UNIVERSIDAD ISRAEL

TRABAJO DE TITULACIÓN

CARRERA: ELECTRÓNICA DIGITAL Y TELECOMUNICACIONES

TEMA: DISEÑO DE UN BRAZO ROBÓTICO AUTÓMATA CONTROLADO POR UN SISTEMA HUMAN MACHINE INTERFACE QUE TRANSPORTE CUBOS DE MADERA ENTRE DOS PUNTOS FIJOS PARA LOS LABORATORIOS DE ELECTRÓNICA DE LA UNIVERSIDAD ISRAEL.

AUTOR: LUIS ARTURO MOSQUERA GARCÉS

TUTOR: ING. WILMER ALBARRACÍN MG.

AÑO 2015

UNIVERSIDAD TECNOLÓGICA ISRAEL

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de titulación certifico:

Que el trabajo de titulación **"DISEÑO DE UN BRAZO ROBÓTICO AUTÓMATA CONTROLADO POR UN SISTEMA HUMAN MACHINE INTERFACE QUE TRANSPORTE CUBOS DE MADERA ENTRE DOS PUNTOS FIJOS PARA LOS LABORATORIOS DE ELECTRÓNICA DE LA UNIVERSIDAD ISRAEL.",** presentado por el Sr. Luis Arturo Mosquera Garcés estudiante de la carrera de Electrónica y Telecomunicaciones, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se designe, para su correspondiente estudio y calificación.

Quito D.M. Abril del 2015

TUTOR

…….

Mg. WILMER ALBARRACÍN

UNIVERSIDAD TECNOLÓGICA ISRAEL

AUTORÍA DEL TRABAJO DE TITULACIÓN

El abajo firmante, en calidad de estudiante de la Carrera de Electrónica y Telecomunicaciones, declaro que los contenidos de este Trabajo de Titulación, requisito previo a la obtención del Grado de Ingeniería en Electrónica y Telecomunicaciones, son absolutamente originales, auténticos y de exclusiva responsabilidad legal y académica del autor.

Quito D.M. Abril del 2015

…….
LUIS ARTURO MOSQUERA GARCÉS
CC: 1724531999

UNIVERSIDAD TECNOLÓGICA ISRAEL

APROBACIÓN DEL TRIBUNAL DE GRADO

Los miembros del Tribunal de grado, aprueban el trabajo de titulación para la graduación de acuerdo con las disposiciones reglamentarias emitidas por la Universidad Tecnológica Israel para títulos de pregrado.

Quito D.M. Abril del 2015

Para constancia firma:

TRIBUNAL DE GRADO

……
PRESIDENTE

…… ……
MIEMBRO 1 MIEMBRO 2

AGRADECIMIENTO

En primer lugar agradezco a DIOS por darme la vida y guiarme en cada paso que doy, dándome sabiduría e inteligencia para cumplir mis metas y objetivos trazados. A mis padres que han sido mi soporte y me han apoyado siempre, dándome el principal ejemplo de honestidad, trabajo, dedicación y esfuerzo. A mis hermanos que me han apoyado en los momentos difíciles de mi trayectoria educativa. A mis profesores que con su enseñanza y sus consejos brindados, han permitido llenarme de conocimientos.

DEDICATORIA

Acto que dedico a DIOS por guiarme en toda la trayectoria de mi carrera y darme fortaleza para terminar exitosamente otra meta en la etapa de mi vida. A mis padres por ser los cimientos en la construcción de mi vida profesional, sentando bases de responsabilidad y deseos de superación, en ellos tengo el espejo en el cual me quiero reflejar. A mis hermanos por su apoyo y confianza en todo lo necesario para cumplir mis objetivos como persona y estudiante.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Una máquina sin voluntad propia, un aparato que está programado para realizar una función específica, un avance tecnológico que satisface necesidades, son varias de las definiciones que se le da a un robot.

A partir de esto, se puede concluir que este tema es sin duda de dominio público, sin embargo, estos conceptos generalizados no garantizan que las personas tengan idea de las aplicaciones que se le puede dar a la robótica en el campo científico.

La importancia de la evolución y el desarrollo de la tecnología, radica en las necesidades que se han creado desde que el hombre existe.

Hablar de los conocimientos teóricos y prácticos inmersos en los campos científicos, es hablar de técnicas, sabiendo que el estudio de estas habilidades y destrezas es la tecnología. Es por este motivo y dentro de la disciplina de sistemas robóticos, que se ha elegido Arduino, como la plataforma a utilizarse en la realización de proyectos multidisciplinarios.

PROBLEMA INVESTIGADO

De la conversación sostenida con profesores de la carrera de electrónica, el diseño del brazo robótico despertará la iniciativa, creatividad y la indagación de los estudiantes en el campo de la Electrónica y la Robótica debido a que en los diferentes laboratorios se realiza prácticas por medio del montaje y simulación de circuitos. En ciertas asignaturas como Diseño Electrónico y Microcontroladores el estudiante realiza cada año proyectos integradores como sistemas autómatas que son indispensables dentro de la carrera, por este motivo se propuso diseñar un brazo robótico para que el estudiante pueda

profundizar el análisis sobre las nuevas tecnologías que abarca el proyecto como es la tecnología Arduino y la interfaz gráfica Labview ya que con su fácil manejo se puede crear sistemas de automatización de procesos.

Por este motivo el problema principal radica en que los Laboratorios de la Carrera de Electrónica y Telecomunicaciones de la Universidad Israel, ubicada en el Distrito Metropolitano de Quito, no cuentan con el diseño de un brazo robótico automático con tecnología ARDUINO para transportar objetos entre dos puntos fijos con monitoreo hecho en Ecuador.

Tampoco se ha implementado un Sistema Human Machine Interface (HMI) para el monitoreo del brazo robótico.

OBJETIVOS

Objetivo General

Diseño de un brazo robótico autómata controlado por un sistema human machine interface que transporte cubos de madera entre dos puntos fijos para los laboratorios de electrónica de la Universidad Israel.

Objetivos Específicos

- Analizar la estructura y componentes necesarios para la elaboración del diseño del brazo robótico.
- Diseñar un brazo robótico automático con tecnología ARDUINO para el transporte de objetos entre dos puntos fijos.
- Diseñar el código fuente en un HMI (sistema human machine interface) para el control y monitoreo del brazo robótico.

CAPÍTULO I

FUNDAMENTACIÓN TEÓRICA

Introducción

Para la elaboración del proyecto, se debe tomar en cuenta algunos conceptos y fundamentos teóricos necesarios para el estudio, análisis y diseño de un brazo robótico automático que transporte cubos de madera entre dos puntos fijos.

Para el diseño se concertará cuáles son los elementos que deben ser utilizados para un correcto funcionamiento mediante una comparación de materiales, tecnologías electrónicas y software de programación establecida con su respectiva valoración.

Marco Teórico

1.1 Robótica

De una forma u otra, la electricidad interviene en la mayoría de los aspectos de nuestra vida y, a medida que avanza la tecnología se hace cada vez más imprescindible. La robótica es una ciencia aplicada orientada a campos como diseño electrónico y mecánico, control, programación, electrónica, entre otros. (OLLERO B., 2001)

En la tabla 1.1 se muestran las leyes que Isaac Asimov, escritor y bioquímico de origen ruso estableció como medida de protección para los humanos, ante la posibilidad de una posible conspiración de las máquinas contra sus creadores. (Mosquera L., 2015)

Isaac Asimov: “la imagen de un robot es la de una máquina bien diseñada y con una seguridad garantizada, que actúa de acuerdo con tres principios”	Leyes de la Robótica	
	Primera Ley	Un robot no puede actuar contra un humano o, permitir que un ser humano sufra daños, mediante inacción.
	Segunda Ley	Un robot debe obedecer órdenes dadas por los humanos, salvo que estén en conflicto con la primera ley.
	Tercera Ley	Un robot debe protegerse a sí mismo, sin entrar en conflicto con la primera y segunda ley.

Tabla 1.1 Leyes de la Robótica.

Fuente: (OLLERO B., 2001)

La robótica se caracteriza por el desarrollo de sistemas cada vez más versátiles, flexibles y adecuados, mediante el uso de nuevos métodos de control y estructuras mecánicas. Entonces, se la puede definir como “el estudio de la construcción, ensamblaje, generación, programación y uso de los robots y autómatas en general”. (OCEANO, 1995)

1.2 Máquina Robot

Se puede considerar a un robot como una máquina complementada con un computador, con dispositivos de entrada y salida sofisticados. Una definición exacta de un robot se asocia principalmente a un mecanismo o dispositivo que puede controlarse de manera digital, mediante la ejecución de un programa almacenado en memoria. En

la tabla 1.2 se exhiben los tres elementos claves a tener en cuenta para el diseño de un robot. (OLLERO B., 2001)

Componentes del robot	**Descripción**
Programabilidad	El robot es un computador, lo que significa disponer de las capacidades de operación lógica.
Capacidad Mecánica	No ser un simple procesador de datos y que lo habilita para realizar funciones en su entorno (el robot es una máquina).
Flexibilidad	Según un amplio rango de lenguajes de programación el robot puede operar y tratar con determinados materiales u objetos.

Tabla 1.2 Elementos de un robot

Fuente: (OLLERO B., 2001)

1.3 Automatización y Robótica

Automatización y robótica son dos tecnologías relacionadas estrechamente. Las necesidades actuales de aumentar la producción y obtener productos de alta calidad, provocan que industrias busquen la manera de automatizar el trabajo basándose en sistemas complejos. En términos industriales se puede definir a la automatización como una tecnología relacionada con el desarrollo de sistemas mecánicos, electrónicos y control de producción. (ANGULO USATEGUI, 2000)

1.4 Clasificación de los Robots

La flexibilidad y utilidad del robot se determinan gracias a la potencia del software del controlador se están dentro de las limitaciones del diseño mecánico y la capacidad de los sensores. En la tabla 1.3 los robots han sido encasillados y descritos brevemente de acuerdo a su nivel de razonamiento, control, y programación. (BARRIENTOS A., PEÑÍN L., BALAGUER C., RAFAEL ARACIL., 1997)

Robots	**Descripción**
Play-back	Robots que comúnmente tienen un control de lazo abierto en cinemática y regeneran una secuencia de instrucciones grabadas.
Controlados por sensores	Toman decisiones que se basan en datos que entregan los sensores y tiene un control de lazo cerrado en su movimiento.
Controlados por visión	Manipulan objetos al utilizar información desde un sistema.
Controlados adaptablemente	Pueden reprogramar Automáticamente sus acciones gracias a los sensores, debido a que poseen una base de datos interna. Tienen un control de lazo cerrado.

Tabla 1.3 Clasificación de los Robots

Fuente: (BARRIENTOS A., PEÑÍN L., BALAGUER C., RAFAEL ARACIL., 1997)

1.5 Tipos de Robots

Los robots, sobre todo los industriales son de varios tamaños y diferente configuración, haciendo referencia a la forma física que se le da a los brazos de robot. A continuación, en la tabla 1.4 se expone algunos tipos, sobre todo los más importantes: (ANGULO USATEGUI, 2000)

Tipo de Robot	**Descripción**
Cartesiano	Cuenta con tres dispositivos deslizantes perpendiculares entre sí, hace que su movimiento sea de tipo lineal.
Cilíndrico	Se basa en una columna vertical que gira sobre una base con articulaciones lineales en el movimiento de altura y radio.
Polar	Tiene dos articulaciones rotacionales y una lineal y utiliza un brazo telescópico que oscila en torno a su eje horizontal.
De brazo Articulado	Tiene tres articulaciones sujetas a una columna que gira sobre su propia base
Antropomórfico	Tiene dos componentes rectos que simulan el brazo humano sobre una columna giratoria.
Poliarticulado o Mixto	Posee varias articulaciones y está diseñado para que su movimiento sea en un determinado espacio de trabajo.

Tabla 1.4 Tipos de Robots

Fuente: (ANGULO USATEGUI, 2000)

1.6 Mecánica del brazo

Debido a su similitud con las extremidades superiores del cuerpo humano, los manipuladores también son denominados Brazos de robot o Brazos Robóticos. (Bueno, 2009)

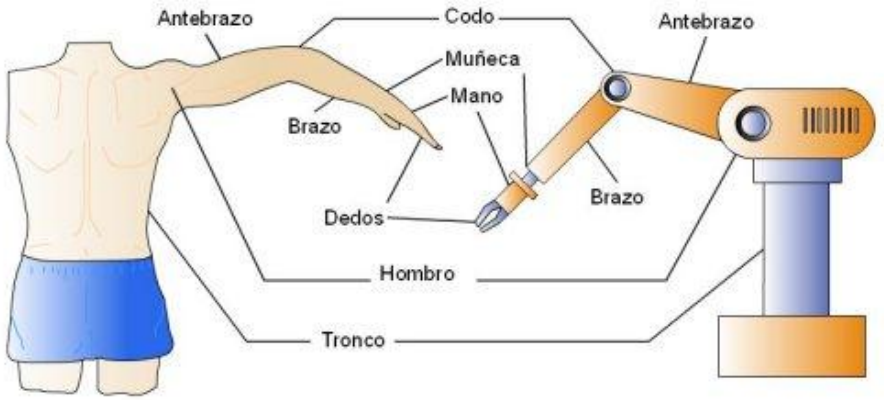


Figura 1.1 Comparación Brazo Robótico – Anatomía humana
Fuente: (Bueno, 2009)

Los robots conocidos como manipuladores son en esencia brazos articulados, es decir, es una cadena cinemática directa y abierta que está formada por una serie de eslabones congruentes conectados a través de articulaciones. (Bueno, 2009)

En la figura 1.2 los eslabones Co, C1,... Cn representan el número de eslabones que componen el manipulador. Una articulación puede ser lineal, si un eslabón se desliza sobre un eje adherido al eslabón anterior; y rotacional, si un eje gira en torno a un eje adherido al eslabón anterior. (OLLERO B., 2001)

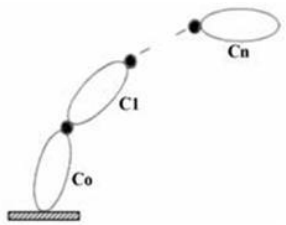


Figura 1.2 Cadena Cinemática.
Fuente: (OLLERO B., 2001)

1.7 Elemento Terminal

En el extremo final del manipulador se conecta un elemento terminal que servirá para que se realice una aplicación en particular. Es decir que el manipulador será diseñado específicamente para dicha función. El punto central del elemento terminal se denomina punto terminal. Si fuera una pinza, su punto terminal sería el centro de represión de ésta. (Bueno, 2009)

1.8 Cinemática y Dinámica

Para poder controlar y determinar el estado de un manipulador se debe establecer la posición del punto terminal (o cualquier otro punto del brazo) en relación a un sistema de coordenadas externo y fijo. Y además analizar el movimiento del brazo cuando los elementos (articulaciones, eslabones, actuadores) aplican sus fuerzas y momentos, tanto desde la parte de la cinemática como de la dinámica. (SPONG, 1989)

