UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

RASPBERRY PI 3 Y QT CREATOR COMO HERRAMIENTA PARA EL CONTROL Y MONITOREO DE DISPOSITIVOS DE ADQUISICIÓN DE DATOS

TESINA

Que para obtener el título de:

INGENIERO EN COMPUTACIÓN

En modalidad de:

Desarrollo de un caso práctico

P R E S E N T A:

EDUARDO LEONEL BOLAÑOS CAMARGO

ASESOR:

M. en C. Jesús Hernández Cabrera

Ciudad Nezahualcóyotl, Estado de México, 2018

Agradecimientos y dedicatoria:

A mis padres Violeta C. y Daniel B., que me motivan día con día a seguir creciendo como persona. Quienes soportaron todos mis arranques de desesperación y lograron calmarme cada vez que pensaba en renunciar. Mi fuerza y felicidad. Gracias por brindarme momentos incomparables en la vida. ¡Los amo!

A mi hermana "la Gorda", Daniela C., la persona que tanto amo y adoro, mi inspiración para seguir adelante, quien me mantuvo por años, cuido de mí y me animaba en los momentos más difíciles que he pasado hasta ahora. Siempre has estado conmigo y sabes que siempre voy a estar para ti hermanita. ¡Te amo! Y gracias por soportarme.

A mi primo, mi hermano Sergio. Con quien he pasado momentos sumamente divertidos, con quien compartí helado de una corcholata. Gracias por cada recuerdo.

A mis tías, Sandra y Elsa, que me cuidaron y educaron cuanto pudieron. Les debo mucho.

A mi familia en general, primos, tíos. Que confiaron y creyeron en mí.

Para el Maestro Marcelo P. Mi guía en esta carrera, en la facultad y quien me brindo de conocimiento más allá de lo que se puede obtener en las clases. Sin usted no hubiera tenido esta gran oportunidad, así como muchas otras. Gracias por dejarme pertenecer a esta gran familia (LAB COM).

Para el Maestro Jesús H. Mi asesor de tesis y gran amigo, quien me invito a participar en el proyecto que presento. Gracias a usted conocí otra rama de la ingeniería en la cual me encantaría especializarme más (domótica). Gracias por estar ahí para solucionar mis dudas y acompañarme en este largo proceso.

Gracias al Doctor Edgar Morales Palafox, no sólo por dedicarle tiempo a revisar mi trabajo sin ser revisor. Por confiar en mi para demás proyectos. Sabe que cuenta conmigo no sólo como su alumno también como su amigo.

A mis revisores Hugo P., Miguel Ángel S., y Juan Carlos C., por tomarse el tiempo de revisar mi trabajo y hacerme las observaciones apropiadas. Gracias por sus horas de clase, donde además de brindarme su conocimiento me brindaron su amistad.

A mis amigos de más de 10 años (Alucard, Kallito, Marcos, etc), mi familia por selección gracias por estar conmigo aun en los peores momentos. Siguen ustedes en titularse.

A mi segunda familia por selección, Ángel B., Dayane H., Ramón G., Eliot S., por hacer más divertida la carrera, por todas las pláticas y siempre apoyarnos en las materias. De igual forma a mi familia de Lab Com, por todos los grandes momentos.

Para terminar este apartado, gracias a la persona de mi vida, a quien más extraño en el universo, Leonel Camargo D., mi abuelito. Desearía que estuvieras aquí conmigo. Daria todo por volver a verte, aunque sea 1 minuto más. Sé que pasará, y te contaré todo lo que he logrado.

Índice

1) Introducción 5

2) Antecedentes del desarrollo de sistemas de software para interconectar con kits de desarrollo de electrónica 7

2.1 Historia de la computación 7

2.1.1 Generaciones 9

2.2 Electrónica 11

2.2.1 Electricidad 12

2.2.2 Circuitos Eléctricos 14

2.2.3 Componentes Eléctricos 15

2.2.4 Compuertas Lógicas 20

2.2.5 Circuitos Integrados 27

2.3 Inicios de los Sistemas de Software 28

2.3.1 Primeras ideas del software 28

2.3.2 Primeros programas 33

2.4 Arquitectura de Computadoras 34

2.4.1 Arquitectura de von Neumann 34

2.4.2 Arquitectura de Harvard 35

2.4.3 Arquitecturas manejadas por conjunto de instrucciones 37

2.5 Microcontroladores y Microprocesadores 37

3) Integración Digital 46

3.1 Convergencia digital 47

3.2 Kits de electrónica actuales 48

3.2.1 Arduino 49

3.2.2 Galileo 54

3.2.3 Raspberry Pi 56

3.3 Domótica 59

3.3.1 X-10 59

3.3.2 KNX 61

3.4 Robótica 63

4) Definición del problema. 69

5) Propuesta de solución. 75

5.1 Elementos a usar en Hardware. 75

5.1.1 Tablas de registros internos de los actuadores Dynamixel. 80

5.1.2 Sensores utilizados. 87

5.1.3 Ventajas en el uso de Raspberry. 89

5.2 Elementos a usar en Software. 89
5.2.1 Instalación de los componentes a usar. 91
5.2.2 QT CREATOR durante el desarrollo. 106
5.2.3 Ejecución. 107
6) Resultados y trabajo futuro. 119
7) Conclusiones 120
8) Referencia de imágenes y tablas. 122
9) Bibliografía. 127

1) Introducción

Actualmente nos encontramos rodeados por dispositivos electrónicos; a cada lugar donde se mire existe un componente electrónico diseñado para facilitar o simplificar las tareas que tenemos los seres humanos*, pero no siempre fue de esta forma. Hace seis décadas el ser humano no tenía idea de lo que era un dispositivo digital, mucho menos llego a creer que algún tiempo en el futuro tendrían la capacidad de tener todo conectado en un solo dispositivo; desde controlar la iluminación de una habitación hasta realizar pagos desde la comodidad de su hogar. Continuamente se busca optimizar todo aquello con lo que el ser humano tiene contacto, tanto en la vida laboral como en aspectos del hogar, para ello se han implementado sin fin de sistemas tanto de software** como de hardware***, o sistemas embebidos (los cuales a diferencia de una computadora son diseñados para un uso específico).

Las empresas diariamente buscan innovar creando nuevos dispositivos ("GADGETS"), tomemos el ejemplo de Dynamixel creador de los actuadores/servomotores más utilizados en la robótica, y nuevos entornos de desarrollo ("IDE"), como lo es LabView aportado por la empresa National Instruments, esto para competir en el mercado. Pero muchos de estos dispositivos llegan a costar demasiado dinero e incluso las licencias de los entornos de desarrollo sobrepasan constantemente el presupuesto considerado para el diseño de un proyecto.

Inicialmente la problemática por la cual surge este trabajo gira en torno a los costos, ya que la tecnología empleada en el primer diseño requería de una gran inversión, debido a estas circunstancias nos vimos obligados a investigar alternativas más económicas que cumplieran los requerimientos ya planteados desde el primer diseño.

Este primer diseño consta de controlar y monitorear tanto el movimiento como el estado de un brazo robótico. Cumpliendo las expectativas, el primer diseño fue completamente satisfactorio, pero a un gran costo. Analizando las diferentes opciones se concluyó que lo más conveniente es el uso de un SBC (Single Board Computer) o computadora de una sóla placa, posteriormente en este trabajo se explicara el motivo por el cual se llegó a esta conclusión. La placa seleccionada es Raspberry Pi 3 modelo B. A lo largo de este documento se describen los beneficios y por qué se optó por utilizar este tipo de tecnología; también se hablará sobre el entorno de desarrollo implementado (QT Creator) y por ende el lenguaje que dicho entorno nos permite ocupar. Ambas tecnologías en conjunto conforman una gran herramienta para la adquisición de datos.

Se tendrá presente la historia y el cómo se manejaban antiguamente los kits de desarrollo, cómo se logró llegar hasta los dispositivos actuales, que son los microcontroladores y microprocesadores, cómo surgieron y sus usos actualmente.

*: Ver "Pére,C., Valdez,D., García, V., & Trotter, S. (2010). *¿Cómo ha cambiado el avance tecnológico de los últimos años nuestro modo de vida?*. febrero 28, 2018, de Mediosfera Reflexiones acerca de los medios y la sociedad Sitio web: https://mediosfera.wordpress.com/2010/04/17/%C2%BFcomo-ha-cambiado-el-avance-tecnologico-de-los-ultimos-anos-nuestro-modo-de-vida/"

**: Programa de computadora compuesto de rutinas y órdenes que indican a esta que debe hacer.

***: Componentes físicos de una computadora, por ejemplo: monitor, mouse, teclado, etc

Para lograr entender los microcontroladores y microprocesadores, se hablará sobre las arquitecturas ocupadas para la creación de estos y para lograr entender la arquitectura de los mismos reviviremos algunos años del pasado, conociendo como era la vida antes de los sistemas digitales y como poco a poco la civilización fue dándose cuenta de los alcances que tienen las máquinas de cálculos; hablaremos sobre la historia de la computación, el nacimiento de las primeras máquinas, como fueron evolucionado hasta llegar a las computadoras actuales. Los inicios del software y quienes fueron los precursores.

Se tocará el término “Digital Integration” o bien Integración Digital, presentando los elementos que lo conforman y cómo se encuentra presente en la vida cotidiana, su relación con el término Convergencia Digital y sus aplicaciones con los kits de electrónica actuales dándole vida a campos como la Domótica y la Robótica, veremos cómo surgieron poco a poco estos dos términos y su evolución hasta llegar a la actualidad.

Como se puede notar, el proyecto no solo consta de redactar las partes de software, también incursionaremos en el apartado de hardware, ya que en la carrera de Ingeniería en Computación impartida en la Facultad de Estudios Superiores Aragón; a pesar de que se llevan materias referentes a la electrónica, no se tiene un conocimiento bien estructurado sobre este tema, por lo tanto decidí formar parte de este proyecto para obtener un mayor aprendizaje y brindarlo en forma de tesina a quienes deseen aprender un poco más sobre la interacción de ambas disciplinas.

2) Antecedentes del desarrollo de sistemas de software para interconectar con kits de desarrollo de electrónica

En este apartado se presentarán una breve reseña histórica acerca del software, su evolución con el pasar del tiempo y como a su ritmo se fue integrando con los kit de desarrollo de electrónica.

Las primeras preguntas que se resuelven son: ¿Qué es un sistema de software? y ¿Qué es un kit de desarrollo de electrónica? Para ello, se hace un recuento en la historia de la computación, dando un panorama muy general de los inicios del cómputo, haciendo un pequeño énfasis en aquellos precursores que nos entregaron lo que ahora todos conocemos mundialmente como computadora. Explicaré a grandes rasgos las generaciones de las computadoras, así como las arquitecturas computacionales.

El software es un elemento crucial al momento de hablar de cómputo, es por ello que se describe un poco sobre los inicios de este gran recurso, para que surgió y como fue empleado en las primeras computadoras.

Para contestar la incógnita de los kits de desarrollo de electrónica comenzaremos recordando lo que es la electrónica, cómo surgió y cómo fue progresando hasta tener los circuitos integrados que ahora todos conocemos y los cuales son esenciales para la tecnología actual.

Todo esto con el fin de retroceder varias décadas al lector para que este pueda conocer el origen del cómputo moderno.

2.1 Historia de la computación

Un punto importante es entender el concepto de Computadora. Una computadora puede ser definida como un dispositivo electrónico, diseñado para aceptar datos de entrada y realizar operaciones sobre ellos, para elaborar resultados que se puedan obtener como salidas. Teniendo esto en cuenta podemos definir que el término cómputo hace referencia al hecho de realizar cálculos.

Desde hace millones de años las personas han sido capaces de realizar cálculos, matemática simple que poco a poco se ha ido haciendo más compleja, se han empleado cientos de artefactos para poder realizar estas cuentas o cálculos. Algunos autores dictan la posibilidad que la primera máquina aplicada para la solución de problemas aritméticos haya sido el ábaco.

Comenzando con los precursores conoceremos a Jhon Napier el cual introdujo el concepto de logaritmo, gracias a esto la tarea de multiplicar se simplificó. A partir de ese momento, se comenzaron a construir las máquinas de cálculo llamadas analógicas o máquinas de medida.

Para 1823 Charles Babbage comenzó a desarrollar la primera máquina digital con ayuda del gobierno británico. Incorporó una rutina de operaciones en tarjetas perforadas, que representó un gran paso para su próxima máquina.

Tuvo la idea de una calculadora digital universal en el año de 1834, a la que llamó Máquina Analítica. Esta máquina no se pudo construir sino hasta un siglo después.

Con la invención del relevador, Konrad Zuse, logro construir sus máquinas. Anteriores a los bulbos, Zuse diseño la primera computadora electromecánica programable, la Z1 opacada por la tan conocida ENIAC.

Aunque los trabajos con las máquinas analógicas continuaban en desarrollo, sucedió un evento de gran impacto en el campo digital cuando Herman Hollerith invento la técnica para lograr procesar gran cantidad de datos por medio de tarjetas perforadas.

Aproximadamente desde 1930 hasta 1970 la tarjeta perforada fue la encargada del procesamiento de datos y remplazo mayormente al procesamiento manual.

A continuación se muestra un diagrama (**Imagen 1.1**) donde podemos visualizar como se utilizaba la tarjeta perforada dando soporte para instrucciones y datos, se representa el bloque procesador y el apartado de memoria que recibe las instrucciones del programa y los datos perforados en lotes de tarjetas.

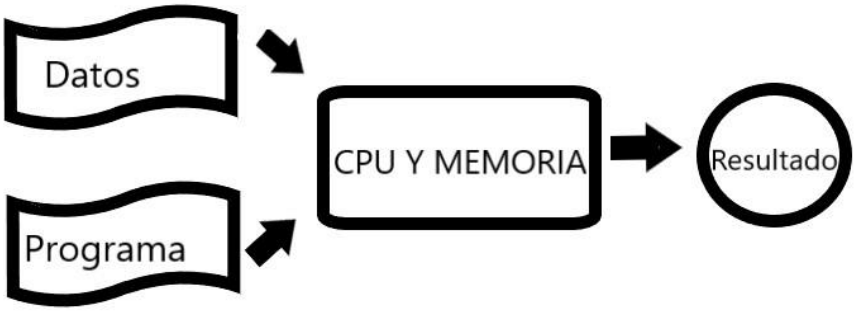


Imagen 1.1 Diagrama del uso de tarjetas perforadas. [A1]

En este modelo podemos observar que el lote de datos se encuentra separado al lote de instrucción, esto se lograba con la ayuda de “tarjetas de control” las cuales marcaban el comienzo y el fin de las tarjetas de instrucciones así como el comienzo y el fin de las tarjetas de datos. Las instrucciones y datos eran almacenados en la memoria para que posteriormente el apartado encargado del control o sistema operativo ordene la ejecución. Para terminar, los resultados obtenidos se imprimían en papel.

La llamada computadora secuencial automática de IBM fue puesta en operación hacia el año de 1944 en la Universidad de Harvard. Dicha máquina constaba de partes electromecánicas provistas por IBM y estaba controlada por una cinta de papel perforada similar a la tarjeta perforada.

Después de presentar este modelo de máquina se diseñaron otras similares, pero fue durante la segunda Guerra Mundial donde se creó lo que muchos autores señalan como la primera computadora nombrada ENIAC (Electronic Numerical Integrator And Calculator), en la cual el cambio de sus programas se hacía mediante el recableado de unas borneras, las cuales son un tipo de conector eléctrico en el que un cable se engancha contra una pieza metálica haciendo uso de un tornillo.

Corría el año de 1945 cuando John von Neumann consiguió participar en el diseño una máquina llamada EDVAC (Electronic Discrete Variable Automatic Computer).

Esta máquina a diferencia de sus predecesoras no fue diseñada para un uso específico o bien para una aplicación concreta, sino que se trató de una máquina de propósito general, esta es capaz de almacenar instrucciones y datos en una memoria, con ese gran avance se pudo sustituir el conexionado fijo entre los componentes físicos por un programa de instrucciones intercambiable.

La llamada máquina de von Neumann se fundamenta en cuatro principios que en la actualidad aún se aplican:

- Trabaja con la información codificada en binario (1,0 – encendido, apagado).
- Programa almacenado en memoria.
- Unidad funcional de memoria de programa.
- Contar con la habilidad de realizar una interrupción de secuencia de instrucciones en un programa.

Fue hasta el 18 de junio de 1958 que en la Universidad Nacional Autónoma de México llegó la primera máquina electrónica de cálculo; una IBM 650 usada. “Inicialmente era manejada con tarjetas perforadas posteriormente se agregaron controladores de cinta, una de las razones del éxito en el mercado de esta máquina es su compatibilidad con máquinas IBM previas.” (Jones, 2013). Provenía de la universidad de California, con ella se inauguró el primer centro de cómputo en México.

Una de las tareas realizadas por las computadoras fuera de las universidades era ayudar a realizar censos de población en Estados Unidos. Muchas de las labores que se le encargaban a las computadoras solían ser tan complejas que se dividían en fragmentos más simples, cada uno de ellos se resolvía con un programa de computo diferente. Al conjunto de todos ellos y la manera en que armonizaban se le dio el nombre de sistema.

Teniendo una pequeña idea de los inicios del cómputo y los orígenes de las primeras computadoras, hablaremos de forma más general sobre éstas. Se dividieron en 5 generaciones de computadoras.

2.1.1 Generaciones

1° Generación:

En esta generación las computadoras estaban constituidas por válvulas de vacío, de las cuales emanaba una gran cantidad de calor, además que eran máquinas muy grandes y trabajan de forma secuencial. Por lo tanto:

1. El programa se cargaba en memoria principal gracias a una sección del sistema operativo llamado “cargador”, dicho programa se encontraba en tarjetas o cintas perforadas.
2. El programa era ejecutado instrucción por instrucción

Al final, se imprimía el resultado.

2ª Generación:

Estas se encontraban constituidas por transistores y aplicaban el uso de circuitos impresos, lo cual implico una reducción en tamaño a comparación de las computadoras de 1ª generación. Facilitaron el trabajo simultáneo entre un cálculo y alguna operación de E/S*. Esta idea llevada a la práctica dio muy pocos resultados, ya que existió una desproporción entre la velocidad de cálculo interno y las velocidades de E/S, lo cual implico que la CPU no se utilizara más que en un pequeño porcentaje de tiempo. La solución a este problema fue que las operaciones de E/S se realizaran ocupando como soporte de almacenamiento unidades de cinta magnética, que eran mucho más rápidas a comparación de las lectoras de tarjetas e impresoras.

