres del 0 al 9 y el punto decimal en la secuencia que comienza en el desplazamiento como un número de punto flotante en notación de ingeniería, exponencial o formato fraccionario y lo devuelve en número.



**Fig. F.18.** Cadena Fraccional a Número

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

• ***Nodo de Expresión – Grados a Radianes***

Use el nodo de expresión para calcular expresiones que contienen una sola variable. En este caso, la función se encarga de hacer la transformación de Grados a Radianes.



**Fig. F.19.** Nodo de Expresión – Grados a Radianes

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

• ***Funciones Seno y Coseno***

Realiza el cálculo del Seno y Coseno, respectivamente del ángulo “x”, donde “x” debe ser un valor en radianes.

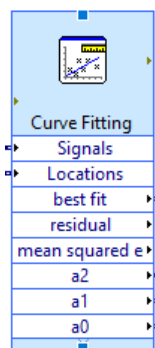


**Fig. F.20.** Funciones ‘Sen’ y ‘Cos’

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

• **Función Ajuste de Curvas**

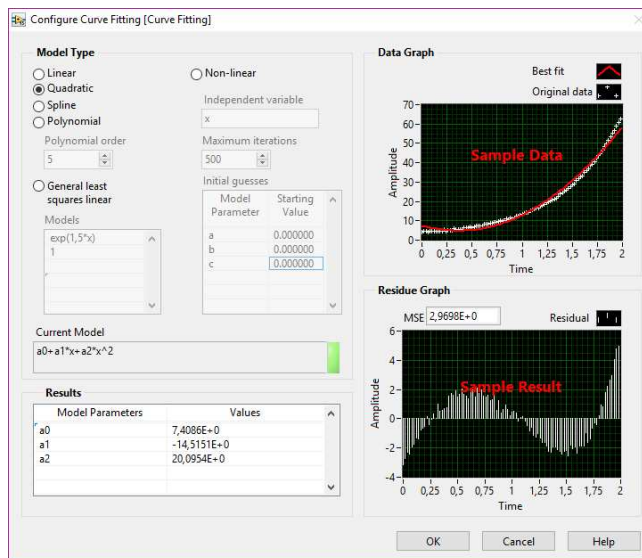
Calcula los coeficientes que mejor representan los datos de entrada en función del tipo de modelo elegido.



**Fig. F.21.** Función Ajuste de Curvas

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

La configuración del tipo de Ajuste requerido dependerá de los ajustes determinados en el menú propio de la Función.



**Fig. F.22.** Menú Propio de la Función Ajuste de Curvas

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

## F.7 VISA

### • Comunicación Serial

Los sistemas de comunicación serial se caracterizan por tener una transmisión a mayor distancia, menor coste, y más sencillo en cuanto al hardware necesario. Suelen ser comunicaciones punto a punto. Generalmente son transmisiones asíncronas y en ráfagas, por lo que suelen delimitar los datos, para esto pueden indicar el principio con un bit de ‘start’ y el final con un bit de ‘stop’, además que pueden incluir bits de comprobación de errores.

Una trama son todos los bits transmitidos, desde el de ‘start’ al de ‘stop’ cuyas características son: La velocidad (medida en Baudios), el número de bits de datos, el tipo de paridad.

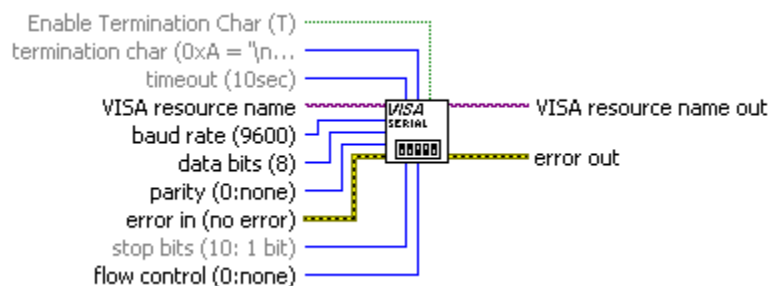
En LabView, la comunicación serial se realiza mediante la librería VISA.

VISA- Virtual Instrument Software Architecture (Arquitectura de software de Instrumento Virtual) es un estándar E/S Application Programming Interfase (API) para la programación de instrumentación. VISA por sí mismo, no proporciona capacidad para programar instrumentos. VISA es un API de alto nivel, llamada desde un driver de bajo nivel.

VISA puede controlar instrumentos seriales, haciendo uso de drivers apropiados dependiendo el tipo de instrumento que se usa. VISA usa las mismas operaciones para comunicarse con instrumentos sin tener en cuenta el tipo de interfase.

### ▪ VISA Configure Serial Port

Configura el puerto Serie con los parámetros de Velocidad, protocolo, paridad etc.



**Fig.F.23.** Configuración de puerto Serial VISA

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0



El puerto se selecciona con VISA resource name, al crear un control o una constante en el Diagrama de Bloques o en el panel frontal se escanean los puertos disponibles en el ordenador.

### ▪ VISA Write

Escribe datos desde el buffer hasta otro dispositivo. Se puede cambiar el tipo de comunicación para que este VI y el siguiente usen una transmisión síncrona o asíncrona.

Cuando transfiere datos desde o hacia un controlador de hardware de forma síncrona, el hilo de llamada se bloquea durante la transferencia de datos. Dependiendo de la velocidad de la transferencia, esto puede dificultar otros procesos que requieren el hilo de llamada. Sin embargo, si una aplicación requiere que la transferencia de datos sea lo más rápida posible, la realización de la operación de forma síncrona dedica la secuencia de llamada exclusivamente a esta operación. En la mayoría de las aplicaciones, las llamadas síncronas son un poco más rápidas cuando se comunica con cuatro o menos instrumentos. Las operaciones asíncronas resultan en una aplicación significativamente más rápida cuando se está comunicando con cinco o más instrumentos. El valor predeterminado de LabVIEW es la E / S asíncrona.



**Fig. F.24.** Escritura en VISA

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

### ▪ VISA Read

Lee el número especificado de bytes desde el dispositivo o la interfaz especificada por el nombre del recurso VISA y devuelve los datos en el búfer de lectura.

El VISA Read debe recibir el número de bytes igual al número máximo de bytes que deben leerse desde el instrumento. El VI terminará la lectura cuando este número de bytes se haya leído o cuando se indica el final de la transferencia.



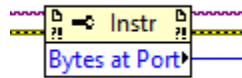
**Fig. F.25.** Lectura en VISA

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

▪ **VISA Bytes at Serial Port**

Obtiene los bytes que hay en el buffer del puerto esperando para ser leídos.

Generalmente se usa junto a la función de lectura de VISA ya que el número de bytes que recibe en la lectura puede variar.



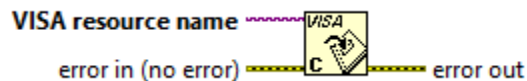
**Fig. F.26.** Bytes en el Puerto

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

▪ **VISA Close**

Cierra una sesión VISA especificado por el nombre de recurso.

La señal de error funciona de forma única en esta función. La función cierra la sesión del dispositivo independientemente de si se produjo un error en una operación anterior.



**Fig. F.27.** Cerrar Sesión VISA

**Fuente:** Elaboración propia mediante herramienta National Instrument LabView V 15.0

Mediante NI-VISA se puede acceder de varios fabricantes. Como visa soporta varios interfaces de comunicación suele ser el método elegido para crear drivers de control de equipos.

## **ANEXO G**

### **Hojas de Datos**