1.9 Cinemática

La Cinemática en los manipuladores se basa en las propiedades geométricas y temporales del movimiento del brazo articulado. A partir de dichos parámetros geométricos se especifica la posición y orientación del manipulador mediante sistemas

de referencia externos y objetos del entorno. “La cadena de cinemática abierta consiste cuando una secuencia de eslabones conecta los extremos de la misma. Mientras que la cinemática cerrada es cuando la secuencia de eslabones forma una trayectoria cerrada”. (SPONG, 1989)

1.9.1. Cinemática directa

Determina la posición y orientación del punto terminal del manipulador, con respecto a un sistema de coordenadas de referencia, conocidos los ángulos de las articulaciones y los parámetros geométricos de los demás elementos del brazo. (SPONG, 1989)

1.9.2. Cinemática Inversa

Determina la alineación que debe adoptar el manipulador para una posición y orientación del punto terminal conocido. (SPONG, 1989)

1.10 Grado de Libertad

El grado de libertad se refiere a cada una de las coordenadas independientes que son necesarias para analizar el estado mecánico del robot. En la cinemática abierta, cada par de eslabón – articulación tiene un solo grado de libertad, ya sea de rotación o traslación. (SPONG, 1989)

1.11 Dinámica

La dinámica en la robótica es fundamental para el diseño de las leyes de control apropiadas para el robot y para la evaluación del diseño y estructura del brazo, utilizando formulaciones matemáticas que analizan el comportamiento del movimiento del brazo mediante variables dinámicas que imponen el movimiento del robot como se muestra en la figura 1.3. (Abadía, 1997)

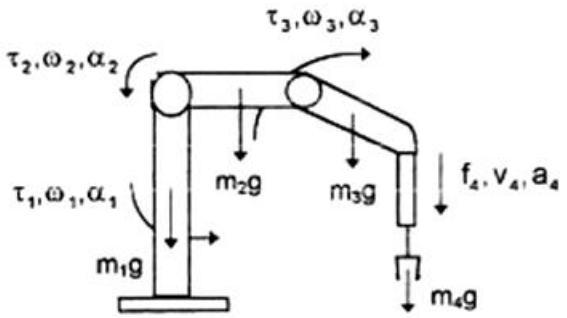


Figura 1.3 Variables dinámicas del robot.
Fuente: (Abadía, 1997)

1.11.1. Dinámica Inversa y Directa

Los métodos dinámicos consisten en dos clases de soluciones: la dinámica inversa y la directa. La dinámica inversa consiste en obtener las fuerzas y torques actuantes en cada elemento y articulación del mecanismo, mientras que la dinámica directa calcula las velocidades y aceleraciones bajo condiciones de carga. (Abadía, 1997)

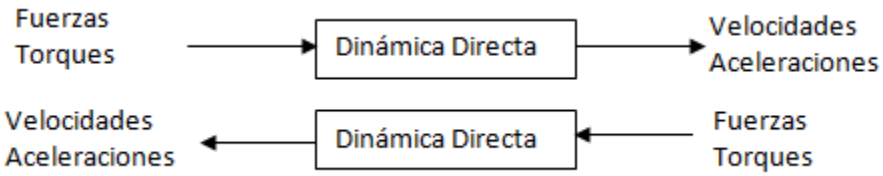


Figura 1.4 Métodos dinámicos para un robot
Fuente: (Abadía, 1997)

1.12 Servomotores

Un servomotor es un dispositivo similar a un motor de corriente continua, que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y mantenerse estable en dicha posición. (Labelec, 2009)

El control de un servo se reduce a indicar su posición mediante una señal cuadrada de voltaje. Para bloquear el servomotor en una posición, es necesario enviarle continuamente una señal, de esta forma el servo conservará su posición y se resistirá a fuerzas externas que traten de cambiarlo de posición. Generalmente el rango de giro de un servomotor cubre entre 90° y 180° de la circunferencia total. (Labelec, 2009)

1.13 Microcontroladores

El microcontrolador es un circuito integrado compuesto por tres partes fundamentales, la unidad central de proceso, memoria y unidades de entrada/salida, que en conjunto forman una microcomputadora, la cual necesita de un programa para realizar algún proceso concreto. (Sin autor, Ayuda electrónica, 2008 - 2014)

Los microcontroladores leen y ejecutan los programas que el usuario escribe dentro de un programador, es por esto que la programación es de vital importancia cuando se diseñan sistemas con estos dispositivos electrónicos. (Sin autor, Ayuda electrónica, 2008 - 2014)

1.14 Arduino

Arduino es una plataforma de Hardware que trabaja mediante código abierto, basada en una sencilla placa con entradas y salidas, y un entorno de desarrollo que implementa un lenguaje de programación de fácil manejo para el usuario, sin embargo, su condición de sistema libre ha propiciado tantas variaciones del mismo, que Arduino no es una pieza de hardware única. (Torrente, 2010)

Las funciones de Arduino pueden resumirse en tres:

- Se tiene una interfaz de entrada, que puede estar unida a los periféricos, o conectarse a ellos por medio de puertos.
- Lleva la información al microcontrolador, que es la pieza encargada de procesar los datos. Este microcontrolador varía debido a las necesidades del proyecto.
- Por último, está la interfaz de salida, que transporta la información procesada a los periféricos encargados de mostrar la versión final de los datos. (Torrente, 2010)

1.15 HMI: Human Machine Interface

HMI es la abreviación en ingles de Interfaz Hombre Máquina, estos sistemas se los considera como “ventanas” de un proceso. Esta ventana puede estar en dispositivos especiales como paneles de operador o en una computadora. (Rojas, 2012)

Esta interfaz permite que el usuario, opere y maneje la máquina, observe el estado del equipo e intervenga en el proceso. La información se proporciona por medio de paneles de control con señales luminosas, campos de visualización o botones, o por medio de un software que utiliza un sistema de visualización que se ejecuta en una terminal. (Rojas, 2012)

1.16 Labview

LabView es un software Comercial propiedad de National Instruments, que tiene un entorno de programación gráfico, es decir se basa en un lenguaje “G”, el cual permite que los programas no se escriban, se dibujen. Es decir que esté programa literalmente traslada los algoritmos en una forma gráfica, haciendo de esto una sencilla programación. (Sánchez A, 2010)

La elaboración de un sistema automatizado requiere del uso de diferentes tipos de mecanismos, algunas de sus aplicaciones son la creación de software de control de procesos, y además la conexión de controladores y actuadores, sensores, motores, servomotores y demás. (Sánchez A, 2010)

1.17 Comparación de Microcontroladores

A continuación en la tabla 1.5 se muestran las ventajas y desventajas de tres tipos de microcontroladores:

Microcontrolador	Ventajas	Desventajas	Valoración (1-5)
ARDUINO	➢ Lenguaje de programación simplificado. ➢ Posee extensiones de Hardware para conexiones externas. ➢ Compilación y depuración directa del Programa al microcontrolador	➢ Costo de adquisición. ➢ Sensibilidad en el manejo del equipo	5
AVR	➢ Costo de adquisición. ➢ Tamaño del equipo.	➢ Lenguaje de programación extenso.	3
PIC	➢ Costo de adquisición ➢ Tamaño del equipo	➢ Lenguaje de programación extenso.	2

Tabla 1.5 Tabla comparativa de los microcontroladores
Fuente: Autor

A través de la tabla 1.5 se comparan tres tipos de microcontroladores. Arduino tiene la mejor valoración de entre los tres debido a la facilidad de programación que tiene y las extensiones de hardware que se pueden adicionar al dispositivo para conexiones externas obteniendo mayores funcionalidades. Es por esto que se ha elegido a la plataforma Arduino para el desarrollo del presente proyecto. (Mosquera L., 2015)

Figura 1.5 Microcontrolador Arduino UNO
Fuente: (Torrente, 2010)

1.17.1. Arduino Shield

Arduino Shield se encarga de mejorar mediante funcionalidades adicionales a la placa Arduino aumentando sus capacidades, por ejemplo, comunicación (con otras placas o el medio), o al momento de gestionar más sistemas. (webelectro, 2015)

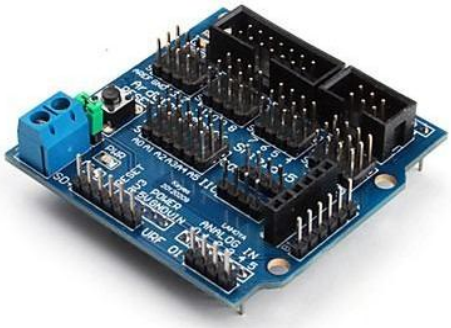


Figura 1.6 Shield Arduino Sensor V.5
Fuente: (webelectro, 2015)

Características técnicas del Microcontrolador Arduino UNO	Características Arduino Sensor Shield
➢ **Microcontrolador:** ATmega328 ➢ **Voltaje de funcionamiento:** 5V ➢ **Alimentación:** 7-12 V ➢ **Corriente DC (I/O pin):** 50 mA ➢ **Memoria FLASH:** 32KB de los cuales 0.5KB son usados para arranque ➢ **SRAM:** 2KB ➢ **EEPROM:** 1KB ➢ **Velocidad del reloj:** 16Mhz	➢ Controlador de secuencia de interfaz analógica y digital. ➢ Controlador de interfaz i2C ➢ Interfaz controladora de 32 servos ➢ Interfaz de comunicación del módulo Bluetooth ➢ Interfaz APC220 WIRELESS ➢ Interfaz rs232. ➢ Interfaz de comunicación del módulo de tarjeta SD. ➢ Interfaz de comunicación APC220 inalámbrica módulo de RF. ➢ Interfaces 12864 LCD serie y paralelo ➢ Interfaz sensor de ultrasonidos.

Tabla 1.6. Características técnicas de ARDUINO UNO y ARDUINO SHIELD
Fuente: (Torrente, 2010)

1.18 Comparación de HMI

A continuación en la tabla 1.7 se muestran las ventajas y desventajas de tres tipos de HMI:

HMI	**Ventajas**	**Desventajas**	**Valoración (1-5)**
LabVIEW (National Insturments)	• Interfaz para plataforma Arduino. • Programación gráfica por diagramas de bloques	• Se requiere Licencia de software	5
LabWINDOWS (National Instruments)	• Interfaz para plataforma Arduino.	• Costo de adquisición • Programación por código fuente. • Se requiere Licencia de software	2
Blender	• Software Libre • Imágenes en 3D	• Programación en Lenguaje Phyton • No cuenta con Interfaz para plataforma Arduino	1

Tabla 1.7. Tabla comparativa de HMIs
Fuente: (webelectro, 2015)

En la tabla 1.7 se comparan tres tipos de HMIs. Debido a la facilidad de programación gráfica por diagrama de bloques y por la compatibilad que tiene para trabajar conjuntamente con la plataforma Arduino se utilizará el sistema HMI LabVIEW versión 2011 para realizar una interfaz de usuario que puede interactuar con el manipulador. (Mosquera L., 2015)

Hoy es posible diseñar sistemas de automatización y medida de bajo costo. La programación gráfica con Labview permite a los no programadores un método fácil para implementar aplicaciones complejas de test, medida y automatización. Con Labview el software define el sistema. (Tutorial LabView.pdf)

Figura 1.7 Ventana de Inicio LabVIEW 2011
Fuente: (ni, 2012)

1.19 Comparación de Servomotores

A continuación en la tabla 1.8 se indican las ventajas y desventajas de tres tipos de servomotores:

Servomotores	Ventajas	Desventajas	Valoración (1-5)
Hitec Ms-311	➢ Características mecánicas y electrónicas	➢ Costo de adquisición.	5
SG90	➢ Costo de adquisición ➢ Características de precisión	➢ Potencia	5
Hitec Hs-422	➢ Características mecánicas y electrónicas	➢ Costo de adquisición. ➢ Tamaño	3

Tabla 1.8. Tabla comparativa de los servomotores
Fuente: Autor

La tabla 1.8 compara tres tipos de servomotores, debido a sus características mecánicas y electrónicas los servomotores marca Hitec Ms-311 y el SG90 son ideales para la implementación del brazo robótico. (Mosquera L., 2015)

Figura 1.8 Servomotor Hitec Ms-311
Fuente: (Servodatabase, 2009)

1.19.1. Servomotor SG90

Las características de precisión del servomotor SG90 son ideales para utilizarlo en el elemento terminal del brazo robótico. En ese punto no se requiere de una gran potencia. (Mosquera L., 2015)

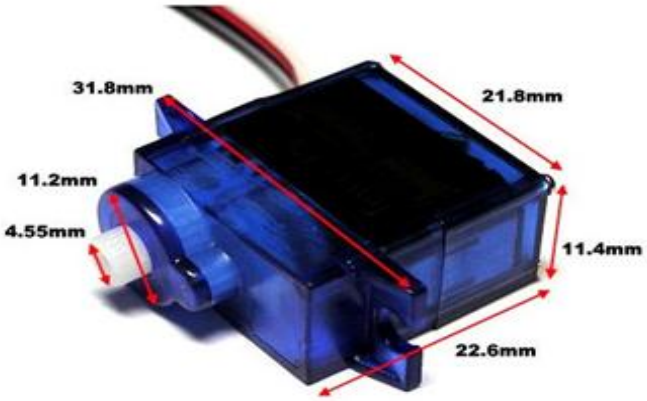


Figura 1.9 Servomotor SG90
Fuente: (Botscience, 2013)

Características Técnicas del Servomotor Hitec Ms-311	Características Técnicas del servomotor SG90
➢ Largo: 39.9 mm ➢ Ancho: 19.8 mm ➢ Alto: 36.3 mm ➢ Peso: 43 g ➢ Velocidad de giro 4.8 V: 0.19 seg/60° ➢ Velocidad de giro 6.0 V: 0.15 seg/60°	➢ Largo: 22 mm ➢ Ancho: 11.5 mm ➢ Alto: 27 mm ➢ Peso: 9 g ➢ Velocidad de giro: 0.12 seg/60° ➢ Voltaje de Operación: 4 V – 6 V

Tabla 1.9. Características técnicas de los Servomotores.
Fuente: (Botscience, 2013)

CAPÍTULO II

BREVE DESCRIPCIÓN DEL PROCESO INVESTIGATIVO REALIZADO PARA EL DISEÑO DE UN BRAZO ROBÓTICO AUTÓMATA

METODOLOGÍA DE LA INVESTIGACIÓN

El problema principal dentro del proceso investigado se centra en que los Laboratorios de la Carrera de Electrónica y Telecomunicaciones de la Universidad Israel en Quito no cuentan con el diseño de un brazo robótico automático con tecnología ARDUINO para transportar objetos entre dos puntos fijos y que sea monitoreado por una interfaz gráfica (HMI).