A continuación, se muestra un diagrama con el funcionamiento de estas máquinas.

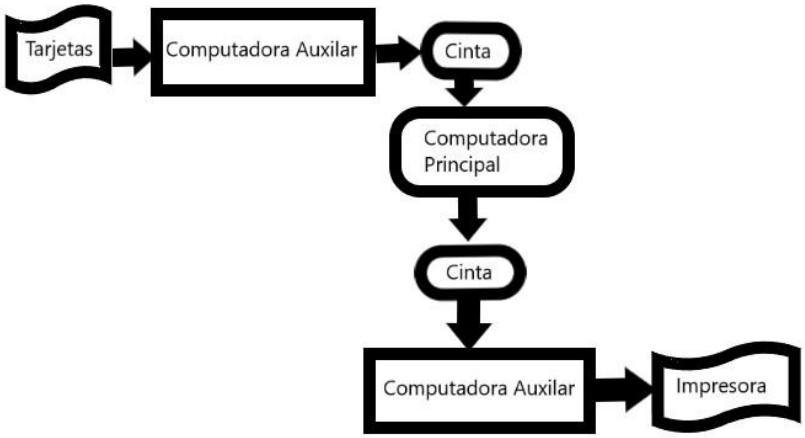


Imagen 1.2 Diagrama a bloques del funcionamiento de una computadora de segunda generación. [A2]

A este tipo de procesamiento se le conoce con el nombre de Procesamiento por Lotes. En esta modalidad, cada lote estaba compuesto por varios trabajos y era necesario esperar que el lote cargado en la cinta magnética se procesara por completo para obtener los resultados de cada trabajo.

3ª Generación:

A inicios de 1964 comenzó la 3ª generación de computadoras las cuales ya implementaban circuitos integrados del tipo SSI (Small Scale Integration) y MSI (Medium Scale Integration) aumentando la velocidad interna de la computadora y reduciendo considerablemente la energía que utilizaban.

Gracias a todos los elementos nuevos incorporados se pudo aplicar la multiprogramación, la cual consiste en que varios programas residan de forma simultánea en la memoria en estado de ejecución. Solo un programa puede hacer uso de la CPU, pero los otros pueden efectuar operaciones de E/S simultáneamente.

*: Entrada y Salida.

En la Tercera generación la "Carga por Lotes" fue sustituida por la "Carga Continua", de esta manera los trabajos se ponen en cola de espera en un disco magnético y el sistema operativo es el encargado de ejecutarlos según su nivel de prioridad.

4ª Generación **(1971-1987)**:

Las computadoras de esta generación poseían el uso particular de circuitos LSI o Large Scale Integration, lo cual permitió incluir una CPU completa en un solo dispositivo, denominado microprocesador. Aquí el procesamiento se realizaba en mayor medida en tiempo real.

Físicamente son muy parecidos a las computadoras personales con las que se cuenta actualmente. En esta generación tocamos el término de microprocesador, que nos hace pensar en la Ley de Moore. Dictada por Gordon Moore, indica: Cada tres años, "la potencia de los ordenadores se multiplica por cuatro". (Tubella & Vilaseca, 2005).

5ª Generación:

Fue durante la época de los 80's cuando se llevó a cabo una revolución en el diseño de una computadora, pues comenzó el uso de arquitecturas tipo RISC (de la cual hablaremos más adelante), superescalabilidad, niveles de caché, etc.

En esta generación podemos encontrar a las conocidas supercomputadoras, las cuales desarrollan funciones inteligentes basadas en experiencias de Inteligencia Artificial.

Se puede decir que el procesador es el corazón de este tipo de máquinas, el lenguaje de este núcleo es de inferencia lógica, el cual es un proceso limitado del razonamiento, es decir, se basa en un hecho anterior y lo toma como fundamento para llegar a una conclusión final.

El objetivo principal del programa de 5ª generación supone que las computadoras deberán tener capacidades cognitivas superiores a las humanas.

2.2 Electrónica

Este capítulo no es tratado como "La historia de la Electrónica", ya que como tal la Electrónica tiene sus inicios desde el descubrimiento de la Electricidad.

La electrónica es descrita como la práctica que se posee para poder manejar, obtener y logar utilizar información a través de dispositivos, los cuales están encargados del movimiento de las cargas eléctricas.

También se considera a la electrónica como una herramienta de cálculo, cada uno de los elementos que conforman un circuito eléctrico posee sus propias ecuaciones, por lo tanto, la información de salida es el resultado del paso de la señal a través de todos los componentes pertenecientes al circuito.

Hasta este punto estamos hablando de dispositivos, cargas eléctricas, señales y circuitos, pero aun no abordaremos esos temas. Continuaremos hablando de señales eléctricas, y para ello debemos definir que es electricidad.

2.2.1 Electricidad

La electricidad es aquella fuerza invisible capaz de realizar trabajo, como puede ser, producir movimiento, calor, luz, entre otros.

El elemento fundamental de la electricidad es una partícula llamada electrón, la cual se encuentra en las orbitas del átomo, a su vez este forma a las moléculas y estas últimas son quienes conforman a la materia, por lo tanto podemos decir que todo cuanto nos rodea obedece principios eléctricos. Absolutamente todo sistema eléctrico se basa en el flujo controlado de electrones.

La corriente eléctrica se provoca al generarse un movimiento ordenado en los electrones dirigiéndose en algún sentido. Podemos visualizarlo de la siguiente manera:

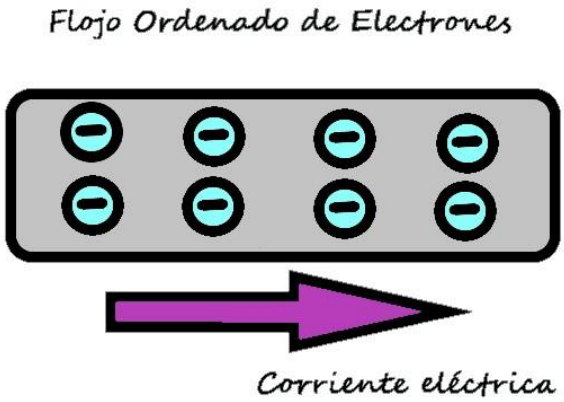


Imagen 1.3 Flujo ordenado de electrones. [A3]

Ahora bien, existe una fuerza la cual está encargada de obligar a los electrones a generar corriente eléctrica, y es conocida como "Fuerza Electromotriz" (FEM), esta posee su propia unidad de medida, el Voltio, *"Un voltio es la diferencia de potencial eléctrica que existe entre dos puntos de un hilo conductor que transporta una intensidad constante de 1 amperio cuando la potencia disipada entre estos dos puntos es igual a 1 vatio."* (Chacón, 2008).

Otra forma de definir a la FEM es: aquella fuerza capaz de producir tensión eléctrica en un circuito cerrado o una diferencia de potencial entre dos puntos de un circuito abierto.

Aquello que genera esta fuerza puede ser una pila, una batería o una celda solar, entre otros.

Ya que comenzamos a hablar sobre unidades de medida, conoceremos una nueva, el Amperio, definido como: *"La intensidad de la corriente que pasa por un conductor cuya resistencia es de un ohmio, cuando la fuerza electromotriz ó diferencia de potencial en sus extremos vale un voltio"*. (Nottoli, 2004).

El amperio es la unidad de medida resultante del cálculo de la intensidad de corriente, este nuevo concepto (intensidad de corriente), hace referencia a la cantidad de carga eléctrica que circula por un conductor en un tiempo determinado.

Matemáticamente representado como:

$$I = \frac{CANTIDAD\ DE\ CARGA}{TIEMPO} = \frac{q}{t}$$

Donde:

q: Esta dado en Coulomb

t: Esta dado en segundos.

Aquí vemos otra nueva unidad de medida, el Coulomb, el cual es la unidad de carga eléctrica que está definido aproximadamente como:

$$1\ Coulombio\ \approx 6.28 \times 10^{18}\ electrones$$

A partir de esto, definimos que:

$$Carga\ del\ protón\ (+e) = 1.602\ \times 10^{-19}$$

$$Carga\ del\ electrón\ (-e) = 1.602\ \times 10^{-19}$$

Volviendo al Amperio conseguimos la siguiente ecuación:

$$I = \frac{1C}{1s} = 1\ Amperio$$

Con esto podemos concluir que:

1 Ampere es el paso de 6.28 x 10^{18} electrones en 1 segundo.

En electrónica se ocupa mayormente el Miliamperio, el cual tiene una equivalencia de 1 x 10^{-3} Amperios, pero también se puede llegar a trabajar con Amperios o incluso con Kiloamperios que equivalen a 1 x 10^{3} Amperios

Continuando con las unidades de medida y la matemática base de lo que es la Electrónica, hablaremos de Tensión Eléctrica.

Como se mencionó con anterioridad la Fuerza Electromotriz es la encargada de generar Tensión Eléctrica, que a su vez es quien da lugar a que los electrones se muevan ordenadamente a través de un conductor, originando Corriente Eléctrica y tiene como unidad de medida el Voltio.

Son muchos términos en la electricidad que se deben conocer para pasar a la electrónica. Pero estos nos brindan los principios básicos.

El Voltio es otra de las unidades de medida más conocidas en el mundo, cuando compramos una pila o una batería de carro, siempre verificamos el voltaje, pilas de 1.5 volts o baterías de 12 volts, una batería y una pila se diferencian por los materiales que contienen en su interior para generar la FEM y a su vez la tensión eléctrica.

Con la tensión eléctrica damos paso a dos términos más, potencial eléctrico y diferencia de potencial.

El potencial eléctrico en un punto es definido como el trabajo necesario para trasladar una carga eléctrica desde el infinito hasta dicho punto.

La diferencia de potencial habla de dos puntos, y es el trabajo necesario para que la carga se traslade de uno punto a otro.

Ambos conceptos se miden en voltios y su ecuación es la siguiente:

$$1\,V = \frac{1\,Julio}{1\,Coulombio}$$

En la ecuación anterior se muestra 1 Julio que es utilizado para medir energía, trabajo y calor.

Utilizando como ejemplo una pila de 1.5 volts podemos decir que proporciona una tensión de 1.5 volts o que entre sus dos terminales (+,-) produce una diferencia de potencial de 1.5 volts.

Esto es lo principal de la electricidad, lo que se debe conocer antes de pasar a la Electrónica. A lo largo de este apartado hemos hablado de circuitos, al menos esporádicamente, pero ¿qué se debe de considerar un circuito?, por lo menos en electrónica.

2.2.2 Circuitos Eléctricos

Los circuitos eléctricos son el producto de combinar diferentes componentes de manera que se emplee el paso de la corriente eléctrica para generar una respuesta de salida a partir de su entrada, la o las ecuaciones de cada componente influye en la manipulación de la información que corre entre las entradas y la salida o salidas. Para hacer efectivo el uso del circuito debe existir el generador de corriente como puede ser una pila, pasar por elementos que modifiquen la información enviada desde el generador, ser captada por un receptor y retornar al otro punto del generador. En caso de existir un corte en este flujo, se interrumpiría la circulación de la corriente y por lo tanto el dispositivo receptor dejaría de funcionar.

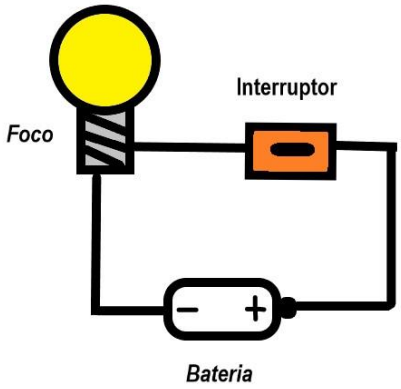


Imagen 1.4 Circuito eléctrico. [A4]

Los elementos principales de un circuito son:

- Generador.
- Líneas conductoras.
- Dispositivos de control.
- Receptor

Aunque cada circuito puede variar, estos elementos son los componentes de un circuito práctico, así que comencemos con el generador.

El generador es el encargado de producir la energía eléctrica, de proveer la fuerza impulsora a los electrones, de generar la tensión eléctrica (Volts) que a su vez da lugar a la intensidad eléctrica (Ampere) a través del circuito, puede ser una pila.

Las líneas conductoras son el medio por el cual pasa la corriente eléctrica, por lo general son cables de cobre, estos varían dependiendo de la intensidad de corriente que se maneje, pueden ser de un diámetro muy reducido hasta cables industriales, ya que ahí se manejan corrientes muy altas.

El dispositivo de control, es aquel que permite o impide el paso de la corriente, por ejemplo un interruptor, este puede hacer que la corriente continúe su paso por el circuito o simplemente detener el paso de los electrones generando una intensidad de 0 y un voltaje de 0, haciendo que el receptor deje de "funcionar".

El receptor es el elemento o dispositivo eléctrico que recibe la corriente eléctrica para realizar alguna función, como puede ser producir calor o movimiento (foco o motor). Este dispositivo es la razón del generar un circuito eléctrico.

En electrónica se involucran diferentes elementos que dan vida a un circuito eléctrico, los electrones o bien la corriente eléctrica se vuelve información que es modificada, alterada, manipulada, o como se le quiera expresar, por los diferentes componentes que lo conforman, entre ellos aparece la mencionada resistencia, pero también podemos tener, inductores, capacitores, diodos, entre otros.

2.2.3 Componentes Eléctricos

Enfoquémonos ahora en aquello que, en lo personal, capta mi atención altamente, los componentes eléctricos. Aquí tenemos presentes dos tipos, los elementos activos y los elementos pasivos.

Se clasifican como elementos activos a los dispositivos que son capaces de cambiar una forma de energía en otra, rectificar o amplificar, entre ellos podemos mencionar:

Transistor (transfer resistor): Este dispositivo es un semiconductor de estado sólido y es el encargado de entregar una señal de salida con base a una señal de entrada, posiblemente es uno de los más utilizados en la electrónica, además lo encontramos en un aparato que utilizamos todos los días, la computadora.

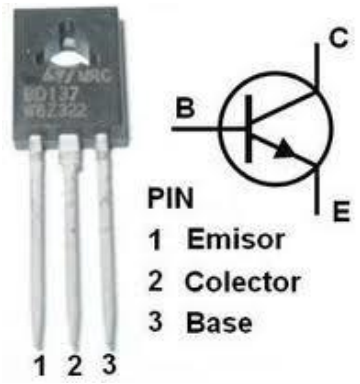


Imagen 1.5 Transistor y su representación gráfica. [B1]

Diodo: Al igual que el transistor es un semiconductor rectificador, pues este componente sólo permite el paso de la corriente en una sola dirección. Consta de dos terminales conocidas como Ánodo (A) y Cátodo (K). El Diodo más conocido es el Diodo Led o diodo emisor de luz, utiliza el mismo principio que un diodo común, con la diferencia de que el Led convierte la energía que en el circula en fotones para así producir la luz que lo caracteriza.

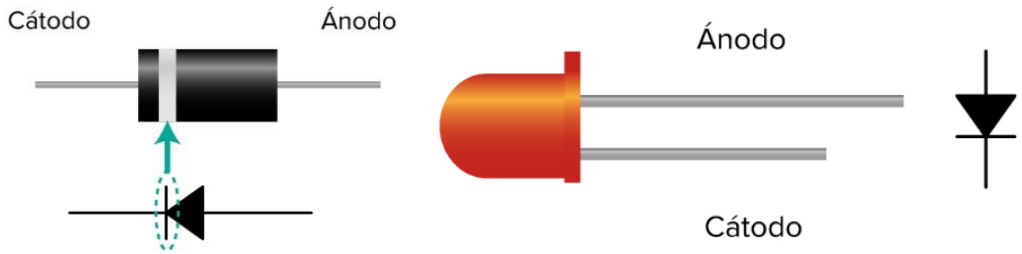


Imagen 1.6 Diodo común y diodo Led con su representación gráfica. [B2.1] [B2.2]

Compuerta Lógica: De este dispositivo podemos hablar mucho, pero solo trataré lo fundamental. Su base es la Lógica Booleana (verdadero y falso, 1 y 0, alto y bajo). Es un circuito de conmutación integrado en un chip, usualmente está compuesta internamente por interruptores electromagnéticos o relés.

Estos son solo algunos ejemplos de elementos activos. La resistencia, el capacitor y el inductor son ejemplos de elementos pasivos ya que son capaces de almacenar o disipar la energía eléctrica.

Resistencia: Este componente limita la cantidad de corriente o divide el voltaje, suelen tener un valor establecido de fábrica fácil de identificar por su código de colores (eso solo en algunas resistencias, las más comunes) pero también existen las resistencias variables y un tipo más de resistencia, conocido como resistencia de alambre bobinado.

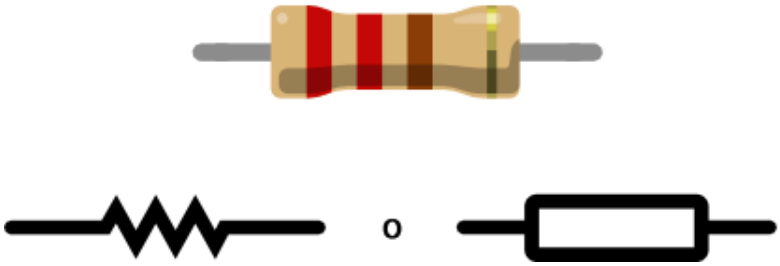


Imagen 1.7 Resistencia y su representación gráfica. [B3]

Capacitor: También es conocido como Condensador, su trabajo es almacenar energía eléctrica, está diseñado a partir de dos placas conductoras separadas por un aislador (dieléctrico), funciona concentrando el campo eléctrico generado por la diferencia de potencial aplicada al dieléctrico. Este dieléctrico puede descargar la energía almacenada cuando la fuente de carga se remplaza por una trayectoria conductora.

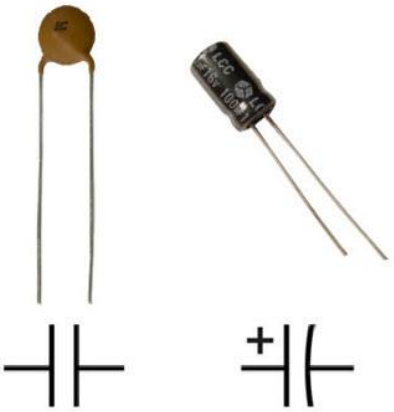


Imagen 1.8 Capacitor cerámico y capacitor común con su representación gráfica. [B4]

Inductor: Este es técnicamente una bobina de alambre, su función es concentrar el campo magnético de una corriente eléctrica, además si una corriente junto con su campo magnético cambia de dirección o de valor, es capaz de generar un voltaje inducido.

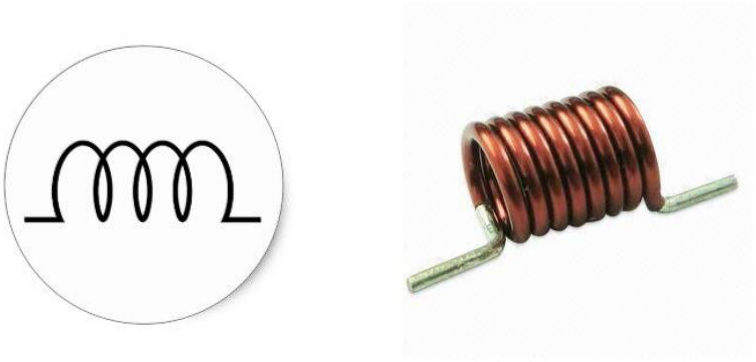


Imagen 1.9 Inductancia y su representación gráfica. [B5.1][B5.2]

En la Electrónica Digital se utiliza la llamada lógica de Boole, lo cual es un tipo de matemática bivalente, que quiere decir eso, pues que se caracteriza únicamente por tener dos estados

(como ya se había mencionado), alto y bajo o 0 y 1, para lo que resta de este apartado lo manejaremos como 0 y 1 ya que estos conforman el sistema binario que es la forma en que trabajan las computadoras.

Bit, Byte, Nibble y palabra, son expresiones muy ocupadas en cómputo, pero en la electrónica también son conocidas y requeridas, como tal los dígitos 0 y 1 reciben el nombre de bit, al conjunto de 4 bits se le llama nibble, 8 bits conforman un byte y a los conjuntos de 0's y 1's se le conoce como palabra.

$$1\ y\ 0 = bit$$

$$1010 = nibble$$

$$10101100 = byte$$

$$1010110010101100 = Palabra\ de\ 16\ bits$$

Utilizando los bits podemos realizar operaciones algebraicas (Aritmética Binaria), lo cual es un tipo de matemática muy peculiar, veremos el ejemplo de tres casos, la negación, la adición y la multiplicación.

Negación o inversa:

$$1! = 0$$
$$0! = 1$$

Simplemente es tomar el valor y utilizar su contrario.

Adición:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = \ 1$$

$$1 + 1 = (10)_2$$

El 10 que se muestra, está en base 2 o sea, es un diez binario, veremos un ejemplo con una operación sencilla:

$$\begin{array}{r} 1010 \\ 1111 \\ \hline 11001 \end{array} +$$

Como podemos ver la adición de números binarios es semejante a la adición con números decimales se ira explicando paso por paso esta operación.

1- De derecha a izquierda (como cualquier operación algebraica se realiza).
2- 0 + 1 = 1 y se coloca.
3- 1 + 1 = 0 y se acarrea 1.
4- 1 + 0 = 1 + 1= 0 y se acarrea 1.
5- 1 + 1 = 0 y se acarrea 1 0 + 1 = 1.
6- El 1 que se acarreaba se coloca pues no hay más números para realizar operaciones.

Y con esto obtenemos 11001, esto se puede comprobar convirtiendo los números binarios a decimales.

Para la comprobación veremos cómo convertir los números binarios en enteros, utilizando el ejemplo anterior.

Para la conversión seguiremos los siguientes pasos:

I- Tomaremos el número a convertir 1010.

II- Dividiremos en posiciones 0 primera posición, 1 segunda posición, 0 tercera posición y 1 cuarta posición. Esto contando de derecha a izquierda.

III- En binario la primera posición equivale a $2^0 = 1$, la segunda posición equivale a $2^1=2$, la tercera posición:

$$1\ 0\ 1\ 0$$
$$2^3 2^2 2^1 2^0$$

IV- Ahora multiplicamos cada valor por su respectiva posición

$$0\ x\ 2^0 = 0\ x\ 1 = 0$$
$$1\ x\ 2^1 = 1\ x\ 2 = 2$$
$$0\ x\ 2^2 = 0\ x\ 4 = 0$$
$$1\ x\ 2^3 = 1\ x\ 8 = 8$$

V- Obteniendo todos los resultados, sumamos.

$$0 + 2 + 0 + 8 = 10$$

VI- Por lo tanto concluimos que 1010 en binario es el número 10 en base decimal.

Lo mismo debe realizarse los mismos pasos para transformar el otro número a decimal obteniendo así: $(1111)_2 = (15)_{10}$.

Si realizamos la suma de 10 + 15 = 25, solo falta convertir el resultante a binario, para eso seguiremos los siguientes pasos.

1- Dividir el número decimal entre 2 (hasta concluir con los número enteros) y solo utilizaremos lo que sobra.

$$\frac{25}{2} = 12\ y\ sobra\ 1$$

2- Ahora tomamos el resultado y ese será nuestro nuevo dividendo y seguiremos dividiendo entre 2 hasta que el resultado sea 0.

$$\frac{12}{2} = 6\ y\ sobra\ 0$$
$$\frac{6}{2} = 3\ y\ sobra\ 0$$
$$\frac{3}{2} = 1\ y\ sobra\ 1$$
$$\frac{1}{2} = 0\ y\ sobra\ 1$$

3- Hasta este punto es cuando terminamos, pues el resultado es 0 y el sobrante es 1.

4- En este punto solo basta tomar los valores sobrantes que fuimos obteniendo a lo largo de las divisiones del último al primero obteniendo en este ejemplo: 11001.

De esta forma validamos que haya sido cierto lo que obtuvimos previamente.

Es a veces tedioso y complicado, sobre todo cuando se trata de equivalencias muy grandes, para ello ocupamos calculadoras programables o científicas.

Multiplicación:

$$0\ x\ 0 = 0$$
$$0\ x\ 1 = 0$$
$$1\ x\ 0 = 0$$
$$1\ x\ 1 = 1$$

Los resultados son parecidos a la multiplicación normal de números decimales, sólo que no debemos olvidar que nos encontramos trabajando en base 2.

Veamos un ejemplo:

$$\begin{array}{r} 10 \\ 11 \quad * \\ \hline 10 \\ 10 \quad + \\ \hline 110 \end{array}$$

Realizando la conversión a decimal nos queda como 2 * 3 = 6.

Y convirtiendo 6 a binario nos da como resultado: 110.

Haciendo uso de estas tres operaciones podemos obtener operaciones más complejas, es muy común que la ALU (La Unidad de Aritmética y Lógica es la sección de la unidad central en donde se ejecutan todos los cálculos y se hacen todas las comparaciones)* de una computadora solo contenga la capacidad de realizar adición, pero es capaz de realizar las demás operaciones gracias al demás hardware.

Ya que conocemos un poco más sobre el álgebra binaria, hablaremos sobre las compuertas lógicas, estos elementos de la electrónica son dispositivos que emplean tablas de verdad para dar una salida, es decir, dependen de la entrada dada para entregar un resultado, estas compuertas lógicas son fundamentales en los PLC's, o Controladores Lógicos Programables, instrumentos de suma importancia en la industria.

2.2.4 Compuertas Lógicas

Bien, en el mundo existen 7 tipos de compuertas lógicas; AND, OR, NOT, NAND, NOR, XOR y XNOR, cada una de ella tiene su propia tabla de verdad como se muestra a continuación:

AND:

ENTRADA 1	ENTRADA 2	SALIDA
0	0	0
0	1	0
1	0	0
1	1	1

Lo cual en un circuito puede ser representado con dos elementos de control, un interruptor:

*: (Inter-American Institute for Cooperation on Agriculture, 1985)

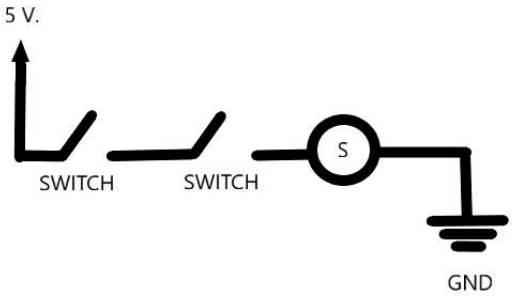


Imagen 1.10 Representación en circuito de una compuerta AND. [A5]

Los 0's en la tabla de verdad representan un interruptor abierto, y los 1's representan el interruptor cerrado, por lo tanto debemos tener ambos interruptores cerrados (en 1) para recibir una respuesta satisfactoria, de esta forma en el circuito el flujo de la corriente eléctrica puede continuar hasta llegar a nuestro dispositivo receptor, en este caso un foco o led emisor de luz.

Y su símbolo en electrónica es:

Imagen 1.11 Símbolo de la compuerta AND. [B6]

OR:

ENTRADA 1	ENTRADA 2	SALIDA
0	0	0
0	1	1
1	0	1
1	1	1

Al igual que la compuerta AND, se puede representar con dos interruptores.

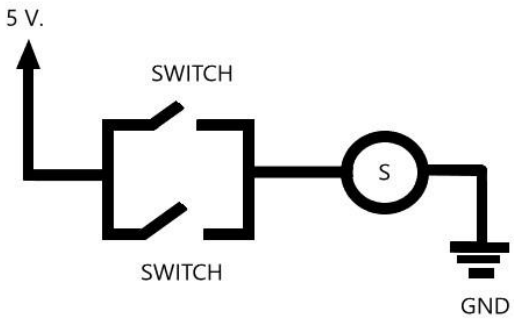


Imagen 1.12 Representación en circuito de una compuerta OR. [A6]

A diferencia de la compuerta AND el arreglo de la compuerta OR consta de 2 interruptores en paralelo, ya que como su tabla de verdad y el mismo nombre de la compuerta lo dicen, no hace falta que ambos interruptores se encentren en 1 para lograr excitar el dispositivo receptor, con tener una sola de las compuertas activas se puede tener el flujo de la corriente sin ningún problema.

Y su símbolo en electrónica es:

Imagen 1.13 Símbolo de la compuerta OR. [B7]

NOT:

Este tipo de compuerta es la más simple, pues se trata únicamente de la inversa de la entrada, en este caso su tabla de verdad es:

ENTRADA	SALIDA
0	1
1	0

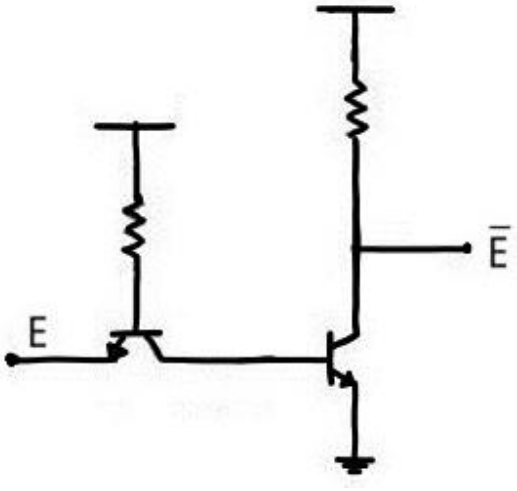


Imagen 1.14 Representación en circuito de una compuerta NOT. [A7]

Únicamente es invertir la entrada. Su símbolo en electrónica es:

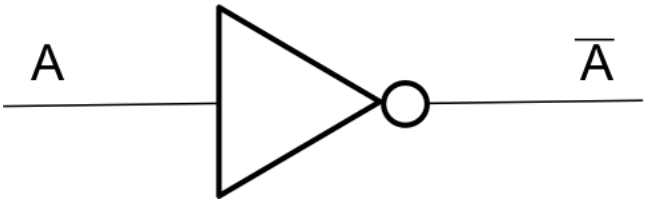


Imagen 1.15 Símbolo de la compuerta NOT. [B8]

NAND:

Esta compuerta consta de una compuerta AND y de una negación, por lo tanto la respuesta obtenida como resultado de la compuerta AND se negara, entregando su inverso, y su tabla de verdad es la siguiente.

ENTRADA 1	ENTRADA 2	SALIDA NORMAL AND	SALIDA NAND
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

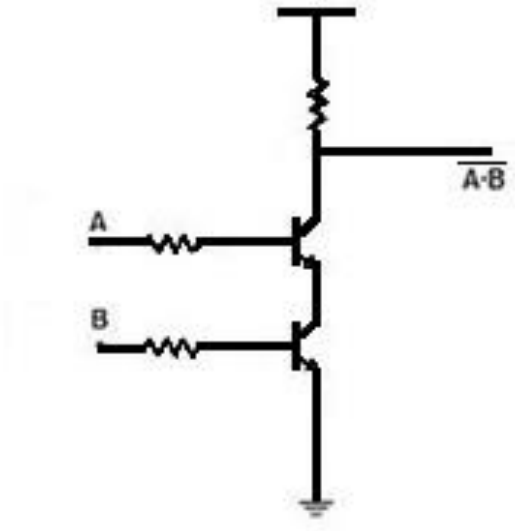


Imagen 1.16 Representación en circuito de una compuerta NAND. [A8]

Esta compuerta es representada con el símbolo:

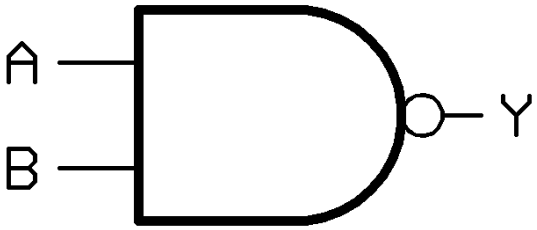


Imagen 1.17 Símbolo de la compuerta NAND. [B9]

NOR:

Al igual que la compuerta NAND es basada en una compuerta AND, la compuerta NOR está basada en una compuerta OR, solo que su salida natural es negada, tal cual se muestra en su tabla de verdad:

ENTRADA 1	ENTRADA 2	SALIDA NORMAL OR	SALIDA NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

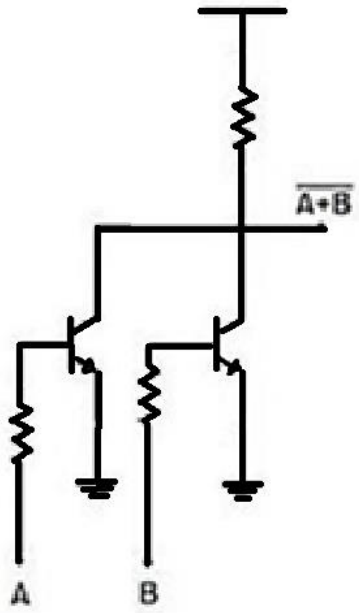


Imagen 1.18 Representación en circuito de una compuerta NOR. [A9]

Esta compuerta es representada con el símbolo:

Imagen 1.19 Símbolo de la compuerta NOR. [B10]

XOR:

Esta compuerta es conocida también como OR exclusiva, es ocupada para el armado de un semisumador, su tabla de verdad es la siguiente:

ENTRADA 1	ENTRADA 2	SALIDA NORMAL OR	SALIDA XOR
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

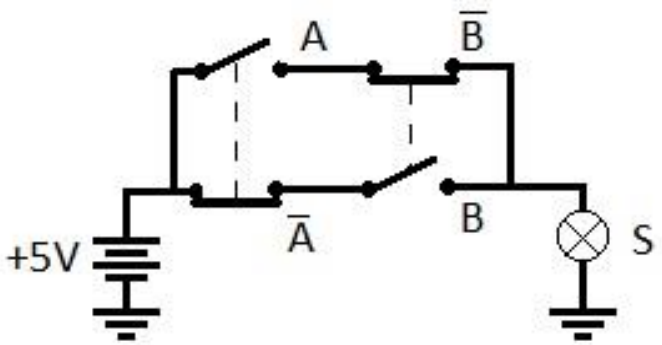


Imagen 1.20 Representación en circuito de una compuerta XOR. [B11]

Como puede notarse, solo se tiene corriente cuando ambas entradas se encuentran en posiciones diferentes (1,0 o 0,1).

Esta compuerta es representada con el símbolo:

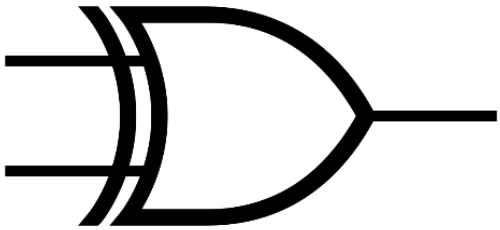


Imagen 1.21 Símbolo de la compuerta XOR. [B12]

XNOR:

Esta compuerta es la negación a la compuerta XOR.

ENTRADA 1	ENTRADA 2	SALIDA NORMAL OR	SALIDA XOR	SALIDA XNOR
0	0	0	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	0	1

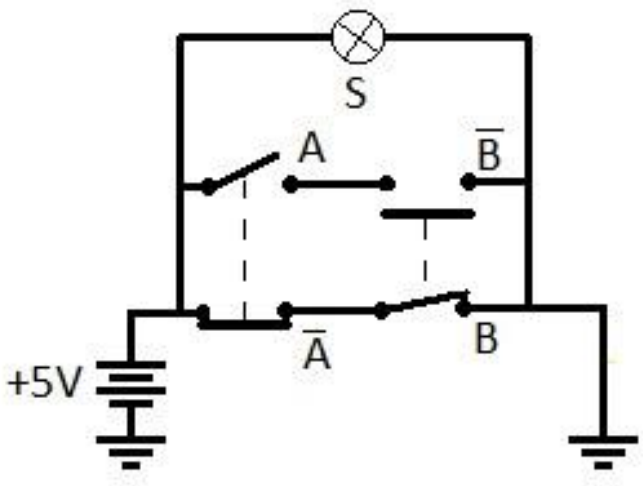


Imagen 1.22 Representación en circuito de una compuerta XNOR. [B13]

En este caso el paso de la corriente solo se dará si ambas entradas están en la misma posición.

Esta compuerta es representada con el símbolo:

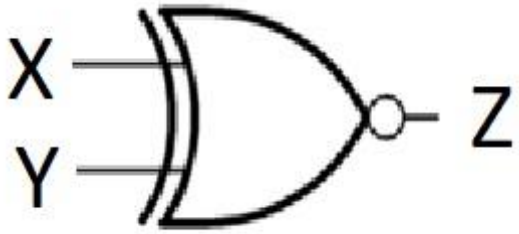


Imagen 1.23 Símbolo de la compuerta XNOR. [A10]

Actualmente ya no es necesario generar compuertas lógicas con elementos básicos, pues se vuelven circuitos muy voluminosos, la solución a esto fue encapsular los comportamientos de las compuertas en pequeños circuitos integrados, con esto se reduce el tamaño de los componentes a solo una pieza que contiene varias compuertas (del mismo tipo).