Para la elaboración del proyecto se propuso como objetivo general diseñar un brazo robótico que transporte cubos de madera de un lugar a otro despertando la iniciativa, creatividad y la indagación de los estudiantes en el campo de la Electrónica y la Robótica. Los objetivos específicos se plantearon para analizar la estructura y componentes necesarios para la elaboración del diseño del brazo robótico.

La idea a defender es: Al diseñar un brazo robótico que transporte cubos de madera de un lugar a otro para los laboratorios de la UISRAEL se despertará el interés y la creatividad en los estudiantes en utilizar nuevas tecnologías.

Siendo la variable independiente: El diseño del brazo robótico que transporte cubos de madera de un lugar a otro para los laboratorios de la UISRAEL. Y la variable dependiente: El interés y la creatividad en los estudiantes en utilizar nuevas tecnologías.

En busca del conocimiento y a través de métodos, este proyecto está basado en una metodología de investigación, por lo que se han utilizado los siguientes métodos:

Con el método analítico se analizó minuciosamente la estructura y componentes necesarios para elaborar el diseño del brazo robótico.

Utilizando el método sintético se pudo comprender la esencia de lo que ya se conoce en todas sus partes y particularidades, debido a que a partir de los elementos que se distinguen en el análisis se pudo indagar en los componentes y elementos que constituyen el proyecto.

Por último el método de deducción permitió que se generalicen las explicaciones y descripciones inducidas que se aplican para comprobar la validez del proyecto.

CAPÍTULO III

PRESENTACIÓN DE RESULTADOS

DESCRIPCIÓN

El diseño de la estructura física del brazo robótico se asemejará a una de las extremidades superiores del cuerpo humano, ya que son en esencia brazos articulados formados por una serie de eslabones congruentes conectados a través de articulaciones teniendo la capacidad de realizar movimientos determinados. En el segmento final de la estructura del brazo se debe colocar una pinza que sujete los objetos que serán desplazados, éstos deberán tener forma de cubo, debido a que la estructura de la pinza facilita su manipulación transportándolos de un punto a otro.

A través del diagrama de bloques de la interfaz gráfica se puede diseñar mediante programación cada movimiento del brazo robótico de forma manual y automática, con la finalidad de que el brazo obtenga mayores aplicaciones mejorando su funcionamiento.

Dentro de la interfaz LabView se debe instalar el toolkit de Arduino que proporciona herramientas útiles para diseñar el sistema de control del brazo robótico y así controlar el movimiento automático y manual del mismo.

DISEÑO

3.1. Diagrama general del sistema

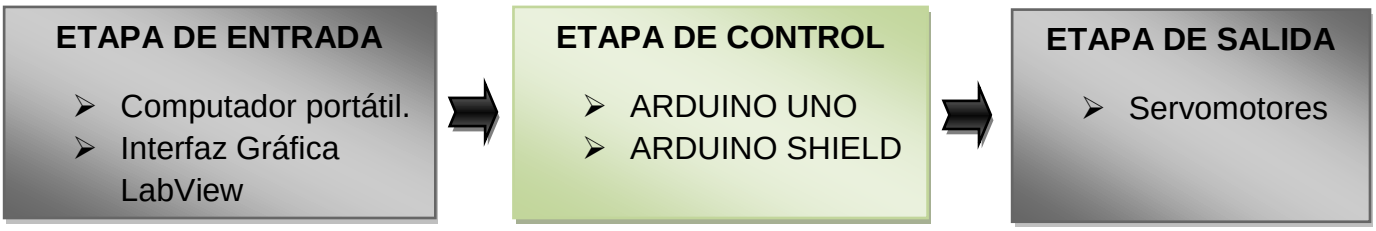


Figura 3.1 Diagrama general del sistema
Fuente: Investigador

3.1.1. Etapa de Entrada

La etapa de entrada del diagrama del sistema consta de la interfaz LabView donde se realiza el diseño del proceso de automatización y control del brazo robótico utilizando el diagrama de bloques y el panel frontal de dicho software. En esta etapa mediante el computador portátil y la interfaz adecuada se podrá manipular el movimiento de los servomotores que se encuentran en cada una de las estructuras del brazo.

3.1.2. Etapa de Control

La etapa de control está constituida por el microcontrolador ARDUINO UNO y conectado a un ARDUINO SHIELD directamente ya que la interconexión entre sus pines proyecta una mayor capacidad en su funcionamiento. En esta etapa es donde se guarda toda la programación realizada en la interfaz gráfica correspondiente al brazo robótico.

3.1.3. Etapa de Salida

Esta última etapa consta de servomotores que servirán para el movimiento y funcionamiento de las partes que conforman el brazo robótico (base, hombro, mano, muñeca y pinza). En el parte final del brazo irá una pinza para sujetar los objetos que serán desplazados.

3.2. Diagrama de flujo del funcionamiento del sistema

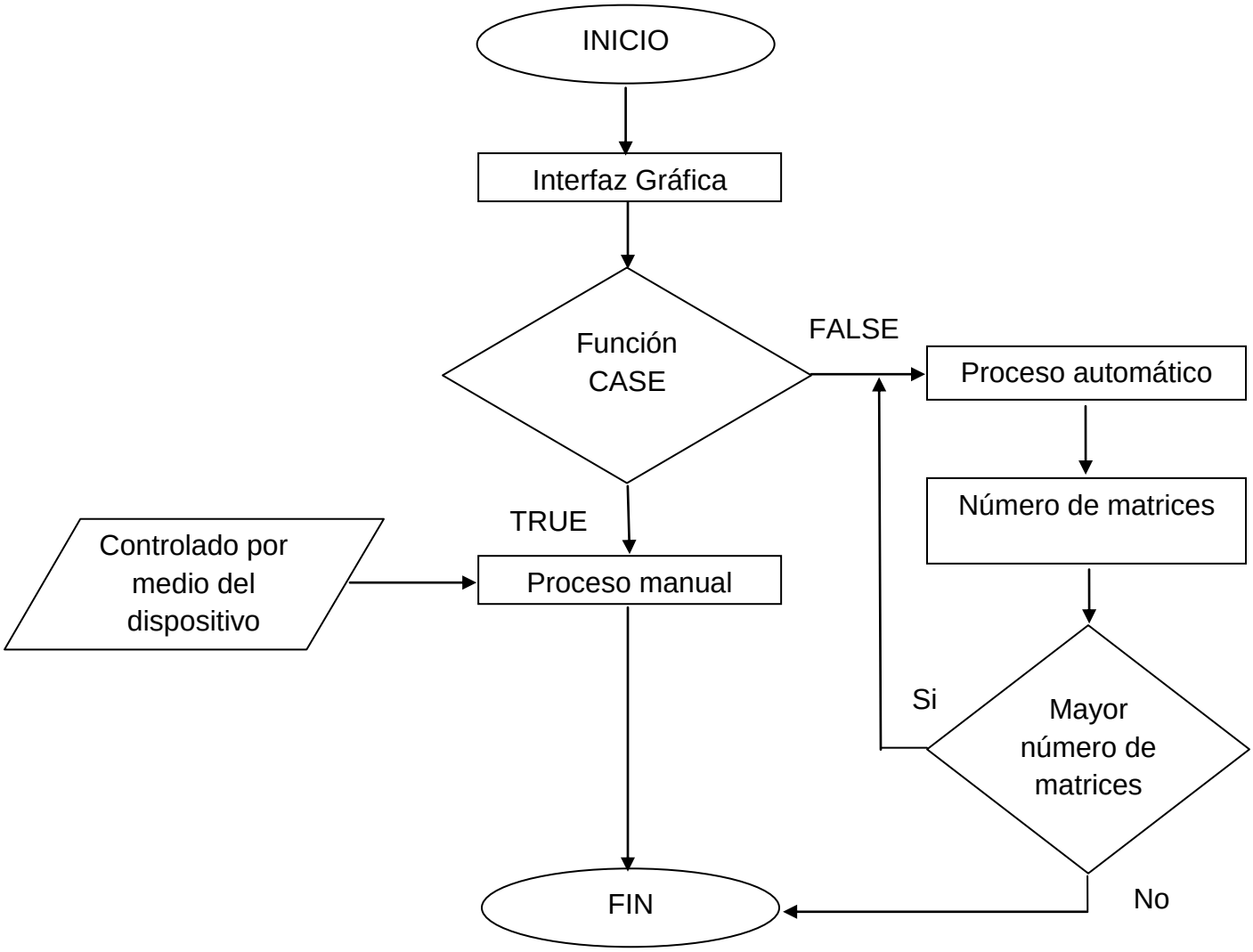


Figura 3.2 Diagrama de flujo del sistema.
Fuente: Investigador

3.3. Diseño estructural del brazo robótico

Fue diseñado en el programa Solid Works, donde se presenta un prototipo mediante dimensiones y medidas que son válidas para el desarrollo del proyecto.

El prototipo del brazo manipulador ensamblado con todas sus partes mecánicas y servomotores queda como se muestra en la figura 3.3:

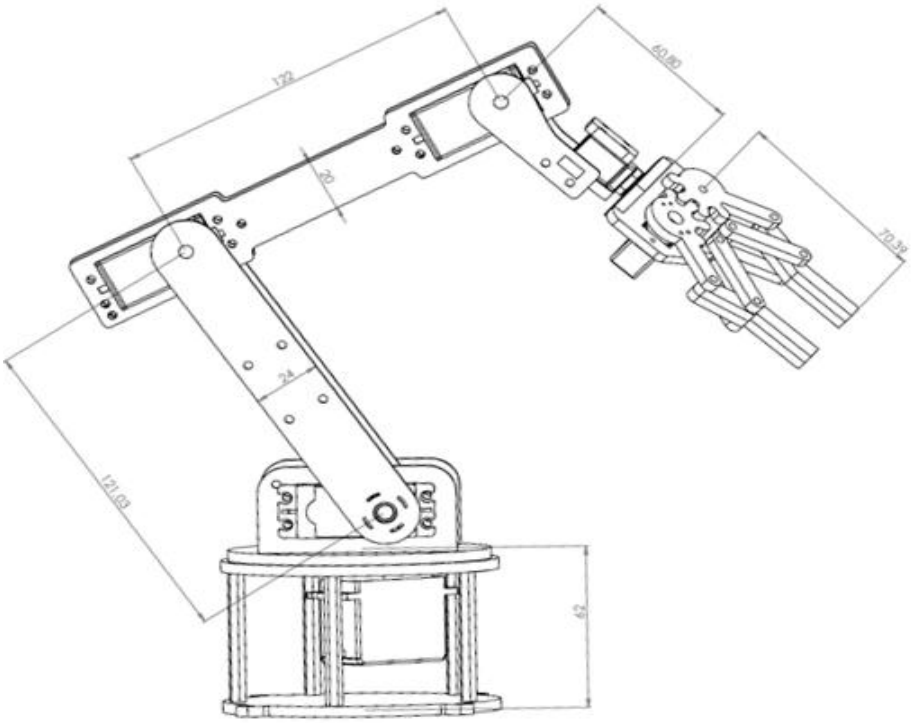


Figura 3.3 Diseño Estructural del brazo robótico
Fuente: Investigador

3.4. Diseño de la estructura mecánica del brazo

La estructura mecánica del brazo robótico debe ser elaborada con láminas de acrílico una variante de plástico más flexible de lo normal que, con su fácil uso se las puede trabajar elaborando diferentes diseños de estructuras en los proyectos, además puede

permanecer a la intemperie durante mucho tiempo sin sufrir daños en su estructura y coloración, mucho más resistente que el vidrio, lo cual no es fácil que se rompa y de esta manera se evita el riesgos de lesiones.

3.5. Diseño del Elemento Terminal

3.5.1. Mecanismo manipulador

Como se estableció en la fundamentación teórica, el elemento terminal es un dispositivo que se une a la muñeca del brazo del robot para que lleve a cabo la realización de la función específica del robot. Sin embargo, para el diseño del brazo se debe tomar en cuenta que el elemento terminal debe soportar una determinada capacidad de carga.

Al final de la estructura del brazo robótico se diseñó una pinza la cual retendrá al objeto al momento de transportarlo.

La pinza cuenta con dos “dedos” que están fijos entre sí por un sistema de engranes, uno de ellos conectado a un servomotor (figura 3.4).

Este mecanismo se ejecuta cuando el servomotor conectado a uno de los engranes gira en sentido horario, haciendo girar al otro engrane en sentido anti-horario logrando que de esta forma la pinza se abra o se cierre.

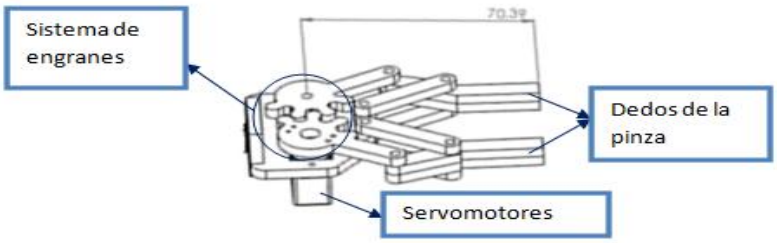


Figura 3.4 Mecanismo del Elemento Terminal
Fuente: Investigador

3.5.2. Funcionamiento y Requerimientos del elemento terminal

Como se describió anteriormente la pinza deberá contar con dos “dedos”, los cuales servirán para poder sujetar adecuadamente el objeto, sin dañarlo ni arrojarlo al momento de transportarlo.

La máxima y mínima distancia que existe entre los “dedos” de la pinza se fija por su configuración mecánica, además de otros aspectos importantes, como es el peso del objeto o la fuerza máxima que pueden ejercer las articulaciones mientras se transporta el objeto. A demás se debe tomar en cuenta la estabilidad de la pinza el momento de retener el objeto.

Para sostener el objeto, los “dedos” de la pinza se mueven de tal forma que se acercan el uno al otro, hasta que comprimen el objeto, la fuerza que ejerce cada uno en sentido contrario limita cualquier movimiento del objeto logrando retener el mismo sin dejarlo caer.

Se debe sujetar al objeto desde su centro de gravedad ya que así se anula todo movimiento que pueda generarse debido a su peso.

3.6. Servomotores y Fuente de Poder Externa

Los servomotores son de gran importancia ya que cada uno se ubica en cada eje del manipulador (base, hombro, codo, muñeca y pinza) deberán utilizar un voltaje dc de 5v y una corriente que esté entre 0A a 1A, por lo que se deberá utilizar una fuente de alimentación externa con estas características y así evitar el uso de corriente que pasa por el dispositivo el momento de conectarse con la placa Arduino debido a que ésta es muy variable y puede afectar el funcionamiento de los servomotores.