2.2.5 Circuitos Integrados

Estos circuitos integrados son divididos en:

- Monolítico: Este tipo de circuito se encuentra fabricado normalmente de silicio, aunque se pueden encontrar también de germanio.
- De Capa Delgada: Estos se encuentran hechos con materiales aisladores, como lo son la cerámica o vidrio. Gracias a esto los componentes integrados tienen un buen aislamiento entre ellos.
- De Capa Gruesa: En este circuito integrado, tanto resistencia como capacitor se forman sobre el sustrato y los transistores se añaden con pastillas discretas.
- Híbridos: Este es la combinación de los dos ya mencionados, sobre una pista de cerámica, por lo general se incluyen transistores discretos en caso de necesitar una alta disipación de potencia.

- Digitales y Lineales: Como sus nombres lo indican, uno se encarga de manejar señales digitales (0 y 1) aquí encontramos a las compuertas lógicas y dependiendo del nivel de complejidad podemos obtener microcontroladores o microprocesadores, y el otro se encarga de las señales analógicas, que son valores que cambian respecto al tiempo, los encontramos en amplificadores o receptores de radio.

También podemos clasificarlos dependiendo de su cantidad de componentes en:

- SSI (Small Scale Integration) entre 10 a 100 transistores.
- MSI (Medium Scale Integration) entre 101 a 1 000 transistores.
- LSI (Large Scale Integration) entre 1 001 a 10 000 transistores.
- VLSI (Very Large Scale Integration) entre 10 001 a 100 000 transistores.
- ULSI (Ultra Large Scale Integration) entre 100 001 a 1 000 000 transistores.
- GLSI (Giga Large Scale Integration) más de un millón de transistores.

La manipulación de la energía eléctrica gracias a elementos pasivos y activos implicados en un circuito es lo que podemos definir como electrónica, modificar la entrada para obtener el resultado deseado. Con la llegada de nuevos elementos, actualmente podemos tener circuitos en la palma de la mano.

Con esto podemos dar por concluida solo una parte de lo que es la electrónica, conociendo un poco de electricidad, unidades de medida, componentes de la electrónica y como evolucionaron hasta llegar a los circuitos integrados. En estos últimos se menciona a los microcontroladores y microprocesadores, de los cuales se hablará más adelante. Por ahora comenzaremos hablando de los inicios del software, como se iniciaron las prácticas que nos llevaron a lo que ahora conocemos como programas.

2.3 Inicios de los Sistemas de Software

El software nació por la necesidad de generar una máquina capas de efectuar cálculos matemáticos de forma rápida y precisa, algo parecido a lo que se buscaba al momento de pensar en la computadora. Todo comenzó en Londres a mediados del siglo XIX, donde Inglaterra pasaba por las óptimas condiciones para impulsar la investigación científica y el desarrollo de la industria. Eran los tiempos de la Revolución Industrial, el comercio textil crecía y los bancos requerían de mayor precisión en los cálculos. Aquí surge el primer personaje de este capítulo Charles Babbage del cual ya se habló tan sólo un poco en el subtema “Historia de la Computación”, pero en este punto hablaremos más a detalle pues además de ser uno de los precursores del cómputo, también lo fue de los sistemas de software.

2.3.1 Primeras ideas del software

Charles Babbage planteó una solución para el cálculo e impresión de las tablas de logaritmos con una mayor precisión, una maquina logarítmica. Se pueden hablar cientos de maravillas acerca de Babbage, pues no sólo se dedicó al aspecto científico, también aporto grandes ideas a la rama de la Economía, Ingeniería, Política y Literatura. Babbage comenzó su proyecto de una

máquina en 1823, a la cual llamo “Máquina Diferencial”, aquí comienza un largo camino de 10 años para lograr ver su sueño hecho realidad o al menos parte de su sueño, ya que para ese tiempo no existía la tecnología adecuada para diseñar las piezas que el buscaba.

Fue en su visita a Francia donde tuvo la oportunidad de conocer el funcionamiento de los telares de Joseph Marie Jacquard.

Imagen 1.24 Telar de Jacquard. [B14]

Este dispositivo está basado en tecnología parecida a la de un reloj, pero poseía una característica única nunca antes vista, aplicaba el uso de tarjetas perforadas, las cuales daban el patrón de diseño a la máquina para ser implementado en la tela. El tejido era controlado por una secuencia de estas tarjetas, las cuales formaban un programa de procesos. Para crear el mismo patrón en la tela, simplemente bastaba con volver a cargar el lote de tarjetas perforadas indicadas y la máquina hacía todo el trabajo. Este método fue adaptado también por IBM años después para sus primeras computadoras.

El diseño de esta máquina analítica se basaba en el uso de cilindros rotatorios llamados “barriles”, cada uno de los barriles estaría encargado del control de una sección de la máquina, como puede ser un almacén de números; todos estos barriles estarían controlados por un barril central, algo así como lo que ahora conocemos como procesador. Con el paso del tiempo Babbage fue ideando formas de crear dispositivos de entrada y salida de datos, fue así como incluyó un aparato para hacer gráficas y un mecanismo de impresión, además adaptó la idea obtenida del telar de Jacquard, un sistema de control a base de tarjetas perforadas.

En los tiempos de este gran inventor no existía la posibilidad de que una máquina almacenará programas, siempre se necesitaba de la intervención humana, alguien tendría que cargar las tarjetas perforadas con los datos, las ecuaciones y esperar el resultado.

2.3.1.1 Augusta Ada Byron

Charles Babbage no estuvo solo, existió alguien de gran importancia, Augusta Ada Byron conocida como Ada Lovelace. Ada tradujo al inglés un artículo escrito por Luigi Federico Menabrea a sugerencia del mismo Babbage, Ada agregó al documento notas y comentarios sobre las ideas y el funcionamiento de la máquina diferencial que enriquecieron aún más el documento.

Incluyó un programa para calcular los números de Bernoulli, de esta forma Ada Lovelace ganó el título de ser la primera mujer programadora en toda la historia. Gracias a estos dos personajes históricos podemos dar la definición de software:

De acuerdo con el Institute of Electrical and Electronics Engineers (IEEE) definimos al software como el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación.

2.3.1.2 Gottfried Wilhelm von Leibniz

Gottfried Wilhelm von Leibniz aportó al nacimiento del software el sistema binario y los fundamentos de la lógica simbólica, fue un gran científico y un gran apasionado de la matemática, a la corta edad de 15 años ingreso a la Universidad de Lepzig, donde estudio derecho y filosofía, pero fue en Génova donde estudió matemáticas.

Decidió dedicarse al estudio de la investigación lógica, haciéndose creador del algoritmo* lógico. Además, gracias a este personaje, el uso del sistema binario se volvió más usado, ya que logro demostrar mediante "numeración diádica" que cualquier numero puede ser representado con una serie de 1's y 0's.

2.3.1.3 George Boole

Otro personaje que tomo las ideas de Leibniz y las llevo más allá fue George Boole. Boole siempre quiso demostrar que la lógica no pertenece al área de la filosofía sino al de la matemática, fue así que tuvo la idea de expresar el pensamiento humano a través de símbolos que se pudieran operar haciendo uso de ciertas leyes y con esto se dio vida al álgebra de Boole.

El álgebra booleana consiste de un conjunto de símbolos y reglas, las cuales no solo sirven para operar con números, sino también con letras, objetos y conceptos en general. En la aritmética convencional se tienen como principales operadores a la suma (+), la resta (-), la división (/) y la multiplicación (*), pero en el álgebra booleana solo contamos con tres, AND OR y NOT, (Y, O, NO).

Como se vio en el capítulo de Electrónica, cada una de las operaciones tiene su tabla**.

Estas tres simples operaciones constan el Algebra Booleana, y con esto podemos realizar operaciones más complicadas. El Algebra Booleana es una de las herramientas más ocupadas y de más grande ayuda a la hora de desarrollar software, es fundamental al momento del plantear los algoritmos.

2.3.1.4 Alan Turing

Personalmente Turing fue uno de los precursores de la informática, tanto de la computadora como del software, más importantes para mí. Fue una persona extrovertida y excéntrica, alguien que sólo se dignaba a tener conversaciones con gente de su nivel intelectual.

*: Serie de pasos finitos y ordenados para resolver algún problema.
**: Revisar tablas de AND, OR y NOT (multiplicación, suma, negación, respectivamente).

Alan Turing nació en el año de 1912. A los 24 años publicó uno de los documentos más importantes acerca del cómputo, mucho tiempo antes de que la primera computadora fuera creada, la publicación se tituló “On Cumputable Numbers” y fue publicada en el año de 1936, en este documento Turing especifica las características y limitaciones de una máquina lógica, sus intenciones nunca fueron crear la máquina, solo redacto su estructura lógica, jamás hablo sobre componentes, circuitos o algo semejante, se enfocó únicamente al funcionamiento. La Máquina de Turing únicamente vive en papel.

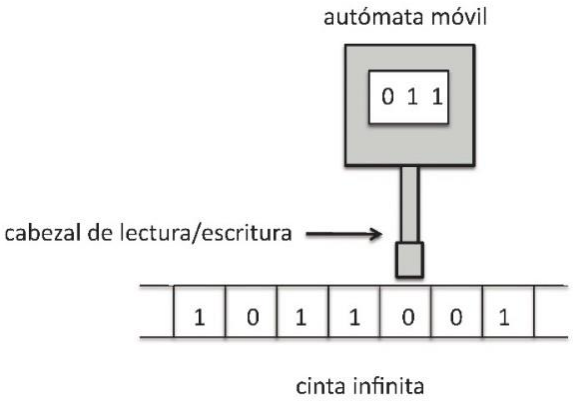


Imagen 1.25 Máquina de Turing. [B15]

La máquina de Turing consistía de:

- Una cinta dividida en celdas, cada ceda contiene en su interior algún símbolo de un alfabeto finito, dicho alfabeto debe contener un símbolo especial al cual se le llama Blanco y uno o más símbolos adicionales. La cinta es extensible a la izquierda y a la derecha, lo que quiere decir, que se suministrara cuanta cinta necesite la máquina de Turing para computar.
- Un cabezal el cual está encargado de la lectura y escritura de símbolos en la cinta por celda.
- Un registro de estado, cuando inicia la máquina de Turing, se tiene presente un estado inicial, y se debe tener un estado final. Los estados deben ser finitos. En este registro de estado se almacena el estado de la máquina.
- Una tabla de acción, la cual no es más que una tabla de instrucciones. Usualmente consta de 5 tupas y funciona de la siguiente manera:
    1. Dado el estado X
    2. En el estado Y
    3. Leyendo el símbolo Z en la cinta
    4. Borra, escribe o borra y escribe un símbolo.
    5. Mueve el cabezal, ya sea hacia la izquierda, derecha o permanece en el mismo lugar
    6. Asigna el mismo o un nuevo estado.

Se considera a la máquina de Turing como el prototipo de la Máquina Lógica.

Cabe recalcar que Turing planteó un concepto fundamental sobre el cual se rigen las computadoras actuales “La máquina debe ser capaz de manejar información haciendo uso de un programa modificable para que pueda ejecutar diversas funciones”.

Durante la Segunda Guerra Mundial (la cual constó de un tiempo entre 1939-1945), Alan Turing fue llevado a una instalación secreta del gobierno británico llamada Bletchley Pack, fue instalado en este lugar con el propósito de descifrar códigos de transmisión del ejército alemán, los cuales eran encriptados haciendo uso de una máquina conocida como ENIGMA.

ENIGMA era una máquina constituida por un mecanismo de cifrado rotatorio, comenzó a ser muy usada en Europa en el año de 1920, tomando fama gracias a su uso por la fuerzas militares alemanas en 1930.

Durante su estancia en las instalaciones de Bletchley Pack, Turing comenzó a desarrollar grandes ideas sobre máquinas inteligentes. Otra de sus publicaciones más destacadas fue un libro titulado “Computing Machinery and Inteligence” el cual es uno de los documentos más importantes en la historia del software.

2.3.1.5 Jon von Neumann

John von Neumann, uno de los matemáticos más importantes durante la segunda guerra mundial en Estados Unidos, al igual que Turing, von Neumann siempre destacó en el área de las matemáticas, en 1930 llego a Estados Unidos comenzando su nueva vida como profesor de matemáticas en la Universidad de Princeton, participó junto J. Presper Eckert y John William Mauchly en la creación de la primer computadora digital nombrada EDVAC (Electronic Discrete Variable Automatic Calculator). Esta máquina fue pensada mucho antes de que ENIAC fuera puesta en funcionamiento, tanto Eckert como Mauchly participaron en la creación de ENIAC (Electronic Numerical Integrator And Computer), fue por eso que accedieron al desarrollo de EDVAC.

A diferencia de ENIAC, EDVAC era capaz de almacenar su programación, en el caso de ENIAC su programación consistía del cambio de conexiones físicas, estaba formada de al menos 1,500 relés, con los cuales se conectaban los pulsos eléctricos por las unidades del sistema. La programación de ENIAC estuvo a cargo de 6 mujeres, recordemos que durante la Segunda Guerra Mundial, los hombres se encontraban en combate, fue por eso que las mujeres al permanecer en el país, se encargaron de hacer los cálculos. ENIAC fue construida para calcular la trayectoria de proyectiles. Antes de la creación de ENIAC, un grupo de 80 mujeres eran las encargadas de los cálculos, fueron consideradas computadoras humanas, con la llegada de ENIAC, 6 de las 80 mujeres “computadora”, fueron seleccionadas para la programación de la que fue considerada, la primera computadora de propósito general, opacando así el invento de Zuse la “Z1”.

Estas 6 mujeres, junto a Ada Lovelice, son consideradas las primeras programadoras de la historia, tristemente Betty Snyder Holberton, Jean Jennings Bartik, Kathleen McNulty Mauchly Antonelli, Marlyn Wescoff Meltzer, Ruth Lichterman Teitelbaum y Frances Bilas Spence, no fueron reconocidas hasta después de los años 80’s donde se creía que ellas eran sólo modelos para hacer lucir mejor a ENIAC. La verdad fue que gracias a estas increíbles 6 mujeres sentaron

las bases para la programación sencilla y accesible que tenemos hoy en día, a pesar de que en su tiempo ENIAC debía ser programada en lenguaje máquina puro, recableando el sistema.

Volviendo a John von Neumann, estableció cuatro puntos importantes en el hecho de "codificar", tal cual lo llamo el a la acción que hoy conocemos como programar:

1- Simplicidad y consistencia de las soluciones para su codificación.
2- Solidez y suficiencia de la codificación.
3- Facilidad y rapidez para traducir expresiones matemáticas a la codificación también para encontrar y corregir errores.
4- Eficiencia de la codificación para llevar a la máquina cerca de su máxima velocidad.

Von Neumann definió la programación como una secuencia de símbolos codificados que debían ser introducidos dentro de una memoria con el fin de hacer que la máquina ejecutara una serie de sentencias secuenciales para resolver el problema en cuestión.

Herman Goldstine y John von Neumann crearon los diagramas de flujo en el año de 1946, para modelar los programas de la computadora ENIAC. Los diagramas de flujo son una herramienta sumamente importante a la hora de representar la secuencia de un algoritmo. El diagrama de flujo consistía de rectángulos conectados por líneas y flechas, el inicio y el final del diagrama se representaban con un pequeño círculo o un triángulo equilátero. Con el paso del tiempo, lo diagramas de flujo se han ido modificando y aun en la actualidad continúan siendo de gran utilidad para los programadores.

2.3.2 Primeros programas

Como podemos ver, los primeros programas tenían como finalidad resolver problemas matemáticos, hoy en día la computadora se utiliza para diversas cosas, no únicamente para el ambiente científico, pero al momento de programar, seguimos ocupando matemática y lógica. A partir de los años 40 en adelante el diseño de software fue evolucionando, comenzando con recableado, hasta la conocida máquina de Zuse Z4, el cual implementaba uno de los primeros, si no el primer lenguaje de programación llamado "**Plankalkül**".

El primer programa almacenado en la memoria de la máquina fue ejecutado en la ENIAC en 1948, o al menos en América fue así, pues al otro lado en Inglaterra solo unos meses antes el prototipo de la Manchester Mark I logró ejecutar un programa almacenado. Recordemos que la Manchester Mark I fue un proyecto donde participo Alan Turing, creando un lenguaje de programación especial para la Mark I, al ser un prototipo, no se le atribuye el éxito de ser la primera computadora en correr un programa almacenado.

El Boot Straping, fue un término denominado a un conjunto de instrucciones que permiten ordenar la carga del sistema operativo de la computadora para iniciar su ejecución, en el libro "La magia del Software", el autor cuenta como Maurice Wilkes relata la historia del primer cargador primario (boot strap) en 1949, "David Wheeler contribuyó de manera importante en la construcción de la EDSAC como estudiante de verano y su primer contribución fue escribir las órdenes iniciales que fueron grabadas en una memoria mecánica de sólo lectura" (Nuncio, 2016). Desde entonces el Boot Strap es de gran importancia cuando se habla de software básico.

En el año de 1952 Alik E. Glennie creo AUTOCODE, el primer compilador usado en una computadora, trabajo en varios proyectos de computación incluido en el Manchster Mark I. Glennie cuestionó el uso de la programación a nivel máquina, explicó que para ser una máquina de uso general la forma de utilizarla era muy complicada, es por eso que redactó su opinión acerca de utilizar una notación conocida para la programación, así se tienen las ventajas en la eliminación de errores y simplifica los programas.

El compilador A-2 fue pensado por Grace Murray Hopper, mejor conocida como Amazing Grace, una bríllate matemática participante en el proyecto de UNIVAC (Universal Automatic Computer), la primera computadora comercial en Estados Unidos. Grace y un equipo conformado por participantes del proyecto UNIVAC escribieron un programa que pudiera hacer la integración de rutinas y transformarlas a lenguaje máquina. A este programa se le denominó como “Rutina de compilación”.

Fue con la llegada del lenguaje de programación FORTRAN (FORmula TRANslation) que la programación metódica y eficiente surgió. En 1954 se dio a conocer el primer FORTRAN creado por John Backus, y fue el primer lenguaje de programación de alto nivel de propósito general. La forma de programar fortran solía ser por medio de tarjetas perforadas de 80 columnas, la finalidad de FORTRAN siempre fue enfocada hacia científicos y el área de las Ingenierías, para la ayuda de cálculos matemáticos, pues las computadoras fueron creadas con esa finalidad.