3.7. Diseño del software

La interfaz usada para el diseño del código de control del brazo robótico fue HMI LabVIEW 2011, interfaz compatible con ARDUINO, que posee una plataforma de programación basada en diagramas de bloque aplicados en sistemas de automatización de procesos. (Sánchez A, 2010)

3.7.1. Panel Frontal de la interfaz gráfica

Interfaz gráfica que acumula las entradas originadas por el usuario y representa las salidas que proporciona el programa. En la figura 3.5 se muestra el panel frontal que

está formado por pulsadores, indicadores, potenciómetros, gráficos, entre otros. Estos pueden estar definicos como un control o un indicador.

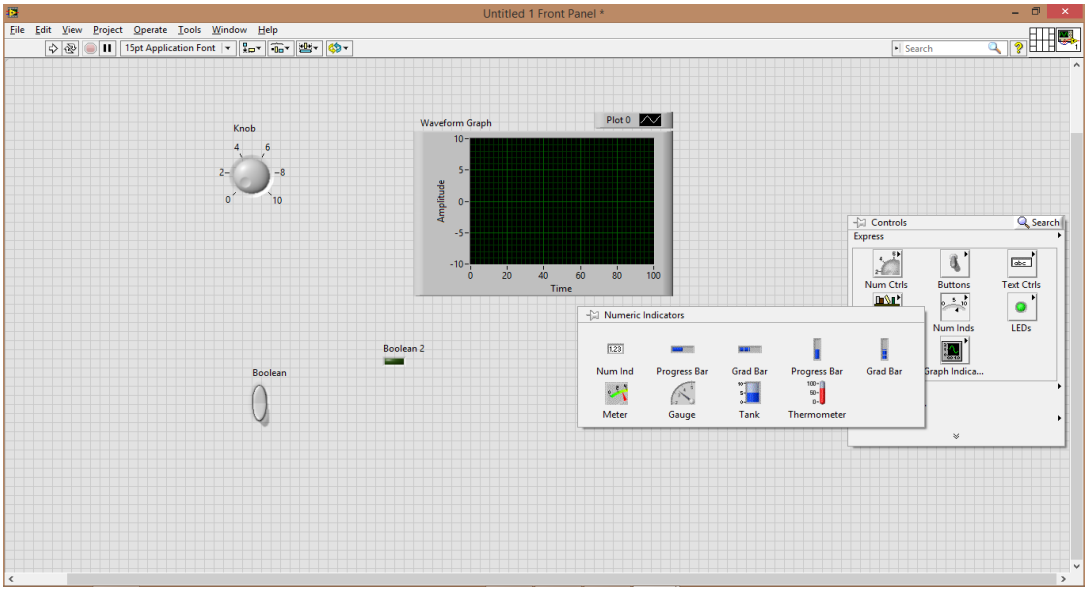


Figura 3.5 Panel frontal de LabVIEW
Fuente: Investigador

3.7.2. Diagrama de bloques de la interfaz gráfica

Dentro del diagrama de bloques se establece el código fuente, que es donde se realiza la implementación del programa para el control y realización de cualquier proceso de entradas y salidas que se crean en el panel frontal. Incluye estructuras y funciones integradas en las librerías de LabView (figura 3.6).

Lo más importante es la estructura, debido a que los bucles y las declaraciones casuales en lenguajes, ejecutan el código que contiene de forma repetitiva o condicional (for, while, case, entre otros).

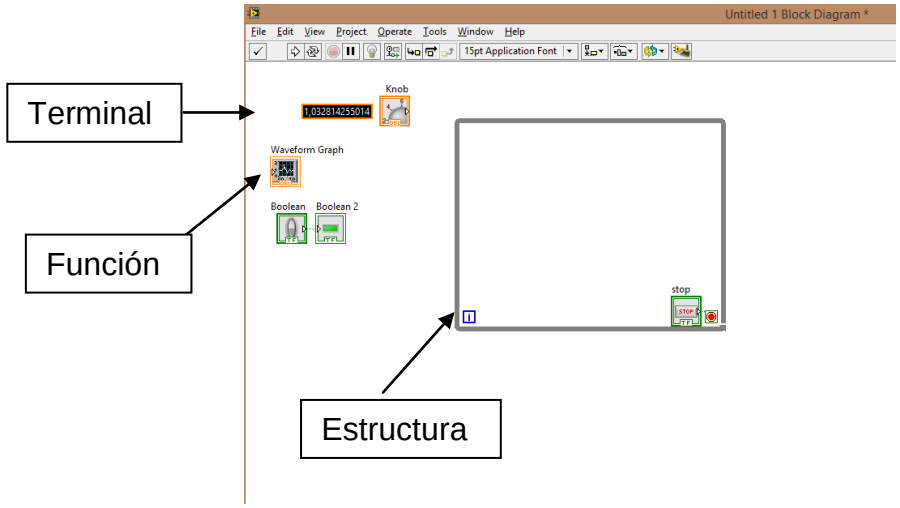


Figura 3.6 Diagrama de bloques de LabVIEW
Fuente: Investigador

3.8. VI Package Manager

National Instruments, empresa propietaria de la interfaz LabView ofrece herramientas y aplicaciones adicionales denominadas “toolkits”, que facilitan aún más el diseño de programación en esta interfaz el momento de conectarlos con otros dispositivos, en este caso Arduino. Es decir una vez instalado el toolkit de Arduino en la interfaz HMI LabView se obtienen nuevas herramientas gráficas dentro del diagrama de bloques que reemplazan la programación de código fuente que debería realizarse desde la plataforma Arduino, diseñando el código o programa solamente desde la interfaz.

Para instalar dicho toolkit es necesario utilizar el program VI package manager, también propiedad de National Instruments mostrado en la figura 3.7. En este programa se busca e instala de forma rápida y sencilla el toolkit de Arduino para Labview.

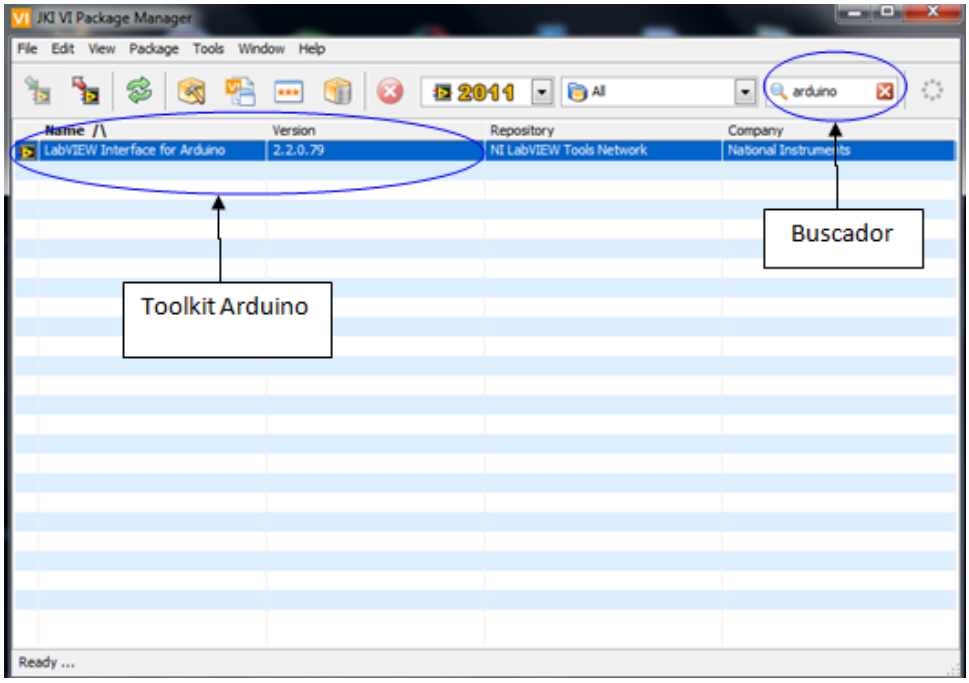


Figura 3.7 Instalación Toolkit Arduino para Labview
Fuente: Investigador

Mediante esta herramienta llamada (Labview Interface for Arduino) se puede obtener diagramas de bloques que sirven para el diseño de la programación en labview y reemplazan el código fuente de Arduino; por ejemplo, el diagrama de bloque INIT que se muestra en la figura 3.8 inicia Arduino ejecutándolo desde la interfaz de Labview para obtener el Arduino Sketch en forma de diagrama de bloque. Éstas y otras funciones de Arduino se ejecutan a través de la interfaz de Labview que se muestra en la figura 3.9.

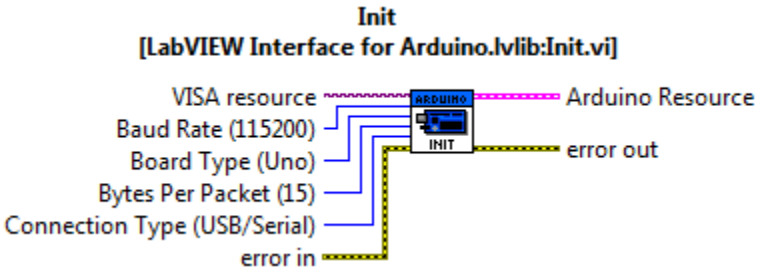


Figura 3.8. Diagrama de bloque INIT
Fuente: Investigador

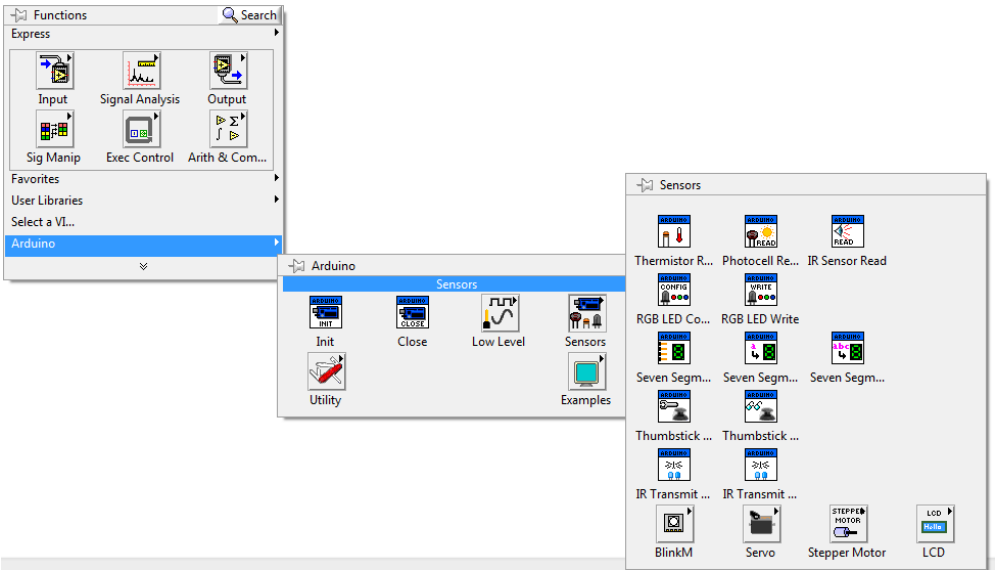


Figura 3.9. Diagramas de bloque para el funcionamiento de Arduino
Fuente: Investigador

3.9. NI VISA

Este programa, propiedad de National Instruments, es un controlador que permite la comunicación entre Labview con otros dispositivos. VISA provee la interfaz de programación entre el hardware y ambientes de desarrollo como LabView.

Este programa identifica el Puerto COM de tal forma que automaticamente establece la conexión con Arduino mediante un cable USB al puerto de la PC. De esta manera LabView reconoce el dispositivo en este caso Arduino UNO.

Para que esto se lleve a cabo en la interfaz de LabView, la función *INIT* de Arduino, que se encuentra dentro del panel de diagrama de bloques consta de un pin llamado “VISA Resource” en el cual se debe crear una constante que vendrá a ser el Puerto COM que es el que establecerá la conexión con Arduino, y que previamente lo identifica Labview por medio de NI VISA.

Al hacer clic derecho en el pin VISA resource de la función INIT se elige la opción “create” y luego “Constant” para crear la constante mencionada con anterioridad (figura 3.10).

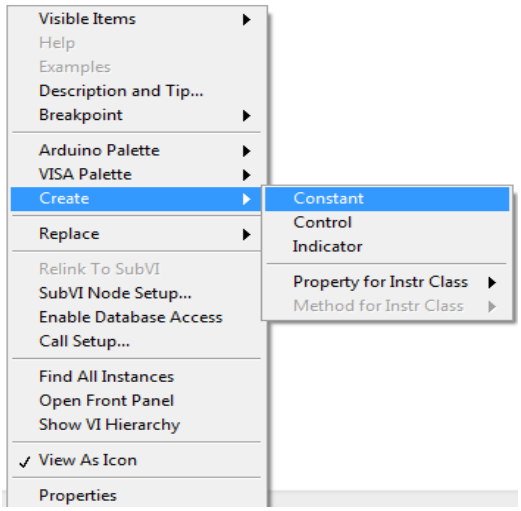


Figura 3.10. Creación de constante para identificación de Puerto COM
Fuente: Investigador

Después de realizar este proceso aparece un rectángulo de color morado conectado al pin *VISA resource* mostrado en la figura 3.11, y al hacer clic en la flecha del recuadro

deben aparecer los puertos COM que contiene la computadora. Se deberá seleccionar el puerto al cual se conectara el Arduino. En caso de no existir ningún puerto existe la opción “Refresh” que sirve para actualizar a tiempo el programa y tratar de que logre reconocer los puertos.

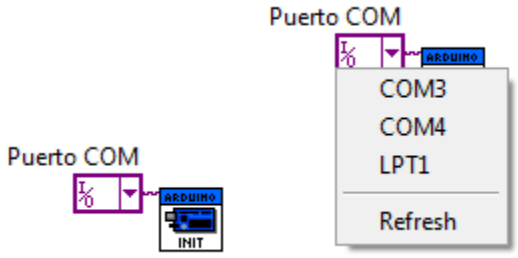


Figura 3.11. Puerto COM conectado al pin VISA Resource de la función INT
Fuente: Investigador.

3.10. Nomenclatura y Configuración de los Servomotores

En la tabla 3.2 se muestra el número del servomotor que estará ubicado en las diferentes partes del manipulador y que servirá de referencia durante la configuración de los mismos y programación del software:

Cabe recalcar que en la estructura del “hombro” del manipulador se utilizan dos servomotores que se accionarán conjuntamente en paralelo debido a que es en ese punto donde se centra la mayor fuerza que debe ejercer el manipulador el momento de cargar y transportar un objeto.

Parte del Manipulador	# Servomotor	
Base	0	
Hombro	1	2
Codo	3	
Mano	4	
Muñeca	5	
Pinza	6	

Tabla 3.2. Nomenclatura de Servomotores para el cada parte del manipulador
Fuente: Investigador

Antes de comenzar con la programación del proceso del funcionamiento del brazo robótico mediante los movimientos de los servomotores se debe establecer el número de servomotores a utilizar mediante la función Set number of servos, esta función que se muestra en la figura 3.12 es parte del diagrama de bloques.

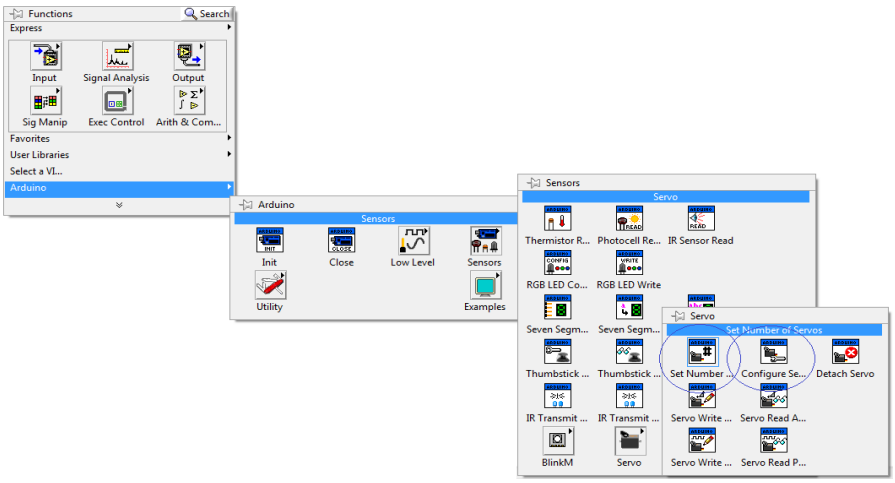


Figura 3.12. Creación de funciones para los servomotores
Fuente: Investigador

En el pin Number of servos se especifican con una constante que el número de servos a utilizar, en este caso son 7 como se muestra en la figura 3.13.

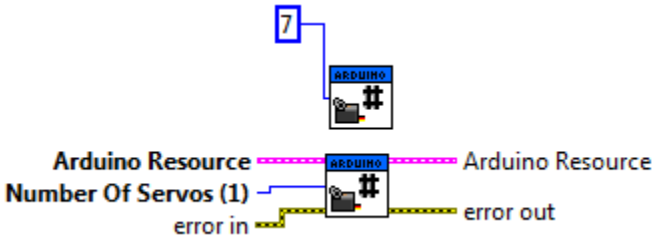


Figura 3.13. Función Set Number of Servos
Fuente: Investigador

Con la función *Configure servo* (figura 3.14), se configura el número de servo que corresponde a cada uno dentro de su propio diagrama de bloques, y además a qué pin de Arduino irá conectado el pin de señal de cada Servomotor. Para el diseño sólo se tomó en cuenta los pines digitales de Arduino para la conexión de los servomotores.

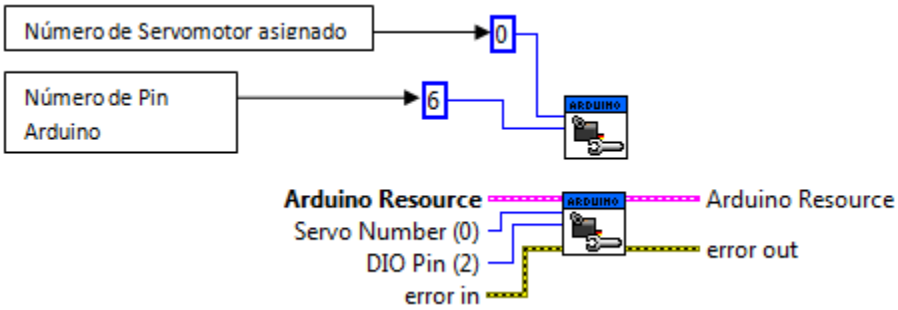


Figura 3.14. Configuración de los Servomotores
Fuente: Investigador

Como se mencionó anteriormente, se debe configurar cada servomotor con la función *configure servo* mostrada en la figura 3.15.

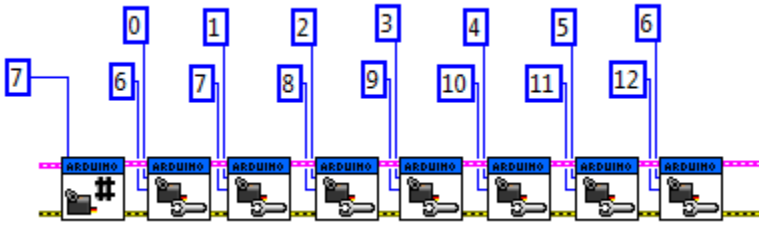


Figura 3.15. Nomenclatura y Configuración de cada uno de los Servomotores
Fuente: Investigador

Una vez conectados los diagramas de bloque de los servomotores a la función INIT de Arduino se debe tomar en cuenta lo siguiente: el pin *Arduino Resource* y el pin *error* deben estar conectados entre cada función para que Labview reconozca qué función es parte de la programación y trate de seguir un orden respectivo (figura 3.16).

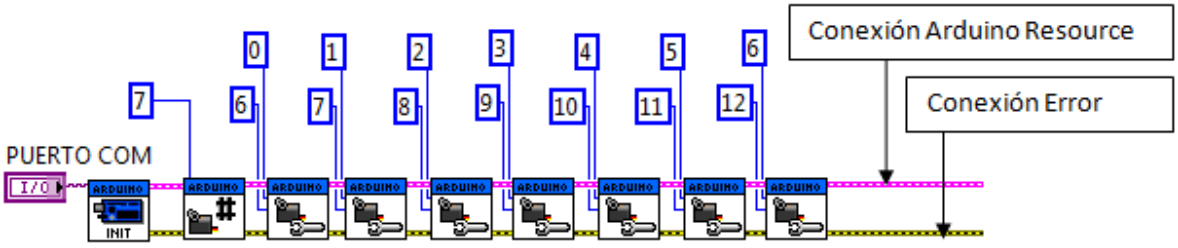


Figura 3.16 Nomenclatura y Configuración de cada uno de los Servomotores
Fuente: Investigador

3.11. Flujo de control

El flujo de control para el diseño del software viene dado por un bucle el cual permitirá que el código se ejecute repetidamente en base a una determinada condición. En LabView dicho bucle se denomina *While Loop*, y en programación se lo puede tomar como una repetición de la sentencia “if”.

A continuación en la figura 3.17 se muestra como está representado gráficamente el bucle *While Loop* dentro del panel de diagrama de bloques en LabView.

El *While Loop* no es más que un recuadro en el cual se puede insertar los diagramas de bloques respectivos y formar un proceso repetitivo a través de una condición. Además cuenta con una terminal de iteraciones, el cual puede indicar el número de veces que se completa el bucle. También viene incorporado un botón de Stop o Parada para finalizar el proceso.

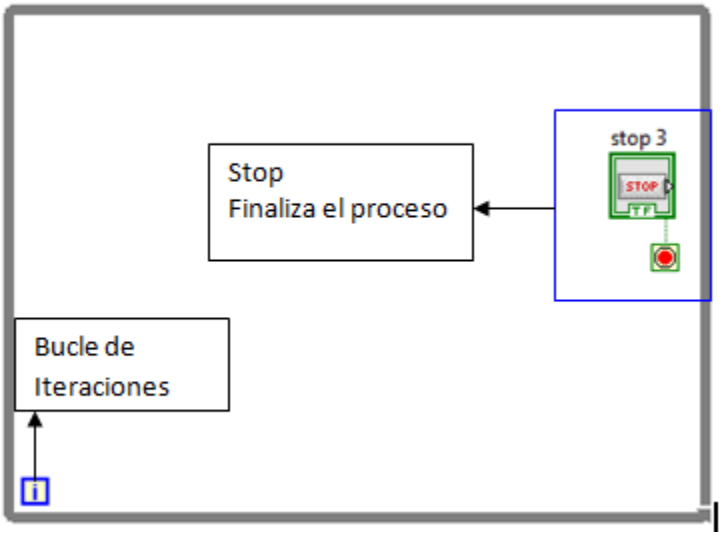


Figura 3.17. Representación Gráfica While Loop
Fuente: Investigador

Una ventaja que se logra a través de esta interfaz gráfica de LabView es que tanto los cables *Arduino Resource* y *error* de las funciones Arduino se las conecta en cualquier punto del marco del cuadro del *While Loop* como se muestra en la figura 3.18, evitando que el panel de diagrama de bloques esté lleno de conexiones (cables) mejorando la visibilidad de los diagramas de bloques principales y secundarios.

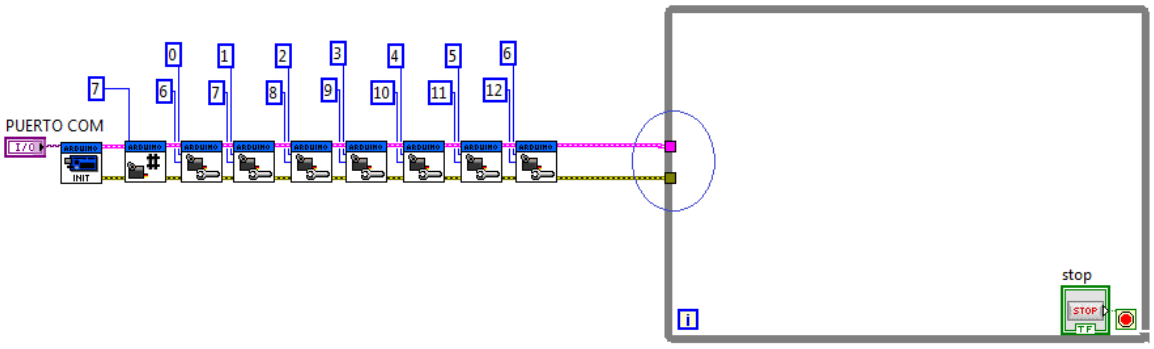


Figura 3.18. Conexión Arduino Resorce - While Loop
Fuente: Investigador.

Dentro del bucle se crea una función *case estructure* el cuál se compone de un bloque de código y una condición. La condición puede ser verdadera o falsa. Si la condición es **verdadera** el usuario puede controlar el brazo robótico de forma **manual** y si es **falsa** el brazo realiza el movimiento de forma **autómatica**.

Dentro de la función *case estructure* existe una pestaña que ayuda a visualizar de manera individual cada condición facilitando la programación (figura 3.19). Esta función evalúa la condición de entrada, de tal manera que si la condición es verdadera, ejecuta el diagrama de bloques diseñado en la pestaña *true,* y se repite hasta que la condición cambie a *false*.

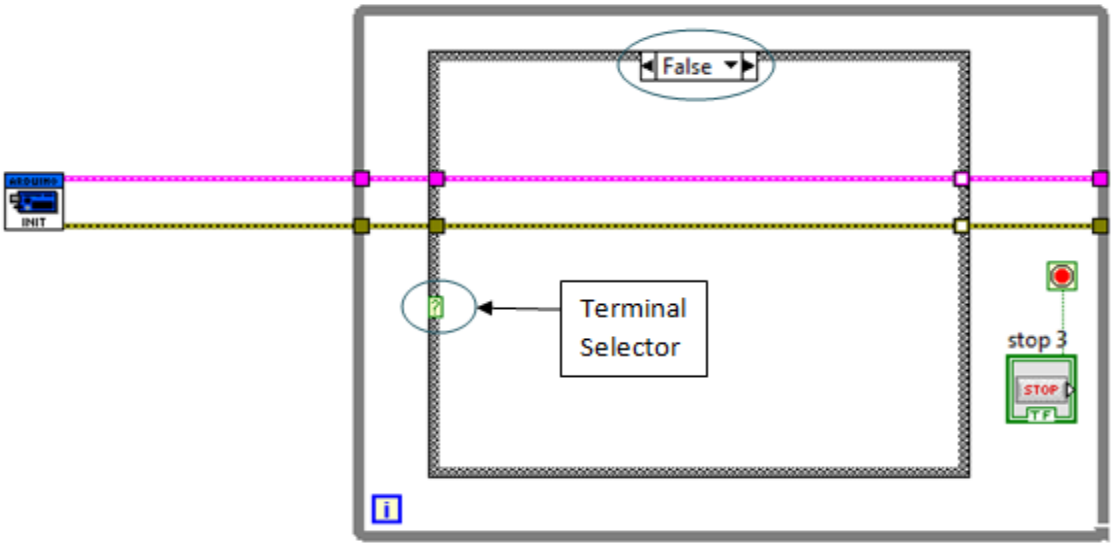


Figura 3.19. Función Case
Fuente: Investigador

El terminal selector evalúa la condición que viene dada por un botón o switch que en forma booleana actuará como verdadero o falso según la posición en que se encuentre (figura 3.20).

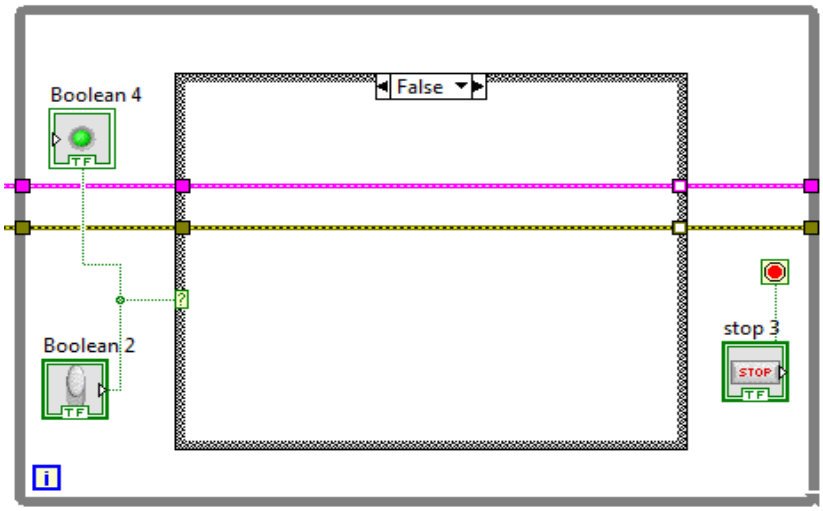


Figura 3.20. Condiciones *true* or *false*
Fuente: Investigador

Para cerrar el proceso dentro de la interfaz gráfica que reemplaza al código Arduino se necesita de la función Arduino llamada "*Close"*, la cual cierra la conexión activa del Arduino UNO (figura 3.21). Esta también se conecta con el pin *Arduino Resource* y el pin *error*.