Como se puede notar, la programación ha cambiado mucho, desde las ideas de Charles Babbage y Ada Lovelace hasta la creación del primer lenguaje de alto nivel. Hoy en día lo más utilizado son los lenguajes de alto nivel, pero en realidad ¿qué se está programando?. En sus inicios, las máquinas gigantes a las cuales denominaron las primeras computadoras, se realizaba un cambio de conexiones físicas, se apretaban algunos interruptores y así se generaba un nuevo programa (lenguaje máquina puro), después comenzaron a usarse tarjetas perforadas, las cuales eran leídas y procesadas entregando un resultado, esto se lograba a través de un control en los componentes del sistema (Hardware), y fue con la llegada del CHIP o CI’s que hoy tenemos microcontroladores y microprocesadores, para logar llegar a los “micros” actuales se tuvieron que generar las Arquitecturas de Computadoras.

2.4 Arquitectura de Computadoras

Desde las máquinas de Babbage ya se tenía una idea del cómo debe ir estructurada una máquina de cálculos, pero fue con la llegada de ENIAC que John von Neumann tuvo la idea de generar un modelo estándar por el cual se debe regir el funcionamiento de una computadora.

2.4.1 Arquitectura de von Neumann

Una vez más aparece este personaje, pues sus ideas marcaron uno de los patrones de diseño de computadoras que se tienen actualmente, además encontramos su arquitectura en los microcontroladores actuales.

Von Neumann explicó que se requiere de una computadora digital capaz de almacenar la información necesaria para cualquier proceso, debe guardar los resultados obtenidos del

proceso inmediatamente y las instrucciones de rutina para realizar tal proceso. Explicó la idea de contar únicamente con un solo tipo de memoria tanto para el almacenamiento de números como el de instrucciones. Teniendo en cuenta que las instrucciones solo vivirían en memoria para ser almacenadas, debe existir algo que ejecute las instrucciones almacenadas, a esto lo llamo Unidad de Control. Ya con el almacén de números e instrucciones, la unidad de control para procesar dichas instrucciones, debe existir un lugar donde realizar las operaciones aritméticas elementales, es decir, la computadora debe estar integrada por una unidad de control, una memoria y una unidad de aritmética donde se puedan desarrollar sumas, restas, multiplicaciones y divisiones.

A esto se le conoce como Arquitectura de Von Neumann, y es actualmente utilizada en las computadoras personales de uso general.

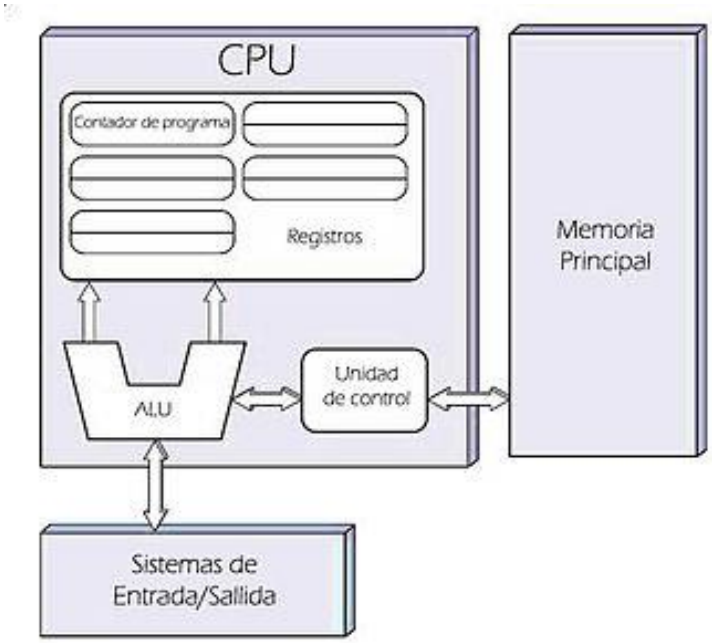


Imagen 1.26 Arquitectura de Von Neumann. [B16]

En respuesta a la arquitectura de Von Neumann se tiene también la Arquitectura de Harvard.

2.4.2 Arquitectura de Harvard

A diferencia de la Arquitectura de von Neumann la arquitectura de Harvard separa el almacén de información y números en dos, una especial para las instrucciones y otra para los números, esto hace más rápido el procesamiento, pues en lugar de esperar a que tanto las instrucciones como los números pasen por el mismo canal de comunicación, puede estar recibiendo instrucciones y datos al mismo tiempo, por lo tanto el procesamiento de la información es más rápida.

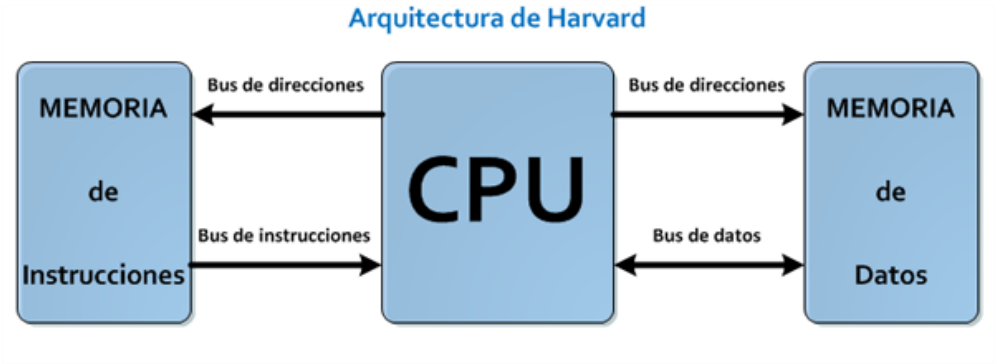


Imagen 1.27 Arquitectura Harvard. [B17]

La arquitectura de Harvard es utilizada en algunos microcontroladores y procesadores.

Entonces podemos decir que la arquitectura de computadora es una lista de requerimientos para su funcionamiento y dependiendo de cuál sea su fin, se ocupará la arquitectura adecuada, lo mismo con los microcontroladores, dependiendo de su propósito se buscara alguno con arquitectura adecuada.

Para continuar con la Arquitectura de von Neumann, hablaremos sobre los componentes que la conforman.

- Unidad de control: Es el encargado de la implementación de los algoritmos con los datos e información proveniente de la memoria, realza las funciones de dirección, monitoreo y responde a emergencias o interrupciones que se pudieran ocasionar. Como su nombre lo dice es el encargado de dar el control a la computadora.
- Unidad de aritmética: También llamada ALU (Arithmetic Logic Unit) o Unidad Aritmética y Lógica, ese el área encargada de las operaciones tanto elementales como la suma y resta como también de las operaciones lógicas. Como se vio anteriormente la lógica de Boole nos sirve para realizar operaciones como AND, OR, etc.
- Memoria: Este es el almacén (según Neumann) donde permanecen los datos a ser procesados tanto por la unidad de control como por la ALU para entregar la información de salida, la memoria interactúa repetidas veces y constantemente con el CPU (unidad central de procesamiento), el cual está compuesto por las unidades previamente mencionadas.
- Sistemas de Entrada y Salida: Son los periféricos con los cuales podemos ingresar los datos a ser procesados y la visualización del resultado obtenido.

No sólo se cuentan con estas dos arquitecturas, los microcontroladores y microprocesadores se pueden clasificar dependiendo de su juego de instrucciones. El conjunto de instrucciones consta de palabras para la comunicación usuario máquina. Hay que recordar que las computadoras no entienden como los humanos, ni siquiera hablamos el mismo idioma por más que así parezca. Las computadoras únicamente entienden en código binario (1 y 0), por lo tanto fue necesario emplear una forma en que los usuarios y las máquinas se entendieran. Una de esas herramientas es el “set de instrucciones”. Dependiendo de la arquitectura manejada, sabemos que instrucciones podemos ocupar para comunicarnos con la computadora.

2.4.3 Arquitecturas manejadas por conjunto de instrucciones

Arquitectura CISC (Complex Instruction Set Computer): El juego de instrucciones que posee este tipo de arquitecturas es extenso, además que posee instrucciones potentes, las cuales son capaces de realizar tareas complicadas haciendo uso de una sola instrucción. Aunque este tipo de arquitectura tiene su lado malo, pues se dificulta el paralelismo en las instrucciones. La mayoría de los microcontroladores actuales ocupa este tipo de arquitectura.

Arquitectura RISC (Reduced Instruction Set Computer): Este tipo de arquitectura posee un conjunto de instrucciones más reducido a diferencia de CISC y mucho más sencillas, por lo tanto se reduce el tiempo invertido por instrucción. Gracias a este tipo de arquitectura se tiene la posibilidad de segmentación y paralelismo en la ejecución de instrucciones, además reduce los accesos a memoria ya que solo instrucciones de carga y almacenamiento acceden directamente a la memoria de datos.

Un dato curioso de este tipo de arquitecturas es el hecho de que algunas veces los microprocesadores basados en arquitecturas CISC descomponen ciertas instrucciones para hacerlas más del tipo RISC y así reducir tiempos de ejecución.

Arquitectura SISC (Simple Instruction Set Computing): Aquí el conjunto de instrucciones es específico a la tarea que se va a ejecutar. Aquí nos apartamos más de un sistema de propósito general para enfocarnos únicamente a una meta única.

Pero ya comenzamos a hablar de microcontroladores y microprocesadores, sin saber en realidad que son, como así es el caso, comenzaremos a hablar de ellos.

2.5 Microcontroladores y Microprocesadores

Microcontroladores:

Un controlador es un dispositivo el cual se emplea para, controlar algún aspecto de un sistema. Recibe una señal del estado del sistema y genera procesos de control para llevar a cabo la su función.

Algunos años en el pasado los controladores solían ser creados con componentes discretos o elementos pasivos y activos, lo cual hacía de estos controladores algo sumamente estorboso, pero fue gracias a la llegada de los microprocesadores que los controladores pasaron a ser diseñados en una placa de circuito impreso de mucho menos tamaño.

El microcontrolador estándar contiene todos los elementos de un ordenador, posee en su interior una unidad de control, una unidad de aritmética y lógica, un gestor de interrupciones y sus respectivas entradas y salidas, pero no sólo eso también se cuentan con algunos elementos básicos de control como lo son temporizadores, contadores, generadores de PWM entre otros. Se puede decir en conclusión que los microcontroladores son circuitos integrados con los componentes de un computador o bien son pequeñas computadoras incluidas en un pequeño CI.

Las arquitecturas más comunes en el diseño de los microcontroladores son:

Arquitectura de von Neumann y la arquitectura Harvard, usualmente y como ya vimos con anterioridad, es más común encontrar microcontroladores con arquitectura de Harvard, pero también los hay con von Neumann. También debemos recordar que acompañadas de estas dos arquitecturas tenemos el conjunto de instrucciones, las cales son CISC, RISC y SISC, entre estas la más usada es de tipo CISC.

Una vez conociendo la arquitectura empleada en los microcontroladores, aprenderemos un poco sobre las memorias que estos utilizan.

La memoria es el almacén de datos e instrucciones que posee el dispositivo microcontrolador, como se sabe, depende de la arquitectura para saber cómo se comunica la memoria (o memorias) con el controlador, las memorias se clasifican por su forma de grabado o borrado, es decir las veces que podemos borrar la información contenida en ellas.

EPROM (Erasable Programmable Read Only Memory): Este tipo de memoria es no volátil, eso quiere decir que no requiere de energía para continuar almacenando los datos previamente cargados, la forma de borrar lo que en ella se contiene es a través de rayos UV, El grabado de esta memoria requiere de una tensión aproximada de 12 a 25 volts.

Imagen 1.28 Memoria EPROM. [B18]

EEPROM (Electrically Erasable Programmable Read Only Memory): Este es otro tipo de memoria no volátil, esta memoria a diferencia de la EPROM, permite el borrado de los datos no por medio de rayos UV si no de forma eléctrica, Es posible borrar y grabar los bytes o palabras de la memoria de forma individual.

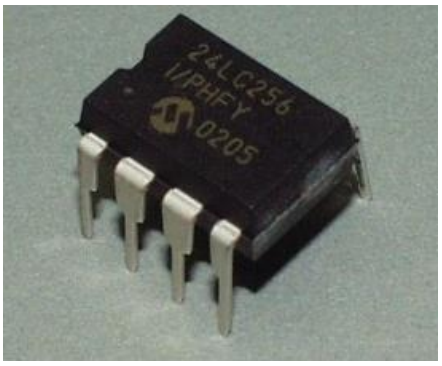


Imagen 1.29 Memoria EEPROM. [B19]

OTP (One Time Programmable): Esta memoria es similar a la EPROM con la diferencia que no se puede borrar la información contenida por ningún medio. La EPROM contiene un apartado

dentro de su estructura física (encapsulado) que permite la entrada de rayos UV en caso de necesitar ser borrada, pero en la OTP no se cuenta con esa accesibilidad en el encapsulado.

Imagen 1.30 Memoria OTP. [B20]

FLASH (EPROM): Esta es el tipo de memoria más común y la cual está reemplazando a las memorias EEPROM, ya que su grabado no requiere de tenciones altas, únicamente requiere de un voltaje común de trabajo, además el borrado se hace en bloques enteros de la memoria.

Imagen 1.31 Memoria Flash. [B21]

NVRAM (Non- Volatile RAM): Esta memoria es de tipo RAM estática, la memoria RAM continua sus trabajo mientras posea una corriente en ella, en este caso, siempre debe ser alimentada por algún tipo de batería.

Imagen 1.32 Memoria NVRAM. [B22]

Los microcontroladores más utilizados son los de 8 bits (palabras de 8 bits), Aunque comúnmente no es necesario más de 4 bits.

Entre las marcas de microcontroladores más populares podemos encontrar a INTEL, Motorola y MicroChip, algunos modelos por mencionar son:

INTEL 8048: El cual fue el primer microcontrolador, por el precio y sus herramientas al momento de desarrollar en él lo hacen popular aun hoy en día.

Imagen 1.33 Microcontrolador Intel 8048. [B23]

INTEL 8051: Este microcontrolador es muy potente y fácil de programar, lo cual le dio su popularidad y fama actuales, además que igualmente es uno de los más utilizados, posee una alta gama de herramientas de desarrollo, además de su extensa documentación, sin lugar a dudas uno de los microcontroladores más recomendados.

Imagen 1.34 Microcontrolador Intel 8051. [B24]

PIC de MicroChip: Personalmente es el único que he tenido el placer de programar, algo complicado pero bastante completo a mi parecer, algo muy interesante de este microcontrolador es que fue el primero en utilizar su set de instrucciones de tipo RISC.

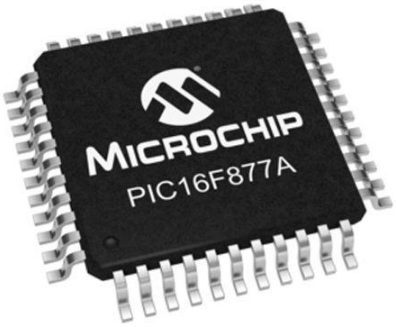


Imagen 1.35 Pic de Microchip 16f877A. [B25]

Los microcontroladores necesitan ser programados para su funcionamiento, a través de la historia han existido muchas maneras de programar, desde el recableado de ciertas terminales de la computadora, hasta darle indicaciones por medio del teclado. Los microcontroladores se programan de forma similar, se necesita de un lenguaje de programación y una forma de cargar

el programa al microcontrolador, como ya vimos existen diferentes tipos de memoria, pero los microcontroladores comerciales en los que nos enfocaremos ahora tienen la posibilidad del grabado y borrado de la información por medio de impulsos eléctricos.

Los dos lenguajes más comunes a la hora de programar un microcontrolador son:

Ensamblador: Este tipo de lenguaje es de bajo nivel, está conformado por mnemónicos (conjunto de códigos simbólicos, cada uno corresponde a una instrucción), es un tipo de lenguaje complicado, pues lleva su tiempo entenderlo y aún más aplicarlo, pero la máxima ventaja que se tiene con este lenguaje es el control total de microcontrolador, ya que sus instrucciones rozan el lenguaje máquina, a continuación se muestra un ejemplo de un programa desarrollado en ensamblador:

```
        LIST    P=16F84A,                               ; usar PIC 16F84A
        #include <p16f84A.inc>
        __CONFIG _CP_OFF&_PWRTE_ON&_WDT_OFF&_XT_OSC     ; code protec           off
                                                        ; power up timer        on
                                                        ; watchdog              off
                                                        ; osc                   XT
        ORG     0
        BSF     STATUS,5                                ; activa la página 1
        MOVLW   B'00000'                                ; carga 00000 en W
        MOVWF   TRISA                                   ; puerto a todos salidas
        MOVLW   B'00000000'                             ; carga 00000000 en W
        MOVWF   TRISB                                   ; puerto b todos salidas
        BCF     STATUS,5                                ; volvemos a la página 0
        BCF     PORTB,0                                 ; ponemos a 0 RB0
        BCF     PORTB,1                                 ; ponemos a 0 RB1
        BCF     PORTB,2                                 ; ponemos a 0 RB2
        BCF     PORTB,3                                 ; ponemos a 0 RB3
        BCF     PORTB,4                                 ; ponemos a 0 RB4
        BCF     PORTB,5                                 ; ponemos a 0 RB5
        BCF     PORTB,6                                 ; ponemos a 0 RB6
        BCF     PORTB,7                                 ; ponemos a 0 RB7
       INICIO                                                 ; etiqueta
        BSF     PORTB,0                                 ; pone a 1 RB0
        GOTO    INICIO                                  ; va a inicio
        END                                             ; fin de programa
```

Contribución realizada por Hugo en el foro “TODOPIC” (H, 2006).

Lenguaje C: Es un tipo de lenguaje considerado de alto nivel, aunque personalmente lo veo de mediano nivel, pues aunque tiene características de alto nivel como una sintaxis más entendible y no requerir de tantas instrucciones, para una simple función como lo es prender un led (ejemplo anterior) aún posee ciertas características de bajo nivel, algunas veces el uso de mnemónicos.

A continuación un ejemplo en C:

```
#include <16f877a.h>
#fuses xt,nowdt
#use delay(clock=4000000)
#define Led pin_b0 //Nombra a pin_b0 como Led
#define Retardo delay_ms(500) //Nombra a delay_ms() como Retardo
void main(){
```

```
//Configuracion de puertos E/S
SET_TRIS_B(0x00);//Configura el puerto B: 0=Salida; 1=Entrada
output_b(0x00); //Limpia el registro del puerto

//Estructura Programa principal
inicio://Etiqueta de retorno
output_high(led);//Pone en alto a led
retardo;//Demora
output_low(led);//Pone en bajo a led
retardo;//Demora
goto inicio;//Regresa a la etiqueta inicio
}
```

Contribución realizada por SAN_TELMO en BLOGSPOT (S, 2006).