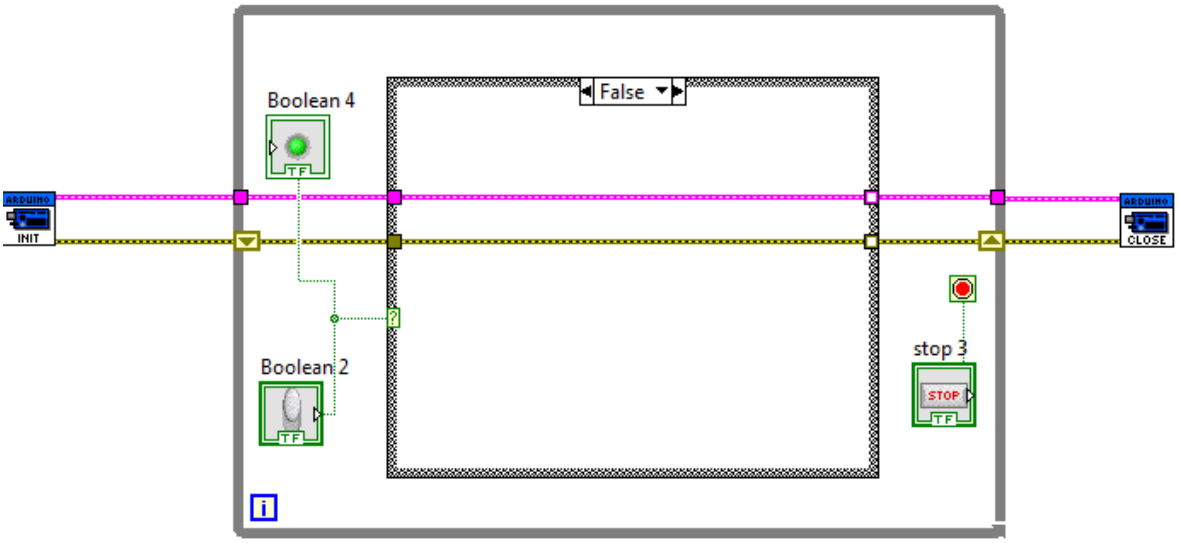


Figura 3.21. Diagrama Entrada – Flujo de Control – Close
Fuente: Investigador

3.12. Control Manual del brazo robótico

Dentro de la función *Case* si la condición es verdadera los servomotores podrán ser controlados por el usuario desde el Panel Frontal mediante indicadores dando movimiento a cada segmento del manipulador, cabe recalcar que el panel frontal será la interfaz gráfica que el usuario podrá visualizar para poder controlar el brazo robótico accionando el control manual.

Para poder controlar los movimientos del brazo róbotico de forma manual, los cuales están dados principalmente por la posición de los servomotores, se utiliza la función Arduino *Servo Write Angle* que se muestra en la figura 3.23, la cual contiene dos pines importantes:

- El primer pin especifica el número de servomotor.
- El segundo pin establece la posición angular dada en grados en la que se encuentra el mismo.

Figura 3.22. Función Servo Write Angle
Fuente: Investigador

Para establecer la posición angular se utiliza un deslizador, que en el panel frontal se convierte en un indicador que actúa como un control numérico permitiendo seleccionar un valor de entre un determinado rango y así mover la posición del servomotor (figura 3.23). El valor en este caso viene a ser el ángulo en el que el usuario desea posicionar al servomotor.

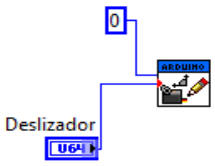


Figura 3.23 Configuración Posición Angular de los servomotores
Fuente: Investigador

Se especifica la posición angular para cada uno de los servos (Base Giratoria, hombro, codo, muñeca y pinza) con sus respectivas funciones *Servo Write Angle* y deslizadores respectivamente (figura 3.24).

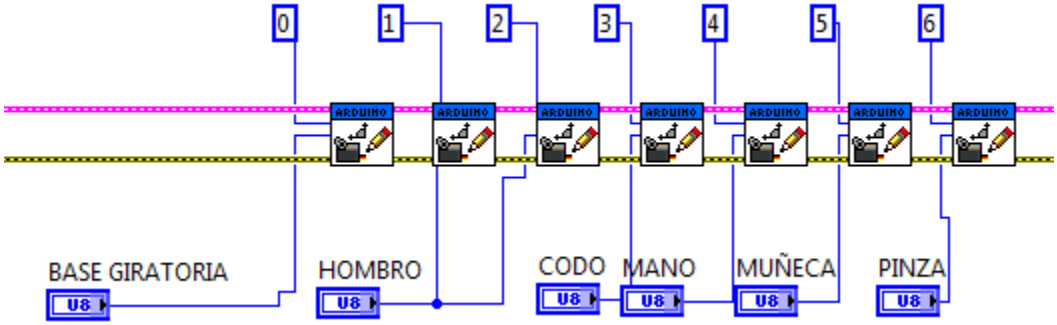


Figura 3.24. Diagrama de bloques para el control modo manual.
Fuente: Investigador

Gracias a los deslizadores donde se especifica el ángulo de cada servo en el panel frontal, se puede controlar cada movimiento manual del brazo robótico.

3.13. Control automático del brazo robótico

Para que el movimiento del brazo sea automático, dentro de la función *Case* la condición debe ser falsa. Dentro del proceso, se almacenan los datos de la terminación de una iteración. LabView transfiere los datos conectados dentro del bucle a la próxima iteración.

Después de que el ciclo se ejecuta, la terminal de iteraciones del ciclo regresa al último valor almacenado. Dichos datos vienen dados por una matriz.

Para poder determinar la posición angular de cada uno de los servomotores se establece una matriz de datos dados por la función *spreadsheet string to array*, la cual transforma una cadena de valores en una matriz mediante un indicador dado por la misma función, en este caso se usa la expresión “%f” como se muestra en la figura 3.25.

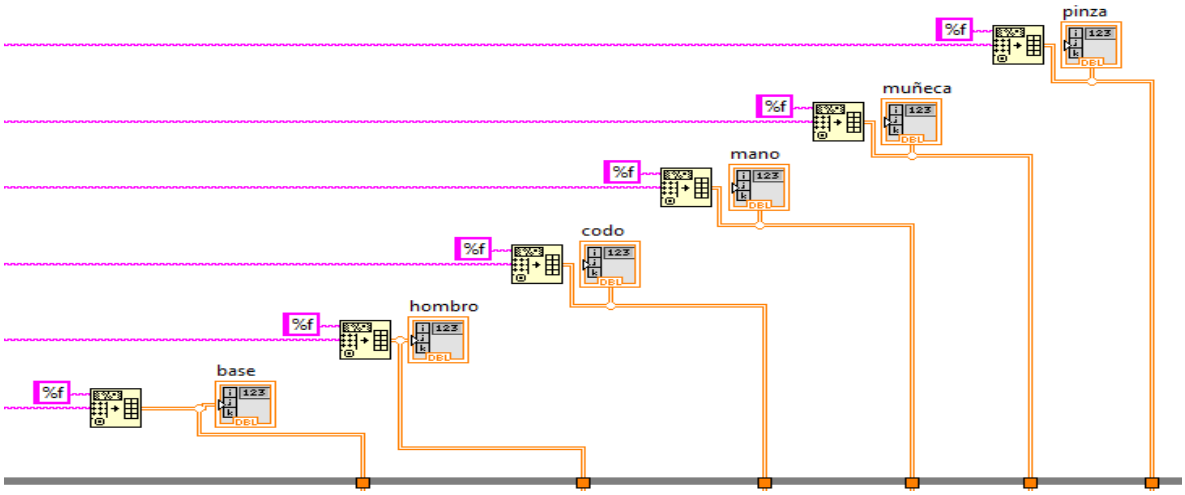


Figura 3.25. Matrices para la posición angular de los servomotores
Fuente: Investigador

Dentro de cada matriz se debe realizar un posicionamiento del ángulo de cada servo para conocer el intervalo en el cual se pueden mover, primeramente se debe utilizar sliders para mover a cada servo y poder conocer estos valores, luego se crea una secuencia de movimientos en conjunto con los valores que ya se conoce, por esa razón es que algunos valores se repiten, incrementan o disminuyen siendo los números que corresponden al movimiento automático.

Ahora cada valor que atraviese por la terminal de iteraciones será seleccionado previamente, y dichos valores serán ejecutados secuencialmente por un tiempo de 10 ms mediante la función *flat sequence structure* (figura 3.26).

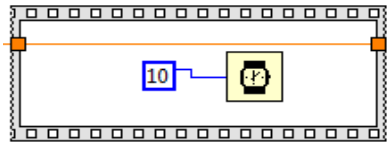


Figura 3.26. Función “flat sequence structure”
Fuente: Investigador

Los valores seleccionados atraviesan una serie de comparadores y selectores como se muestra en la figura 3.27, debido a que solamente se necesitan los datos que determinen las posiciones angulares que cada uno de los servomotores toman para su determinado movimiento secuencial.

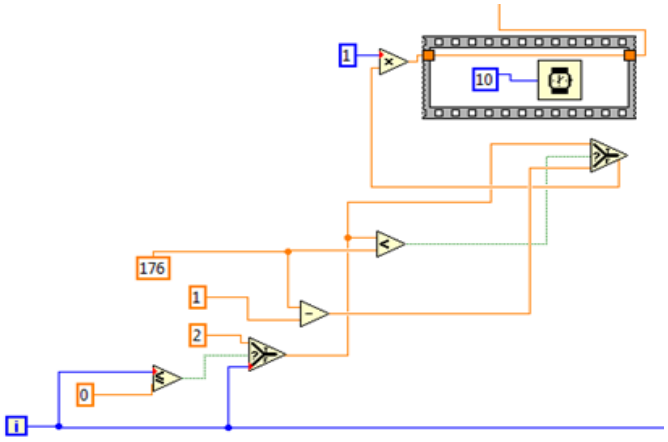


Figura 3.27. Selección de valores de la matriz
Fuente: Investigador

Finalmente se utiliza la función *index array* para poder devolver el valor seleccionado de la matriz en un indicador para el servomotor, es decir, que a ese valor el servomotor lo tome como su posición angular (figura 3.28).

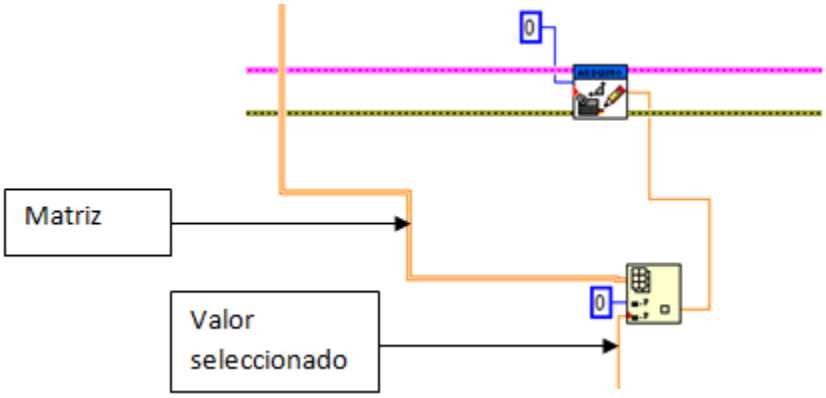


Figura 3.28. Entrada de la posición angular seleccionada al servomotor
Fuente: Investigador

Cada servomotor recibe sus determinados valores, por lo que la conexión de los selectores, comparadores y la función *flat sequence structure* debe hacérselo de forma individual, sin embargo es la misma conexión para cada uno. De esta forma el valor se selecciona cada 10 ms y se lo transfiere al servomotor correspondiente, de este modo el manipulador actúa de forma autómata repitiendo el proceso hasta que la condición del While Loop deje de ser falsa.

3.14. SIMULACIÓN

A continuación en la figura 3.29 se presentan las simulaciones realizadas dentro de la interfaz LabView utilizando medidores que determinan el ángulo y posición de cada servomotor.

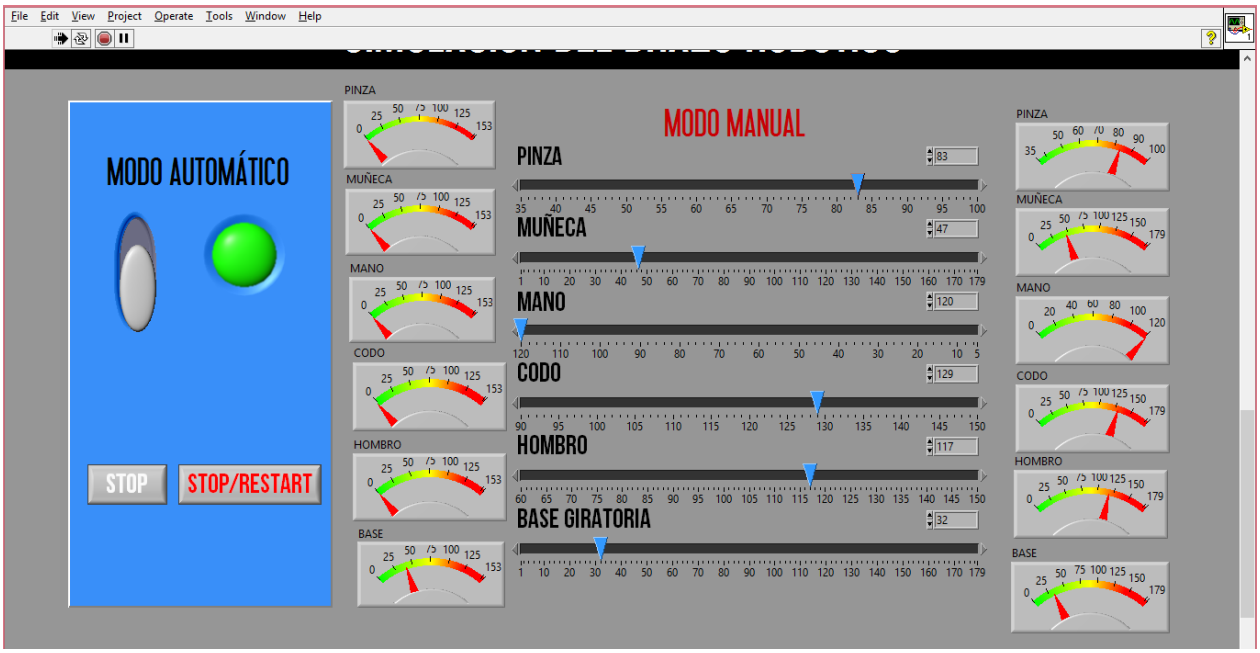


Figura 3.29. Simulación del brazo robótico.

Fuente: Investigador

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- Con el programa Solid Works, software utilizado para realizar diseños en Cad, se pudo obtener las dimensiones y medidas del brazo robótico utilizadas para el diseño de la estructura del brazo robótico.
- La interfaz Labview permitió gracias a la facilidad de programación gráfica por diagrama de bloques y por la compatibilidad que tiene para trabajar conjuntamente con la plataforma Arduino realizar el diseño del brazo robótico.
- Con el diseño del sistema human machine interface (HMI) y la descarga de los software VI Package Manager e Instrumento de Arquitectura de Software Virtual (VISA), se pudo obtener el diseño del control manual y automático del brazo robótico.
- La fuente de poder DC externa debe ser exclusiva para la alimentación de los servomotores, debido a que Arduino no da la suficiente corriente para el control de los mismos.