Esta programación, al menos para mí, es un poco más entendible que en lenguaje Ensamblador, aunque ambos códigos hacen prácticamente lo mismo.

Para cargar el programa se necesita de un “Grabador”, como el que se muestra a continuación.

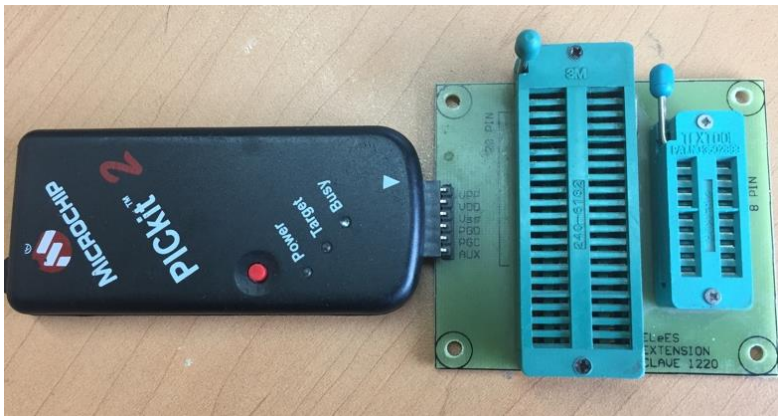


Imagen 1.36 Grabador PICkit 2. [A11]

Microprocesadores:

Como ya vimos el microcontrolador es una especie de computadora encapsulada de una sola pieza, pero como tal una computadora común requiere de cinco componentes los cuales son memoria, dispositivo de entrada, dispositivo de salida, microprocesador y un reloj para sincronizar las acciones de la computadora. El propósito de una computadora es que a partir de un programa o bien una serie de instrucciones almacenadas se lleven a cabo operaciones sobre un conjunto de datos. Entonces el microprocesador es un componente de la computadora que se encarga de ejecutar toda la manipulación de los datos además de llevar el control de los demás componentes.

Comúnmente dentro de un microprocesador podemos encontrar tres subsecciones.

1- Archivos de registro: En este lugar se almacenan los datos que el procesador tiene en uso mientras se ejecuta el programa, además puede incluirse información como el estado del procesador y su contador de programa, el cual permite al procesador llevar el seguimiento de su posición actual en el programa, por así decirlo, saber dónde está.

2- Apartado de control: Esta subsección es la encargada de saber que función va a ser la que interactúe con los datos, esto gracias a la decodificación de las instrucciones que entran al procesador. El controlador también está encargado de las acciones ejecutadas por otras secciones externas al procesador.

3- Unidad de Aritmética y Lógica: Aquí es donde se llevan a cabo todas las operaciones con los datos, suma, resta, etc.

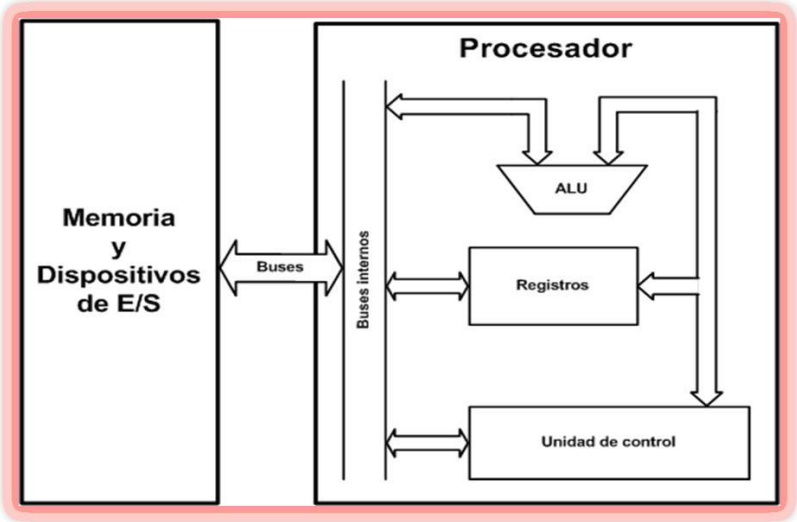


Imagen 1.37 Diagrama a bloques de un Procesador común. [B26]

Veamos un poco de la historia del microprocesador, comenzando con los primeros microprocesadores:

INTEL 4004: Creado en 1971 fue el primer microprocesador creado en un solo chip. Cabe mencionar que antiguamente el procesador estaba dividido en 3 partes, la ALU, la unidad de control y el banco de registros. INTEL como en los microcontroladores fue el pionero en la creación de microprocesadores y el 4004 fue el que dio origen a todos los demás. Era un CPU comercial y de 4 bits.

INTEL 8080: Este fue el primer CPU de la primera computadora personal el Altair 8800, hay que hacer un paréntesis aquí, pues el Altair 8800 fue considerado el primer kit de electrónica, una serie de componentes que uno mismo podía ensamblar para crear su máquina personal.

Z80: Basado en el Intel 8080, este microprocesador es característico por contar con una longitud de palabra de 8 bits, creado en el año 1976 por la compañía Zilog. Un año más tarde se da el lanzamiento de la primera computadora con el microprocesador z80 a 1.77 MHz u con una memoria RAM de 4KB.

INTEL 8086-8088: Creados en 1978, estos dos microprocesadores fueron de suma importancia para la empresa INTEL, ya que al realizar una buena venta hacia la empresa IBM, INTEL logró posicionarse en la lista de las mejores 500 empresas por parte de la revista FORTUNE.

AMx86: Fabricados por la empresa AMD, fueron conocidos como los clones de INTEL, pues eran completamente compatibles con los códigos de INTEL, aunque AMD logró superar la frecuencia de reloj de los procesadores INTEL.

De aquí en adelante las dos empresas más conocidas a nivel mundial han sido INTEL y AMD, con la llegada de INTEL PENTIUM, PENTIUM PRO, PENTIUM II, CELERON, PENTIUM III y IV, hasta llegar a la actual CORE i3, i5 e i7 hasta el momento. Por parte de AMD tenemos el K6 y K6-2, K7, ATHLON XP, PHENOM hasta llegar a FUSION.

Viendo a los microprocesadores por su estructura lógica podemos detallar que está conformado por registros, una unidad de control, una unidad de aritmética y lógica y algunas veces, dependiendo del fabricante se tiene un apartado de coma flotante.

Las instrucciones almacenadas en memoria principal como números binarios organizados secuencialmente son ejecutadas por el microprocesador de la siguiente forma:

- Prefech: Esta fase consta de realizar la prelectura de la instrucción desde memoria principal.
- Fetch: Es la fase de envío, la instrucción se manda al decodificador.
- Decodificación: Aquí se determina qué tipo de instrucción es y por lo tanto que se debe de realizar.
- Lectura: En caso de ser requerido, se realiza una lectura de operando.
- Ejecución: Se lanzan las máquinas de estado que llevan a cabo los procedimientos necesarios.
- Escritura: Aquí se graban los resultados en memoria principal o en los registros.

Estas fases suelen ser realizada de forma cíclica, la duración de estos ciclos es determinada por la frecuencia de reloj.

Los registros son técnicamente un tipo de memoria de alta velocidad y poca capacidad con fines especiales que el micro posee para realizar ciertas funciones, en total se cuentan con 32 registros.

La memoria interna que posee el microprocesador es conocida como Memoria Cache, es un tipo de memoria de acceso rápido, donde se almacenan temporalmente los datos de los procesos en ejecución, existen varios niveles de memoria caché, L1,L2 y L3.

L1: Se encuentra en el núcleo del microprocesador y posee una capacidad de 256 kb. Se utiliza para el almacén de datos e instrucciones importantes, se divide en 2 subniveles, Data cache e Instruction Cache.

L2: Es la encargada de almacenar datos de uso frecuente, es más lenta que L1 pero es más rápida que la memoria principal (RAM), También se ubica en el microprocesador pero no directamente en el núcleo.

L3: Es el último nivel de las memorias localizadas en el microprocesador, esta memoria es más lenta que L2 pero continua siendo más rápida que RAM. A esta memoria se accede en caso de no tener los datos necesarios en L1 y L2.

El evolucionar de las máquinas analógicas a sistemas digitales ha sido una travesía sumamente complicada, teniendo un panorama donde antes las máquinas tenían el tamaño de edificios y ahora poseemos todo un sistema computacional en la mano (microcontroladores), es algo increíble.

La manera de programar cambio extremadamente, utilizando en las primeras máquinas el recableado y por lo tanto una configuración diferente hasta la programación en Ensamblador y C, donde el programa ya es almacenado y el sistema sabe cómo ejecutarlo. Haciendo uso de los programas almacenados y grabándolos en los microcontroladores es como tenemos kits de electrónica, donde el microcontrolador por sí solo no prendera el led, puede que en su interior ya contenga el código que indique a este la señal que debe mandar, pero necesitamos de elementos pasivos o activos para completar el circuito.

Todo ha sido un cambio y constantemente nos encontramos en la actualización de los sistemas, combinando tanto la electrónica como la programación para generar kits de electrónica más simples de utilizar.

En el siguiente capítulo, conoceremos más sobre un término que apenas comienza a surgir por el mundo, el cual se refiere al uso de kits de desarrollo de electrónica actuales.

3) *Integración Digital*

Al igual que los capítulos anteriores, se inicia con la descripción general del tema y posteriormente se presenta el tema con detalle. Para una comprensión más clara del tema se inicia con la definición de algunos términos.

- Integración: El diccionario de la lengua española lo define como “Acción o efecto de Integrar” aunque también nos da otra definición “Proceso de unificación de varias entidades antagónicas” (Programa educativo visual, 1995).
- Digital: Que suministra los datos mediante dígitos o elementos finitos o discretos*.

Con base en las definiciones anteriores, para el presente trabajo se realiza una definición conjunta. Integración Digital es la unificación de una o más máquinas que suministran datos haciendo uso de dígitos. Pero ¿qué queremos decir con unificar?, se sabe que unificar es reducir muchas cosas a una o hacer de ellas un todo; con esto vamos a dar una definición un tanto más clara:

Integración Digital es el hecho de tomar diversas máquinas capaces de manejar señales discretas y hacer de todas ellas una sola, dicho de otra forma, hace referencia a la idea de tomar la información o datos de cualquier dispositivo electrónico para ser manipulados por otro dispositivo, haciendo uso de algún protocolo de comunicación.

Esta última definición es un tanto más técnica pero es la más adecuada al contexto que se quiere manejar en este trabajo.

Podemos poner el ejemplo de dos dispositivos uno como monitor de datos y el otro como sensor, el sensor toma los datos medidos (de ser un sensor analógico transforma la señal obtenida a una señal digital en algunos casos) y supongamos que ambos están conectados por protocolo de I2C (“bus de comunicaciones que hace uso de dos líneas para transmitir la información: una para datos y otra para señal de reloj”)**, por lo tanto el sensor mandará la información digital al dispositivo que se encuentra monitoreándolo, recibirá los datos por el protocolo indicado y los podrá manipular a su gusto, ya sea mostrándolos al usuario o haciendo cualquier otra transformación para ser delegados. El punto es tener el control y manipulación de datos por ambos dispositivos electrónicos. Esto es exactamente lo que ocurre en el presente proyecto, tener el control del brazo mecánico a través de una interfaz de usuario con la cual también podamos monitorear ciertos aspectos del brazo y conseguir impedir una volcadura en él.

La integración digital va más allá todavía, existen un conjunto de técnicas y herramientas que nos permiten automatizar algunos aspectos del hogar, a esto se le conoce como Domótica, teniendo en cuenta esto, podemos notar que la integración digital puede estar presente en cualquier hogar.

*: “Diccionario de Google”: https://www.google.com.mx/search?newwindow=1&hl=es&source=hp&ei=ITukWtyBPaGmjwSdyI6YBw&q=digital&oq=digital&gs_l=psy-ab.3..35i39k1l2j0i67k1l3j0l2j0i203k1j0l2.22277.23124.0.23365.10.8.1.0.0.0.160.726.1j5.7.0....0...1c.1.64.psy-ab..2.8.820.6..0i131i67k1j0i131k1.93.lLL22IZICxU

**: Pena, M. & Pardal, J. (2015). Mini estación meteorológica. En Arduino y Node.js(p. 90). sin informacion: Javier Pardal Moncho Pena.

Por ejemplo teniendo un control de iluminación completo o parcial de la vivienda mediante alguna aplicación instalada en nuestro dispositivo móvil o algún otro sistema de hardware enfocado a este control, podríamos fácilmente apagar o prender las luces haciendo uso del celular. Ahora supongamos que debemos tener el control total del hogar, luces, clima, controles de acceso y cámaras colocadas en él, pero nos encontramos a una distancia considerable, pues como el mismo término Integración Digital nos señala que haciendo uso de algún protocolo podemos conectar dos o más sistemas, en este caso podemos hacer uso de Internet con ayuda del estándar Ethernet, así encontrándonos desde cualquier punto del mundo podemos tener acceso al control del hogar.

Conforme avanzamos en esta "era digital" muchos términos han ido surgiendo, entre ellos, la ya mencionado Integración Digital y de igual manera podemos mencionar Convergencia Digital, aunque ambos términos parecen ser lo mismo, las definiciones de ambos nos muestran lo contrario. Existe actualmente un sinfín de términos que engloban lo que algunos expertos llaman "nuevo mundo", pero en este trabajo únicamente se hablara de los dos ya mencionados.

Hace más de 100 años, Henry Ford innovó en la industria automotriz creando una línea de ensamblaje, conectando todas las tareas y procesos que conformaban la construcción de un automóvil a través de una banda transportadora automatizada.

Evidentemente, gracias a este avance se logró disminuir el tiempo de producción y se automatizaron cientos de procesos. Algo semejante ocurre actualmente gracias a la Integración Digital, diversas empresas se han ido integrando a esta nueva idea para conseguir el control y monitoreo total de la empresa (al menos en producción) haciendo uso de dispositivos de adquisición de datos. Tener los datos de los dispositivos encargados del ensamblaje o empaquetado del producto de alguna empresa en tiempo real nos da la posibilidad de precisar si puede o no existir alguna falla en el mecanismo y de esta manera solucionarlo antes de que ocurra un problema mayor.

El término también es ocupado en diferentes áreas, no sólo en la ingeniería, pero la mayoría habla sobre la forma en que la sociedad se une más y más a los dispositivos digitales. Al menos así lo ven en las ciencias sociales. Obviamente nos enfocaremos al aspecto referido en la Ingeniería.

3.1 Convergencia digital

Definimos convergencia como el punto donde dos o más líneas se unen, y digital, como ya se había mencionado, es un dispositivo capaz de suministrar los datos mediante dígitos o elementos finitos o discretos.

La convergencia digital es la capacidad de tener acceso a la misma información desde diferentes dispositivos, ejemplificando es el hecho de poder consultar el correo electrónico desde una computadora, un celular o inclusive una televisión inteligente, prácticamente al mismo tiempo. Es contar con el acceso a la información desde cualquier medio, a eso se refiere la convergencia digital. De poco en poco el mundo ha ido cambiando su panorama sobre como compartir la información, pero es mucho más complicado para un usuario tener que recurrir a ciertos dispositivos para obtener información específica, en cambio se facilita más este evento al

acceder a los mismos datos desde cualquier dispositivo. Obviamente para lograr esto se debe contar con un protocolo, como el visto en la Integración Digital, y los medios de visualización de datos deben ser compatibles. No es tan simple como parece.

Como ya se analizó anteriormente, el uso de dispositivos analógicos solía ser muy complicado de esta forma fue evolucionando hasta llegar a los sistemas digitales gracias a la creación del transistor, los usuarios siempre han sido demasiado exigentes y es por ello que continuamente se actualizan los dispositivos, por ejemplo, antiguamente las cámaras solían ser muy grandes y estorbosas, pero con la innovación digital, ahora todo cabe dentro de un sensor que no solo da la posibilidad de captar una imagen, sino que también puede ser utilizada para grabar video, esto solo por mencionar un caso, pero no conformes con la miniaturización, los usuarios querían poder acceder a sus recursos desde los diferentes puntos del globo. Haciendo uso de la red de Internet se pudo satisfacer la necesidad del usuario, supongamos el siguiente caso; nos encontramos viendo una película en la computadora de escritorio, pero por uno u otro motivo debemos salir, es ahí donde entra el concepto de Convergencia Digital, la película al estarla visualizando en la red, nos permite dejar la computadora de escritorio y pasarnos a una tablet o un celular para continuar con la reproducción multimedia, seguir viendo la película es tan sencillo como iniciar la sesión personal en otro dispositivo y continuar disfrutando.

Imagen 3.1 Convergencia Digital. [B27]

Personalmente me parece fascinante no solo el concepto, también el cómo se logra realizar toda esta unión entre los dispositivos y el como ayuda a la sociedad a mantenerse comunicada y conectada. Entiendo que esto también tiene sus repercusiones pues obliga a los usuarios a mantenerse en lo que coloquialmente se conoce como “EN LINEA”. Cada paso que se da en la era digital, nos aleja más de la humanidad y la convivencia personal. Pero en este trabajo lo que menos busco es criticar los nuevos elementos de innovación que van surgiendo, aquí busco acercar un poco al lector a nuevos términos que cada vez se harán más frecuentes con el pasar de los años.

3.2 Kits de electrónica actuales

En el capítulo pasado, vimos cómo fueron los inicios en el cómputo, la programación y la electrónica; como el avance tecnológico nos brindó los microcontroladores y microprocesadores además de aprender cómo eran cargados y almacenados los programas dentro de ellos.

Con la llegada del internet y el acceso a una cantidad ilimitada de información, podemos ocupar cualquier navegador para explorar en el mundo de la red la existencia de los kits de electrónica, lo cual nos arrojará resultados acerca de productos que contienen varios componentes para realizar circuitos eléctricos con elementos pasivos y algunos cables. En esta parte del trabajo me enfocaré a algunos sistemas embebidos. “Un sistema embebido es un dispositivo controlado por un procesador, dedicado a realizar una única tarea o una serie de ellas” (Benchimol, 2011).

Actualmente existen demasiados kits de electrónica, algunos son más conocidos que otros, y ¿Cómo son? O ¿Cómo lucen? Bueno pues los kits actuales prácticamente ya cuentan con todo, son el conjunto de un microprocesador o microcontrolador montado en una placa, semejante a una motherboard (“Placa cuya función es interconectar todos los demás dispositivos dentro de una computadora y fuera de ella. Su rendimiento es crucial, su elección es clave y su calidad es definitoria”)*, pero esta contiene la característica de hacer mucho más fácil la interacción con el usuario, pongamos el ejemplo de un Arduino.

3.2.1 Arduino

“Arduino es una plataforma electrónica de código libre basada en hardware y software fácil de usar”, así es como nos lo presentan en su página oficial** y es verdad; tanto es fácil de usar como la programación es bastante sencilla, pero ¿Qué lo hace ser un kit de electrónica?, fácil la interacción de la placa con componentes (obviamente electrónicos) por medios de sus interfaces. Hagamos esto más gráfico.

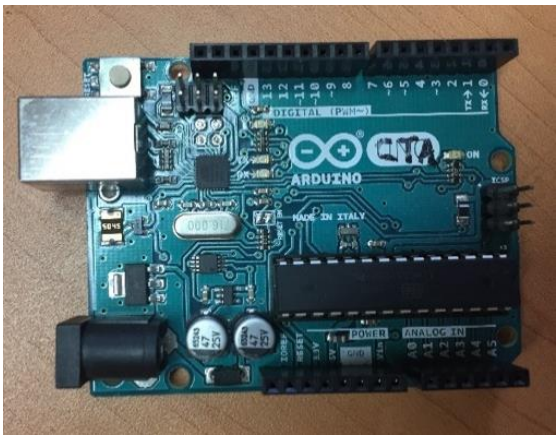


Imagen 3.2 Arduino UNO, vista superior. [A12]

En la **Imagen 3.2** podemos ver los componentes que conforman esta placa (Arduino UNO).

*: Richarte, J. (2018). Motherboard: partes y funcionamiento. En Servicio Técnico 03: Motherboard: partes y funcionamiento: curso visual y práctico: PCS • Notebooks • Redes • Mobile • y más (p. 2). Argentina: Redusers.

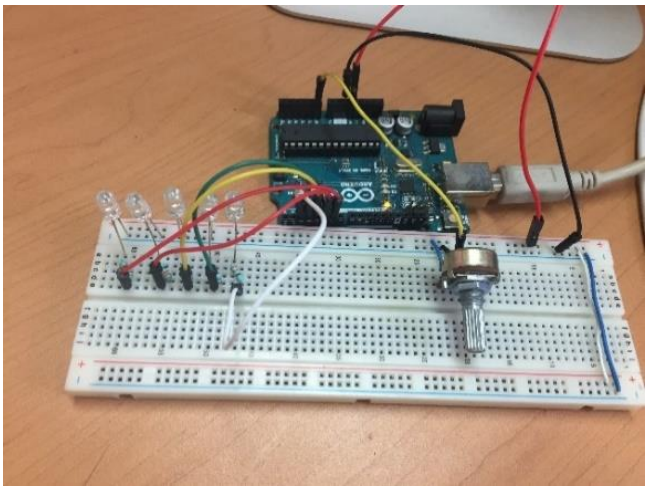
**: *“https://www.arduino.cc/en/guide/introduction”*

Tenemos el microcontrolador ATMEGA328P-PU, entradas y salidas digitales (pin 2- pin 13), TX y RX (pin 0 y pin 1), entradas analógicas (pin A0- pin A5), salidas de voltaje (5v y 3.3v), pin de reset y IOREF* además de tener interfaces AREF**. Otros dos elementos que faltan por mencionar son el puerto serial para cargar el programa y la alimentación.

Dependiendo de la placa a utilizar varían las interfaces incluidas en la placa por ejemplo la diferencia descomunal entre la placa Arduino UNO y la placa Arduino Pro Mini, donde Pro Mini es obviamente más pequeña, tiene 14 entradas y salidas digitales, 6 entradas analógicas botón de reset, procesador ATmega328 y los respectivos agujeros para los pines (**Imagen 3.3**). Cada placa es diferente y posee sus propias especificaciones.

Imagen 3.3 Arduino Mini Pro. [B28]

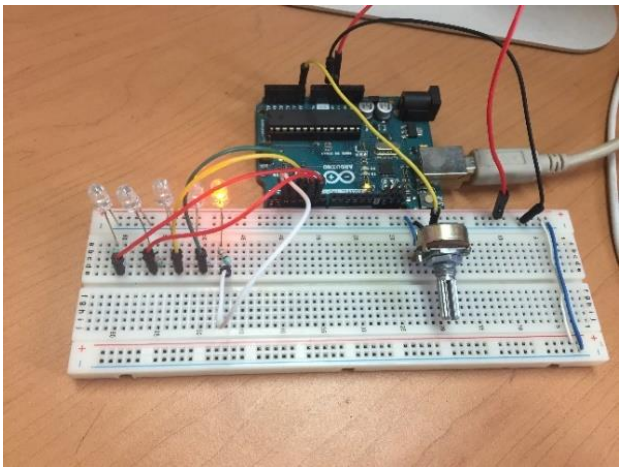
Veamos cómo funciona el Arduino UNO con un ejemplo simple:



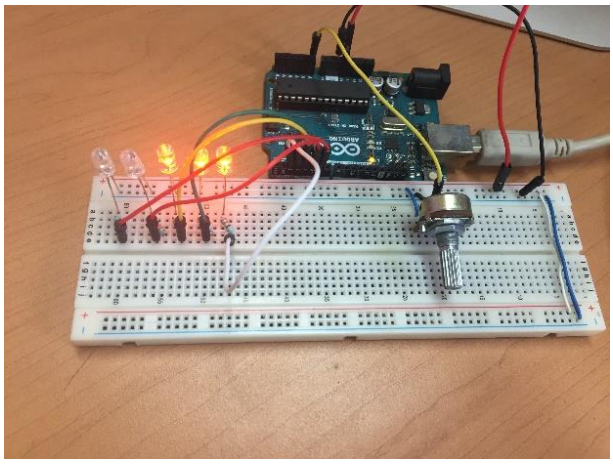
[A13].

*: *"Input Output Reference es el pin que nos suministra la tensión para el estado alto de los pines digitales"*(del Valle, L.. (2015). 26. Arduino entradas y salidas. enero 4, 2018, de PROGRAMAR FÁCIL Sitio web: https://programarfacil.com/podcast/26-arduino-entradas-y-salidas/)

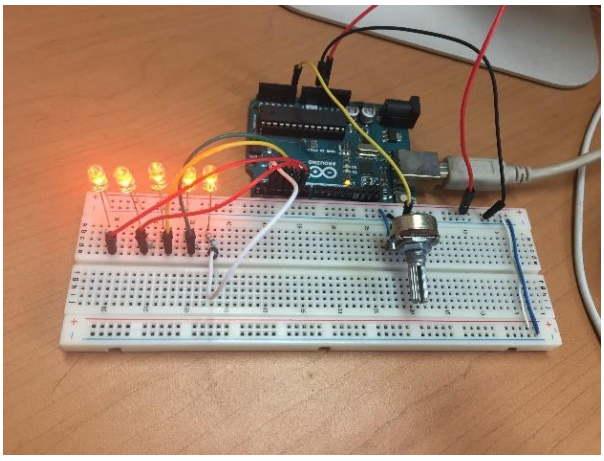
**: "*Analog Reference es el pin que nos suministra la tensión para el rango máximo de los puertos analógicos.*"*(*del Valle, L.. (2015). 26. Arduino entradas y salidas. enero 4, 2018, de PROGRAMAR FÁCIL Sitio web: https://programarfacil.com/podcast/26-arduino-entradas-y-salidas/*)*



[A14]



[A15]



[A16]

El programa fue cargado al Arduino haciendo uso del puerto serial, con esto tenemos la lectura dada por el potenciómetro y su interacción con los leds. Esta placa es un kit de electrónica por su interacción con los componentes electrónicos tanto digitales como analógicos (en este ejemplo resistencias, leds y el potenciómetro). Y por parte de la programación tenemos el control de los elementos a ocupar gracias al microcontrolador ATMEGA.

El programa se muestra a continuación:

```
const int analogPin = A0; // Pin de lectura Analógica
int valor;                // Variable Valor
int position;             // Variable Position
int leds[6]={2,3,4,5,6};  // Arreglo con el número de los pines a ocupar como salidas

void setup() { // Inicialización de elementos de interacción física
  pinMode(leds[0],OUTPUT); // Elemento 0 del arreglo leds como salida PIN 2
  pinMode(leds[1],OUTPUT); // Elemento 1 del arreglo leds como salida PIN 3
  pinMode(leds[2],OUTPUT); // Elemento 2 del arreglo leds como salida PIN 4
  pinMode(leds[3],OUTPUT); // Elemento 3 del arreglo leds como salida PIN 5
  pinMode(leds[4],OUTPUT); // Elemento 4 del arreglo leds como salida PIN 6
}
void loop() { // Loop. Elemento cíclico cargado en el microcontrolador

  valor = analogRead(analogPin);   // Leemos el valor mandado por el pin A0 y lo
asignamos a valor
  position = map(valor, 0, 1023, 0, 100); // Se mapea el valor obtenido para tenerlo
en forma de porcentage
  if(position>=80){   //Si el potenciómetro está al 80% o más
    prender(5);       //Llamada al método prender
    }
  if(position>=60 && position<80){ //Si el potenciómetro está al 60% o más y es menor
al 80%
    prender(4);                   //Llamada al método prender
    }
  if(position>=40 && position<60){   //Si el potenciómetro está al 40% o más y es
menor al 60%
    prender(3);                    //Llamada al método prender
    }
  if(position>=20 && position<40){   //Si el potenciómetro está al 20% o más y es
menor al 40%
    prender(2);                    //Llamada al método prender
    }
  if(position>=10 && position<20){   //Si el potenciómetro está al 10% o más y es
menor al 20%
    prender(1);                    //Llamada al método prender
    }
  if(position <10){ //Si el potenciómetro está al 10% o menos
    apagaTodo();    //Llamada al método apagaTodo
    }
  delay(500); //Espera 500 milisegundos para comenzar de nuevo
void prender(int maximo){ // Método Prender que recibe un numero entero
  int i=0;                // Inicializar la variable i en 0
  for(i;i<maximo;i++){     // Comienza el ciclo desde 0 hasta el número enviado al
método -1
    digitalWrite(leds[i],HIGH); // Elemento del arreglo leds en la posición dada por
i se enciende
    }
  if(maximo<5){           // Si el número enviado al método es menor a 5
    int aux=maximo;       // Inicializamos una variable auxiliar "aux" con el valor
enviado al método
    while(aux<5){         // Mientras la variable auxiliar sea menor a 5
      aux++;              // "aux" aumenta en 1
      digitalWrite(leds[aux],LOW);  // El elemento del arreglo leds de la posición
dada por el valor de "aux" se apaga
      }
    }
  }
void apagaTodo(){ // Método apagaTodo
  int tam=sizeof (leds) / sizeof (int); //Se calcula el tamaño del arreglo leds
(cantidad de elementos)
  int i=0;                              // Inicializar la variable i en 0
```

```
  for(i;i<tam;i++){                    // Comienza el ciclo desde 0 hasta el tamaño
del arreglo leds - 1
    digitalWrite(leds[i],LOW);        // El elemento del arreglo leds de la posición
dada por el valor de "i" se apaga
  }
```

Como se puede notar, se trata de lenguaje C en programación imperativa. “Como su nombre lo indica, implementa los programas como una serie de órdenes que la máquina tiene que cumplir. Es el tipo de programación más extendida y los lenguajes de programación imperativa suelen ser de ámbito general” (Vazquez, Gomez, Martín, Molinero, 2006) contamos con sentencias de control como el if y sentencias ciclicas como el for y el while. Hace mucho tiempo programar esta sencilla rutina hubiera tomado mucho más tiempo y más código, sin mencionar la forma de “alambrar” el circuito y el cargar el programa al microcontrolador. Los kits de electrónica actuales como lo es el Arduino nos facilita muchas cosas y hace más amigable la interacción con el usuario. Gracias al diseño de la placa tenemos una mayor facilidad al momento de manipular los pines de comunicación con otros elementos electrónicos haciendo uso de Jumpers (**Imagen 3.4**) o alambre. Y por parte de la programación, la interfaz de desarrollo utilizada es bastante accesible y fácil de entender, además cuenta con algunos ejemplos para los primerizos, además es con esta que podemos cargar el programa al microcontrolador. Cada placa de Arduino tiene su propia forma de cargar el programa, desde el IDE se debe hacer la selección del modelo al cual se cargará el programa, para que el compilador lo traduzca al lenguaje máquina.

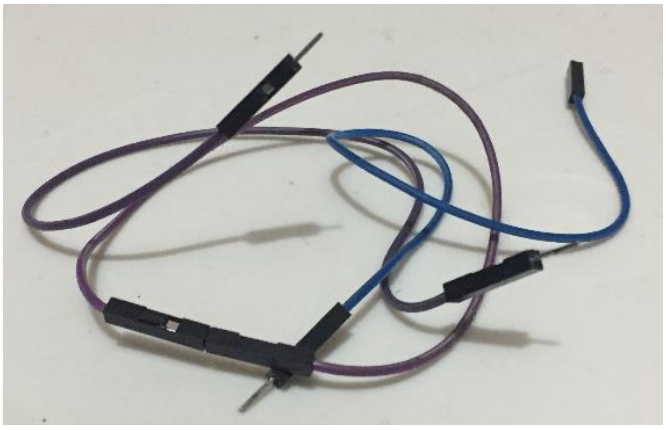


Imagen 3.4 Jumpers. [A17]

El kit de electrónica Arduino fue inventado en el año 2005 dentro del instituto IVRAE por Massimo Banzi.

El proyecto comenzó para fomentar el aprendizaje en alumnos de cómputo y electrónica del mismo instituto. La primera versión de Arduino estaba basado en una simple placa de circuitos, un micro controlador simple y resistencias, este solo podía interactuar con elementos simples como leds. Además aún no se contaba con un lenguaje de programación para su manipulación.

Fue con el pasar de los años que poco a poco se fueron agregando más y más elementos al proyecto hasta lograr crear la placa que ahora todos conocemos. Pero el Arduino “UNO” el cual fue utilizado en el ejemplo pasado, no es la única versión de él.

Se cuentan con otras placas programables para diversos usos desde prácticas estudiantiles hasta sistemas de IoT*, estos los podemos encontrar en la página oficial.

Pero Arduino UNO y todos sus hermanos sólo son algunos de los kits de electrónica existentes, comencé con este pues es uno de los más conocidos y utilizados en el mercado, sin mencionar que gracias al Laboratorio de Cómputo del Centro Tecnológico, de la Facultad de Estudios Superiores Aragón tengo acceso a esta tecnología.

Impresionantemente Arduino es la empresa que se encuentra acaparando el mercado. Pero eso no quiere decir que sea la única, MicroCHIP es otra de ellas, famosa por el desarrollo de PIC e Intel, la cual nos provee de una placa nombrada Galileo**. Estas tres empresas son de lo más conocido en cuanto a microcontroladores, sistemas embebidos y sistemas de IoT, sin mencionar que Intel es uno de los más grandes proveedores de microprocesadores en el mundo.

Los PIC de MicroCHIP son utilizados desde hace décadas, muy interesantes pero un tanto más complicados de utilizar, pues no vienen montados en una placa que reduzca el trabajo de alambrado y carga del programa.

3.2.2 Galileo

Por otra parte, la placa Galileo está basada en Arduino, por ello su parentesco fisico; entradas y salidas digitales, entradas análogas, voltajes, tierras etc. Con la diferencia más notable que Galileo es más grande en comparación al Arduino UNO, casi del tamaño de un Arduino MEGA.

Imagen 3.5 Arduino Mega. [A18]

*: Internet of Things. Internet de las cosas, mantener cosas de uso cotidiano enlazadas en la red para consultas, monitoreo o control.

**:Intel posee diversas placas pero nos centraremos en Galileo, pues es a la que tengo acceso.

Hace uso de la arquitectura de Intel utilizando un microprocesador Intel® Quark™ SoC X1000 (caché de 16 K, 400 MHz) montado en un zócalo Quark 393pin FC-PGA* (**Imagen 3.6**) por lo tanto es compatible con algunos sistemas operativos, el cual deberá ser cargado a una micromemoria SD y cargado en el Galileo desde la ranura indicada. Físicamente contamos con un puerto Ethernet 10/100MB, puerto USB 2 (Host Port), microUSB (Client Port), conexión a toma corriente y un puerto JTAG** (**Imagen 3.7**).

Imagen 3.6 Zocalo y procesador de Galileo. [A19]

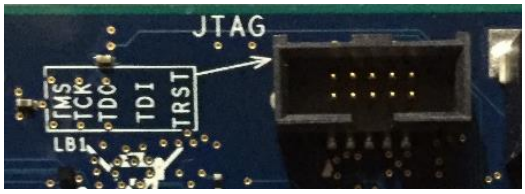


Imagen 3.7 JTAG de Galileo. [A20]

Imagen 3.8 Galileo vista superior (lado izquierdo), vista inferior (lado derecho). [A21]

*: "Flip Chip-Pin Grid Array, es un microchip diseñado por Intel para los microprocesadores más rápidos, donde la parte que se calientas más se encuentra alejada de la placa base."

"Talukdar, M.. (2014). Dictionary of Computer & Information Technology. New Delhi: Prabhat Prakashan."

**: "Es una parte esencial del proceso de depuración de subsistemas externos en la computadora"

(Catsoulis, J.. (2005). Designing Embedded Hardware: Create New Computers and Devices. United States of America: O'Reilly Media, Inc.

Galileo posee un nivel de procesamiento mayor a lo que cualquier placa Arduino común pueda otorgar, por ende su costo es igual de elevado, pero su funcionalidad es sorprendente. No sólo se enfoca en cargar un software y hacerlo funcionar con su respectiva interacción a los dispositivos electrónicos conectados, esta placa tiene la posibilidad de contener un sistema operativo Linux en él, podemos hacer de este un servidor web que interactúe con elementos de electrónica, por ejemplo, una página web que prenda una serie de leds. Personalmente eso se me hace sumamente interesante y un sistema que podemos explotar fácilmente.

También nos da la opción de no usar un sistema operativo, simplemente podemos cargar el programa desde el IDE de Arduino (son completamente compatibles, solo se necesita descargar el driver indicado), alambrar y probar, como se hace con cualquier placa de Arduino.