Recomendaciones

- El proceso automático se puede mejorar ampliando el número de datos de la matriz haciendo que el brazo realice el número de iteraciones que el usuario desee.
- El diseño del brazo robótico se puede mejorar, especificando tareas adicionales como la ejecución de nuevos movimientos controlados manualmente o en si la programación de un nuevo control autómata dependiendo de los requerimientos del usuario.
- El material con el que se podría elaborar la estructura mecánica del brazo robótico puede ser una variante de plástico más flexible de lo normal que pueda permanecer a la intemperie durante mucho tiempo sin sufrir daños en su estructura y coloración, mucho más resistente que el vidrio, lo cual no es fácil que se rompa y de esta manera se evita el riesgos de lesiones.

BIBLIOGRAFÍA

Abadía, J. A. (1997). Energía y Computación. En J. A. Abadía, *Energía y Computación.*

ANGULO USATEGUI, J. 2. (2000). Robótica práctica Tecnología y Aplicaciones. En J. ANGULO USATEGUI. Madrid: Paraninfo.

BARRIENTOS A., PEÑÍN L., BALAGUER C., RAFAEL ARACIL. (1997). Fundamentos de Robótica. En P. L. BARRIENTOS A., *Fundamentos de Robótica: Mc Graw Hill.*

Botscience. (2013). *Botscience*. Recuperado el Diciembre de 2014, de http://botscience.net/store/index.php?route=product/product&product_id=83

Bueno, A. (16 de Noviembre de 2009). Recuperado el 10 de Enero de 2015, de http://www.portaleso.com/usuarios/Toni/web_robot_3/robot_indice.html#indice

Labelec. (2009). *Labelec*. Recuperado el Diciembre de 2014, de http://www.info-ab.uclm.es/labelec/solar/electronica/elementos/servomotor.htm

ni. (13 de Agosto de 2012). *National Instruments*. Recuperado el 20 de Diciembre de 2014, de http://www.ni.com/white-paper/7564/en/

OCEANO. (1995). *Diccionario Enciclopédico Oceano.* Madrid.

OLLERO B., A. (2001). *ROBÓTICA, Manipuladores y Robots Móviles.* Barcelona: Marcombo Boixareu Editores.

Rojas, D. (17 de 03 de 2012). *Scibd.* Obtenido de Scibd: http://es.scribd.com/doc/85749234/Interfaz-Hombre-Maquina-HMI#scribd

Sánchez A. (2010). Tutorial Labview. En Sánchez A, *Tutorial Labview* (pág. 10). ITST.

Servodatabase. (2009). *Servodatabase.com*. Recuperado el Diciembre de 2014, de http://www.servodatabase.com/servo/hitec/hs-311

Sin autor. (2008 - 2014). *Ayuda electrónica*. Obtenido de Ayuda electrónica: http://ayudaelectronica.com/%C2%BFque-son-los-arduino-shields/

SPONG, M. (1989). Robot dinamics and control. En M. SPONG, *Robot dinamics and control.* USA: Ed. John Wiley.

Torrente, Ó. (2010). Arduino. En Ó. Torrente, *Arduino: Curso Práctico de Formación.* AlfaOmega.

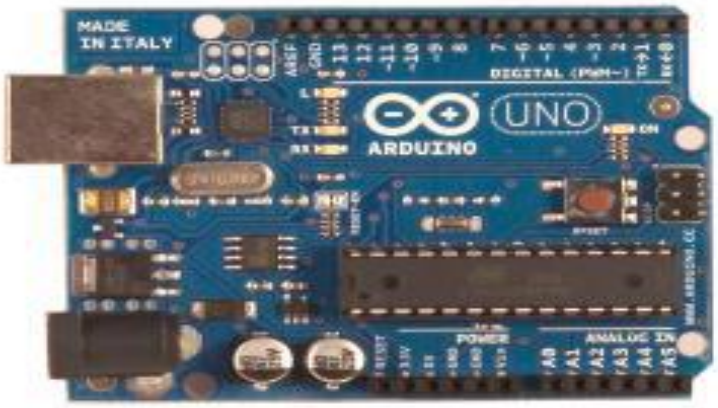
Tutorial LabView.pdf. (s.f.). Obtenido de Tutorial LabView.pdf: http://www.esi2.us.es/~asun/LCPC06/TutorialLabview.pdf

webelectro. (16 de Enero de 2015). *webelectro*. Recuperado el 02 de Febrero de 2015, de http://www.webelectro.cl/producto/arduino-sensor-shield-v5-0-expansor

ANEXOS

DATASHEET ARDUINO UNO Y ARDUINO SHIELD

Arduino UNO



CE

Product Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Index

RS radiospares RADIONICS

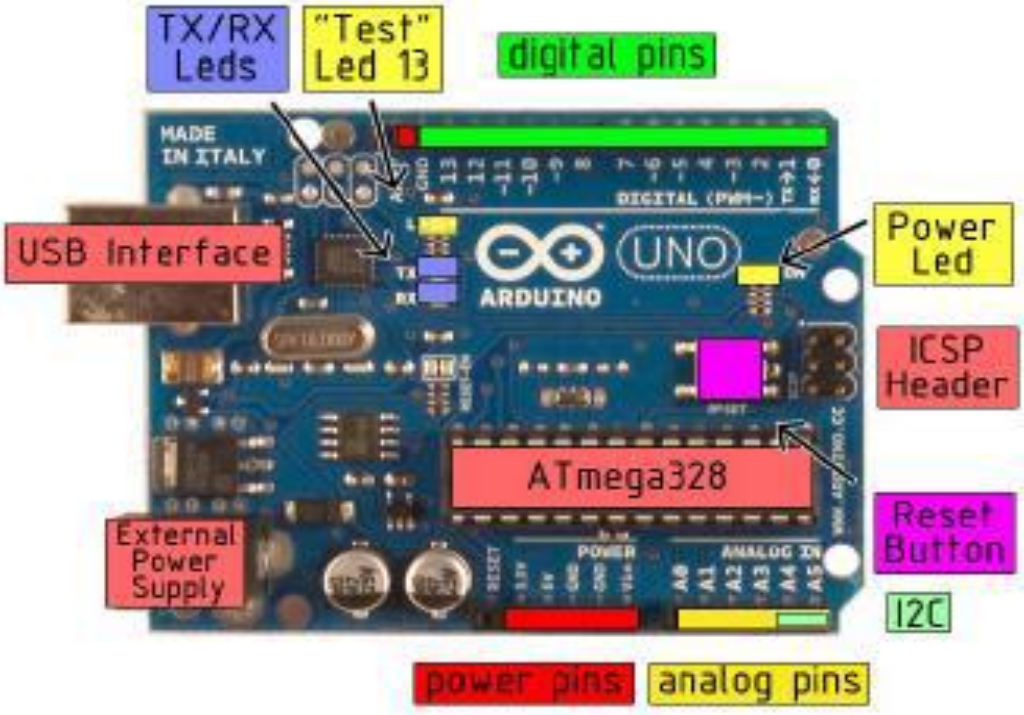
Technical Specification

EAGLE files: arduino-duemilanove-uno-design.zip Schematic: arduino-uno-schematic.pdf

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

the board



Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. TThese pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- I²C: **4 (SDA) and 5 (SCL)**. Support I²C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

- **AREF**. Reference voltage for the analog inputs. Used with analogReference().
- **Reset**. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and Atmega328 ports.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an ".inf file is required..

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega328 datasheet.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno w/ ATmega328" from the **Tools** > **Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader).

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

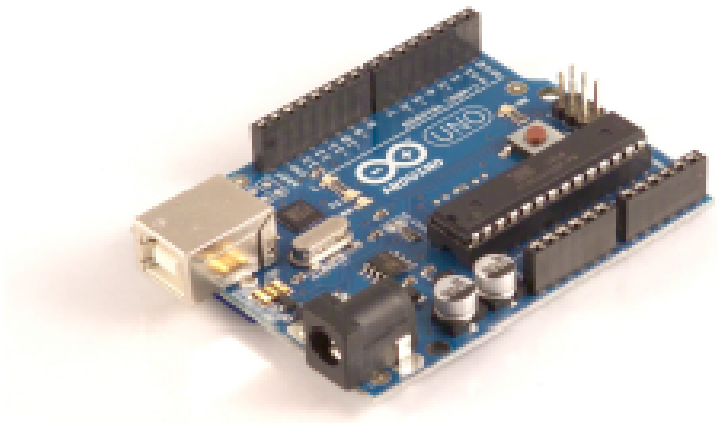
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.



How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

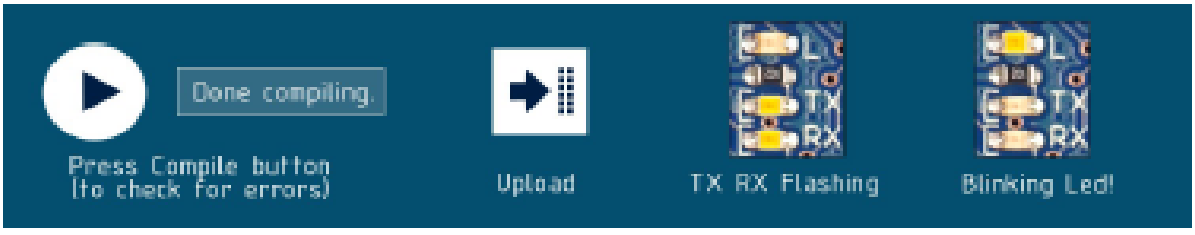
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

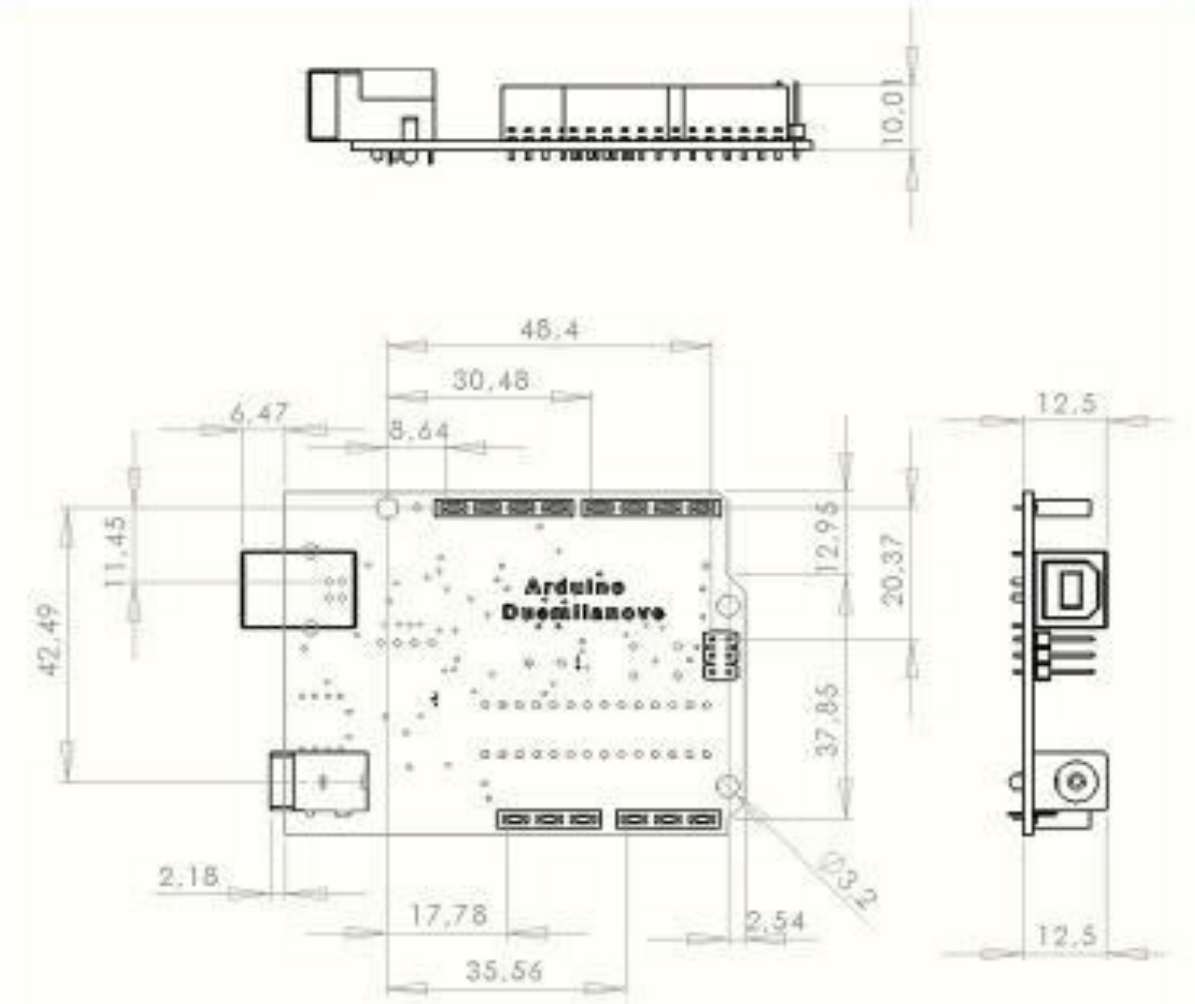
Once you have your skecth you'll see something very close to the screenshot on the right.

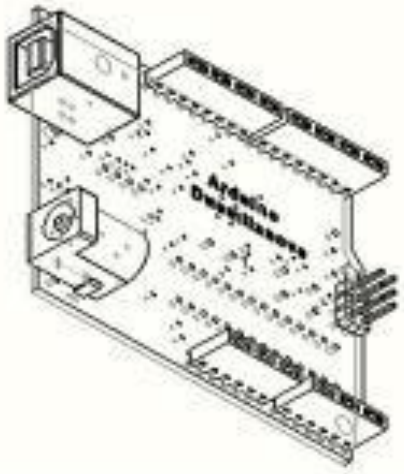
In **Tools>Board** select

Now you have to go to
Tools>SerialPort
and select the right serial port, the one arduino is attached to.



Dimensioned Drawing





radiospares RADIONICS

Terms & Conditions

1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, or for any products that have been altered or modified in any way by a Customer. Moreover, The producer shall not be liable for any defects that result from Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producer shall have a reasonable time to replacements. Replaced products shall be warranted for a new full warranty period.

1.3 EXCEPT AS SET FORTH ABOVE, PRODUCTS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." THE PRODUCER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

1.4 Customer agrees that prior to using any systems that include the producer products, Customer will test such systems and the functionality of the products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above, and no additional obligations or liabilities shall arise from the producer providing such services.

1.5 The Arduino™ products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™ products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™ products which is solely at the Customer's risk, and that Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

1.6 Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products and any use of Arduino™ products in Customer's applications, notwithstanding any applications-related information or support that may be provided by the producer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilities and expenses it incurs to the extent directly caused by: (i) an actual breach by a Customer of the representation and warranties made under this terms and conditions or (ii) the gross negligence or willful misconduct by the Customer.

3. Consequential Damages Waiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

4. Changes to specifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

Enviromental Policies

The producer of Arduino™ has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.

DATASHEET DEL ARDUINO SHIELD

Arduino I/O Expansion Shield

(SKU: DFR0014)

Introduction

The Arduino I/O Expansion Shield provides an easy way to connect sensors, servos and RS485 device to Arduino board. It expands Arduino's Digital I/O and Analog Input Pins with Power and GND. It also provides separate PWM Pins which are compatible with standard servo connector. Another unique feature is that the I/O shield has a build-in RS485 converter which allows Arduino communicating with RS485 devices. The communication socket provides an extremely easy way to plug a wireless module such as APC220 RF module and DF-Bluetooth module. It has an individual power input for Servos. A servo power jumper allows user to select using external power or internal power to drive the Servos.

Diagram

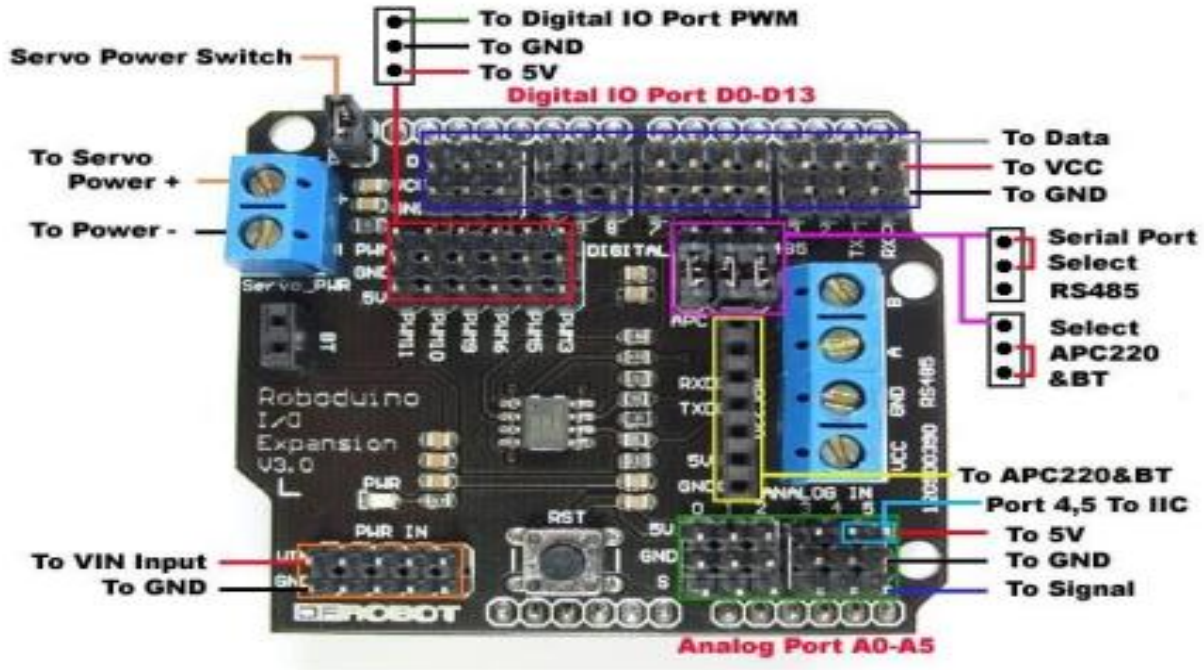
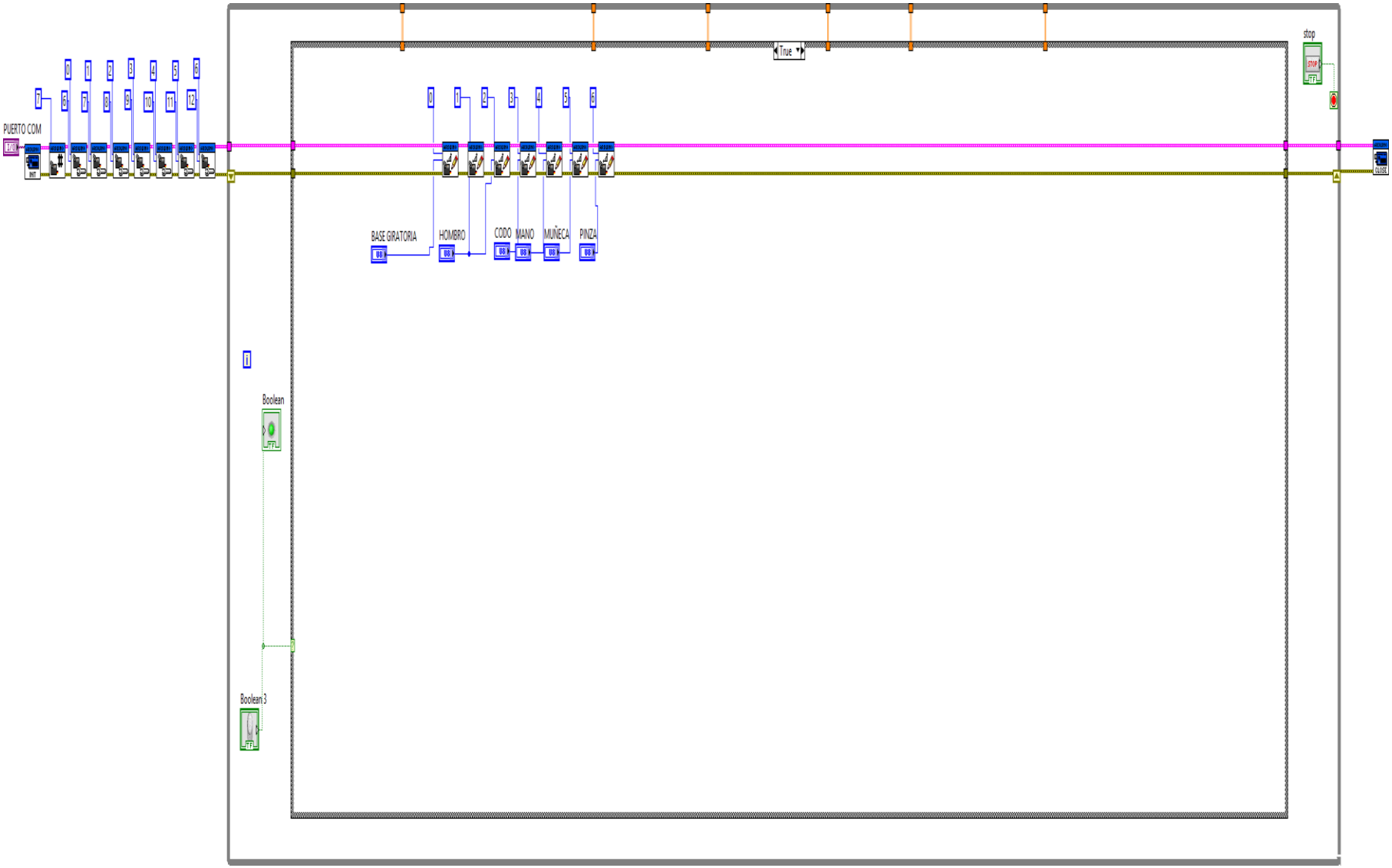
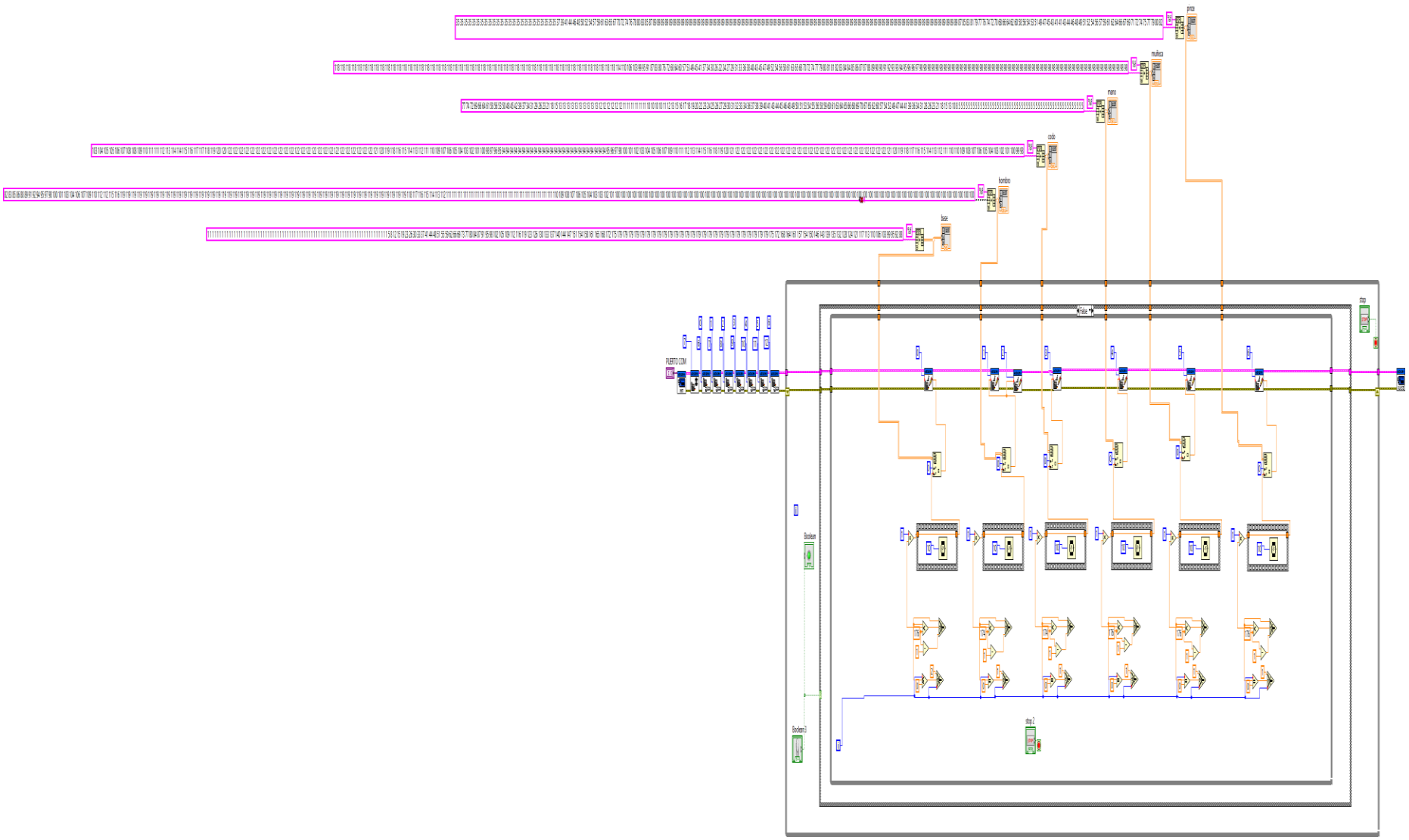


Figure 1 Arduino I/O Expansion Shield

DISEÑO DEL DIAGRAMA DE BLOQUES EN LA INTERFAZ GRÁFICA PARA CONTROL MANUAL



DISEÑO DEL DIAGRAMA DE BLOQUES EN LA INTERFAZ GRÁFICA PARA CONTROL AUTOM´TICO



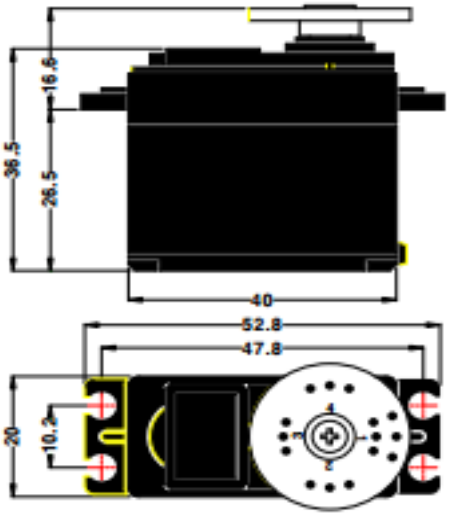
DATASHEET DE SERVOMOTORES

PREPARED BY JUN HEE, LEE
UPDATE:APR 01, 2002

ANNOUNCED SPECIFICATION OF HS-311 STANDARD SERVO

1.TECHNICAL VALUE

CONTROL SYSTEM	:+PULSE WIDTH CONTROL 1500usec NEUTRAL	
OPERATING VOLTAGE RANGE	:4.8V TO 6.0V	
TEST VOLTAGE	:AT 4.8V	AT 6.0V
OPERATING SPEED	:0.19sec/60□ AT NO LOAD	0.15sec/60□ AT NO LOAD
STALL TORQUE	:3.0kg.cm(42oz.in)	3.5kg.cm(48.60oz.in)
IDLE CURRENT	:7.4mA AT STOPPED	7.7mA AT STOPPED
RUNNING CURRENT	:160mA/60□ AT NO LOAD	180mA/60□ AT NO LOAD
STALL CURRENT	:700mA	800mA
DEAD BAND WIDTH	:5usec	5usec
OPERATING TRAVEL	:40□/ONE SIDE PULSE TRAVELING 400usec	
DIRECTION	:CLOCK WISE/PULSE TRAVELING 1500 TO 1900usec	
MOTOR TYPE	:CORED METAL BRUSH	
POTENTIOMETER TYPE	:4 SLIDER/DIRECT DRIVE	
AMPLIFIER TYPE	:ANALOG CONTROLLER & TRANSISTOR DRIVER	
DIMENSIONS	:40x20x36.5mm(1.57x0.78x1.43in)	
WEIGHT	:43g(1.51oz)	
BALL BEARING	:TOP/RESIN BUSHING	
GEAR MATERIAL	:RESIN	
HORN GEAR SPLINE	:24 SEGMENTS/□5.76	
SPLINED HORNS	:SUPER/R-XA	
CONNECTOR WIRE LENGTH	:300mm(11.81in)	
CONNECTOR WIRE STRAND COUNTER	:40EA	
CONNECTOR WIRE GAUGE		



2.FEATURES

LONG LIFE POTENTIOMETER, TOP RESIN BUSHING

3.APPLICATIONS

AIRCRAFT 20-40 SIZE,STEERING AND THROTTLE SERVO FOR CARS, TRUCK AND BOATS

4.ACCESSORY & OPTION

CASE SET/	GEAR SET/	HORN SET/
HS322T:1EA	HS322G1:1EA	R-XA:1EA
HS322M:1EA	HS322G2:1EA	
HS322L:1EA	HS322G3:1EA	
PH/T-2 2x30 NI:4EA	HS322G4:1EA	
	HS300RB:1EA	

HITEC RCD KOREA INC.