Ahora, ya sabemos que las placas comunes de Arduino manejan microcontroladores y Galileo maneja un microprocesador. A nivel de procesamiento, Galileo gana por velocidad y cálculos. Pero los microcontroladores Arduino, no requieren de una memoria física sabemos que un microcontrolador posee una memoria interna, donde almacena la rutina a ejecutar y las variables implicadas. Ambos sistemas son aplicables en diversos proyectos, por su fácil manejo y excelente rendimiento, cada vez se ven más de estos productos en la vida cotidiana.

Desde un robot que arme un Cubo de Rubik hasta brazos completamente controlados por Arduino, en el caso de Galileo, el acercamiento al termino Internet de las Cosas es sorprendente. Tener conectado hasta el refrigerador a un sistema de compras sería sumamente interesante.

Por otra parte, existen más dispositivos además de estos dos, es turno de hablar un poco acerca de Raspberry Pi.

3.2.3 Raspberry Pi

Raspberry Pi 3 es el centro de este documento, en capítulos siguientes se hablara más a detalle de él. Raspberry es un tipo de placa conocida como Single Board Computer (SBC) o Computadora de Una Placa, contiene todos los elementos de una computadora común y corriente en una simple y pequeña tarjeta no mayor a 9 cm x 5cm aproximadamente. Impresionante las dimensiones para todo aquello que puede realizar. Cuenta con 40 pines para comunicación externa con elementos de electrónica semejantes a los implementados en Arduino UNO o Mega (aunque estos son hembras y en Raspberry machos).

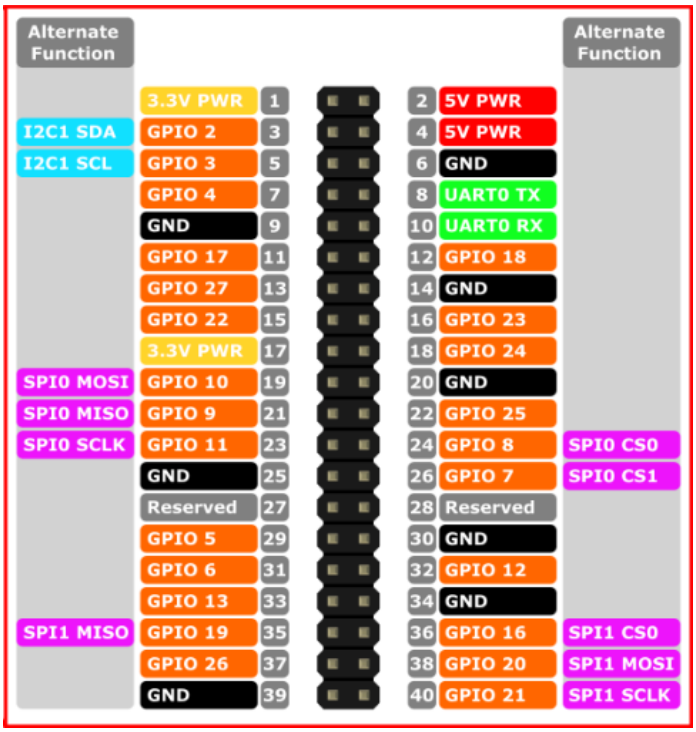


Imagen 3.9 Pines de salida y entrada de Raspberry 2 modelo B y 3 modelo B. [B29]

En la **Imagen 3.9** podemos ver gráficamente la distribución de los pines en la Raspberry (PIN 39 y 40 son los más cercanos a los puertos USB) sólo que en este dispositivo tendremos algunos pines representados con el acrónimo GPIO (General Purpose Input/Output) los cuales pueden ser ocupados de diversas formas. Esta tarjeta cuenta con 1, 2 o 4 puertos USB dependiendo del modelo, un puerto de Ethernet 10/100 en todos los modelos excepto Raspberry pi 1 Modelo A, Jack de 3.5mm para salida de audio, Raspberry Pi 3 cuenta con conectividad a WiFi y Bluetooth, puerto de alimentación Micro USB de 5v, Conector RCA (PAL y NTSC), HDMI, Interfaz DSI para panel LCD, bus de conexión para una cámara especial de Raspberry, en modelos 1 A la memoria es de 256 MiB (compartidos con la GPU), en modelos 1 B y 1 B+ 512 MiB (compartidos con la GPU) y en modelos 2 B y 3 B 1 GB (compartidos con la GPU).

Ahora lo más importante en los modelos 1 A, 1 B Y 1 B+ es el procesador que los conforma es un ARM 1176JZF-S a 700 MHz (familia ARM11), el modelo 2 B cuenta con un quad-core ARM Cortex A7 a 900 MHz y en el modelo 3 B tenemos quad-core 64-bit ARMv8 a 1.2GHz, por parte del almacenamiento externo solo Raspberry pi 1 A y 1 B son compatibles con SD / MMC / ranura para SDIO, los modelos 1 B+, 2 B Y 3 B tienen la ranura para MicroSD esto para la carga del sistema operativo (LINUX).

También existen los modelos Zero y Zero W, el modelo Zero es más pequeña que los nuevos modelos, con 512 MB en memoria RAM, procesador ARM1176 a 1 GHz y solo un núcleo, en lugar de HDMI posee un MiniHDMI y sustituyendo los puertos USB solo tiene 2 MicroUSB, uno para alimentación y otro para datos. En Zero W, es completamente lo mismo sólo con un ligero cambio, posee WiFi y Bluetooth.

Raspberry fue un proyecto que nació en la universidad de Cambridge, como una forma de enseñar cómputo y electrónica a los niños, inicialmente se utilizó un microcontrolador ATMEGA 644. El fundador de la fundación Raspberry es Eben Upton que con ayuda de David Braben pudieron sacar adelante este magnífico proyecto, llegando a escuela e impulsando el aprendizaje en los niños, mostrándoles una nueva forma de ver la computación y la electrónica "ábrelo y mira cómo funciona" (Universidad Politécnica de Valencia, 2013).

Con Arduino, simplemente diseñamos el programa y lo cargamos a la placa, realizamos las conexiones (alambrado) necesarias y probamos. En Galileo tenemos la posibilidad de recurrir al sistema operativo haciendo uso de SSH*, conectamos con la placa y manipulamos el sistema, únicamente haciendo uso de una terminal pues no posee interfaz gráfica o bien programando desde el IDE de Arduino como ya se mencionó con anterioridad. Pero en el caso de Raspberry, es obligatorio el uso de un sistema operativo, ya sea en su versión ligera (solo terminal) o completa (con Interfaz Gráfica), es básicamente y a grandes rasgos una computadora con posibilidades de interactuar con los elementos de electrónica que queramos. Facilidades de una computadora, interfaz gráfica completa y simple de usar, posibilidad de programar en ella con cualquier IDE compatible, interacción con los GPIO, un sinfín de posibilidades en el campo del desarrollo gracias a esta placa.

De los microcontroladores tipo PIC, pasamos a circuitos impresos con un microcontrolador montado como lo es Arduino, de ahí cambiamos el microcontrolador por un microprocesador tal cual se hizo con Galileo, y terminamos hasta el momento con sistemas SBC que es el caso de Raspberry. Pero estos no son los únicos kits de electrónica, existen muchos más. Quise hacer énfasis en estos pues ya he tenido la posibilidad de trabajar con anterioridad en ellos, algunos otros que podemos encontrar en el mercado son:

- BeagleBone
- Pandaboard
- Tinker Board de Asus
- Gumstix's
- Banana Pi
- Orange Pi

Las diferencias más notables entre todas estas placas, es el procesador, y la memoria RAM. Donde Raspberry tiene un procesador Quad Core a 1.2 GHz y una memoria RAM de 1GB, Tinker Board gana pues posee un procesador RK3288 Quad Core a 1.8GHz y una memoria RAM de 2GB. Y el precio entre ambos productos no es tan diferente, desde los $1,000.00 M/N hasta los $1,600.00 M/N máximo.

Cada proyecto donde se quiera hacer uso de un kit de electrónica debe ser bien estudiado para saber qué sistema embebido deberá ser utilizado, aunque la mayoría tiene la misma funcionalidad, otros tienen un "plus" como interfaces gráficas, mayor cantidad de puertos, usos de sistemas operativos o incluso un pinout diferente.

*: Secure Shell. Protocolo de comunicación seguro, donde los datos enviados son encriptados hasta llegar a su destino. Utiliza la arquitectura cliente/servidor.(Barrett, D.,Silverman, R., & Byrnes, R.. (2005). What Is SSH?. En SSH, The Secure Shell: The Definitive Guide: The Definitive Guide(pp. 1-2). United States of America: O'Reilly Media, Inc.)

Conociendo algunos kits de electrónica, veamos dos de los campos en donde se pueden utilizar.

3.3 Domótica

Cuando hablamos de domótica, lo primero que se nos viene a la mente es hogares inteligentes, una casa donde prácticamente no debas hacer mucho, pues todos los aparatos electrodomésticos sabrán que hacer y cuando hacerlo. Esta idea no se aleja mucho de la realidad, pero tomaremos la definición dada por la Fundación Privada Institut Cerdà:

"vivienda domótica es aquella en la que existen agrupaciones automatizadas de equipos, normalmente asociados por funciones, que disponen de la capacidad de comunicarse interactivamente entre ellas a través de un bus doméstico que las integra". (Junestrand, Passaret, & Vázquez, 2004).

Básicamente tener dispositivos del hogar conectados entre sí para transferencia de datos o manipulación de ella a través de algún protocolo, aquí entra el concepto de Integración Digital previamente revisado.

3.3.1 X-10

Ahora bien, podemos decir que la Domótica comenzó entre 1976 y 1978 con la invención del sistema X-10 desarrollado por ingenieros de Pico Electronics Ltd en Escocia. Aunque este sistema no es del todo fiable, proporciona automatización en algunos elementos a un precio muy razonable, esta es una de las mayores razones de su éxito, tanto el precio como la facilidad de instalación. El protocolo X-10 consta del envió de datos haciendo uso de corrientes portadoras, dicho en otras palabras, enviar los datos por medio de la línea eléctrica, al hacer uso de la corriente eléctrica se accede fácilmente a los dispositivos conectados a ella. Es por eso que la instalación de los sistemas X-10 era sumamente sencilla.

En su lanzamiento al público norteamericano el sistema X-10 contaba con un módulo para lámpara, un módulo para electrodomésticos y su respectiva consola de control para el envío de señales, esto se presentó como "Sistema de Control del Hogar" de Sears y "Plug\'n Power" de Radio Shack. Sus actualizaciones fueron un módulo para interruptores de pared y un reloj automático.

Cada componente tiene su propio código, el emisor manda la señal en forma de datagrama y el receptor obtiene esa señal, posteriormente procede a ejecutar la orden marcada en dicho datagrama. Un sistema de control básico, mandas una señal, se recibe y se ejecuta. La transmisión de información se lleva a cabo con la ayuda del código binario. Como el envío se hace a partir de la corriente eléctrica el "1" lógico lo da un pulso de 120Khz en duración de 1 ms*, el "0" lógico es obviamente la ausencia de tal pulso.

*: Milisegundo

El datagrama está compuesto por tres bloques, código de inicio, código de casa y código de dispositivo o de función. El código de inicio consta de 4 bits, este avisa a los dispositivos X-10 que una orden se está enviando. Su código binario es 1110. Para saber a cuál dispositivo va dirigida la orden, se hace uso del código de casa, este también está compuesto por 4 bits, podemos ver los códigos en la Tabla 1.1. En el código de dispositivo o función se deben enviar 5 bits, el ultimo bit marcara la diferencia entre un datagrama de dispositivo o de comando. En caso de tratarse de una orden a ejecutar el último bit será 1 y en caso de ser de dirección será 0. Los otros 4 bits contendrán la orden codificada o el dispositivo. Los códigos en ambos casos pueden verse en la Tabla 1.1 y Tabla 1.2.

Código de casa	Código de Dispositivo	Valor Binario	Valor Decimal
A	1	0110	6
B	2	1110	E
C	3	0010	2
D	4	1010	A
E	5	0001	1
F	6	1001	9
G	7	0101	5
H	8	1101	D
I	9	0111	7
J	10	1111	F
K	11	0011	3
L	12	1011	B
M	13	0000	0
N	14	1000	8
O	15	0100	4
P	16	1100	C

Tabla 1.1 [C1]

Función	Valor Binario	Valor Decimal
Todas las unidades apagadas	0000	0
Todas las luces encendidas	0001	1
Encender	0010	2
Apagar	0011	3
Atenuar intensidad	0100	4
Aumentar intensidad	0101	5
Apagar todas las luces	0110	6
Código extendido	0111	7
Petición de saludo	1000	8
Aceptación de saludo	1001	9
Atenuación preestablecida (1)	1010	A
Atenuación preestablecida (2)	1011	B
Datos extendidos (analógico)	1100	C
Estado = on	1101	D
Estado = off	1110	E
Petición de estado	1111	F

Tabla 1.2 [C2]

3.3.2 KNX

Actualmente el sistema X10 sigue siendo una potencia en el mercado, siguen vendiéndose productos e inclusive kits de domótica X10.

Aunque X10 dio el arranque para iniciar con las nuevas tecnologías ligadas a la Domótica, muchas más se abrieron paso en este campo, tal es el caso de KNX.

La asociación KNX comenzó a surgir en el año de 1997 gracias a la unión de 3 grandes sistemas de automatización, inicialmente conocido como la Asociación Konnex terminó adoptando el nombre de "Asociación KNX". Los tres sistemas implicados son BatiBUS Club International, European Home System Association y European Installation Bus Association.

- BatiBUS es un protocolo abierto fabricado por la empresa Merlin, eso quiere decir que cualquier empresa puede hacer uso de él. Su funcionamiento se basa en la técnica CSMA-CA*, muy parecido a Ethernet con una gran ventaja, la solución a colisiones. Debe discernir dependiendo de la prioridad que tiene cada instrucción. Este estándar consta que todos los elementos implicados permanezcan a la escucha de instrucciones, y sólo a quien va dirigida la instrucción es quien la realiza, es como estar en una clase y de un momento a otro el profesor le hace una pregunta a un alumno, a lo que el alumno deberá responder a dicha pregunta pero todos los demás escucharon la pregunta. Algo parecido funciona este protocolo.
  Para la identificación de dispositivos, hace uso de una técnica semejante a la utilizada en X10. Cada uno posee su propio identificador.
- EHSA (European Home System Association) se encaró de la mejora y evolución de EHS, el cual es un estándar semejante a BatiBUS en lo que respecta a ser utilizado por cualquier empresa, o sea, es un protocolo abierto. Dicho sistema está basado en la topología del modelo OSI, haciendo uso de los niveles de aplicación, red, enlace y físico. Al igual que en X10 y BatiBUS, cada dispositivo contiene un su interior su respectiva dirección única para lograr ser identificado dentro del entorno de red y no sólo eso, también contiene información para el enrutado de los paquetes por los diferentes segmentos en la red.
- EIBA (European Installation Bus Association) se encuentra compuesto por líderes en el mercado eléctrico los cuales en 1990 decidieron impulsar el uso del sistema EIB. Este sistema buscaba contrarrestar la llegada de otros productos domóticos provenientes de Japón y Estados Unidos. Al igual que EHS, EIB hace uso del modelo OSI con una arquitectura descentralizada, en donde cada dispositivo conectado tiene su propio microprocesador. Una muy buena idea para tener un mejor control en el bus de datos.

*: "Carrier Sense Multiple Access, está diseñado para redes que comparten el medio de transmisión. Cuando una estación quiere enviar datos, primero escucha el canal para ver si alguien está transmitiendo. Si la línea se encuentra desocupada, la estación transmite. Si está ocupada, espera hasta que esté libre." (German´s Domain. (s.f.). Acceso por contención, aleatorio o no determinístico. Retrieved February 01, 2018, from http://docente.ucol.mx/al970310/public_html/CSMA.htm)

El modo de transmisión de datos en este estándar es muy flexible pues permite utilizar cualquiera de los 4 modos disponibles: Par trenzado (TP/TwistedPair), Ondas Portadoras (PL/Power Line), Radio Frecuencias (RF/Radio Frequency) e internet Protocol (IP). Admeas nos brinda tres opciones de configuración de las cuales ya sólo son disponibles dos: Modo-S (Instalacion Profesional / KNX SystemModel), Modo-E (Instalacion sencilla / KNX EasyMode), dadas estas facilidades se dice que con el estándar KNX podemos personalizar nuestra configuración y la forma de transmisión de datos. Al bus de comunicaciones se puede conectar tanto sensores como actuadores que permiten el control de todas las aplicaciones posibles como pueden ser: iluminación, sistema de seguridad, audio/video, control de bienes de gama blanca, entre otros.

Imagen 3.10 Donde se puede encontrar KNX. [B30]

La página oficial de KNX los describe cómo:

"*KNX Association es el creador y propietario de la tecnología KNX – el ESTÁNDAR mundial para todas las aplicaciones de control de la vivienda y el edificio, abarcando desde control de la iluminación y las persianas, así como variados sistemas de seguridad, calefacción, ventilación, aire acondicionado, monitorización, alarma, control de agua, gestión de energía, contador, así como electrodomésticos del hogar, audio/video y mucho más. KNX es ESTÁNDAR mundial para el control de la vivienda y del edificio con una única herramienta de puesta en marcha (ETS), independiente del fabricante, y cuenta con una completa gama de medios físicos (TP, PL, RF y IP), así como de modos de configuración soportados (sistema y modo fácil). KNX es un estándar aprobado a nivel europeo (CENELEC EN 50090 y CEN EN 13321-1) e internacional (ISO/IEC 14543-3).*" (https://www.knx.org/mx/).

La transmisión de datos de este protocolo se hace mediante bits y por telegrama. Un telegrama está compuesto de bits. Como ya sabemos un bit se representa por 1 si hablamos de un nivel de corriente alto (por lo general entre 3.3 y 5 volts) y 0 en ausencia de la misma. Al momento que un sensor es activado, el acoplador del dispositivo genera y envía un telegrama al bus. Los bits se envían carácter por carácter, teniendo en cuenta que un carácter está formado por 8 bits (1 byte) el envío de cada carácter requiere de 13 impulsos de bit:

1. Bit de Arranque (1 bit)
2. Bits de Carácter (8 bits)

3. Bit de Paridad (1 bit)
4. Bit de Parada (1 bit)
5. Bits de retardo (2 bits)

Ahora bien, la longitud de un telegrama (n caracteres) depende del punto de tipo de dato, en otras palabras, de los datos útiles que informan acerca del evento ocurrido, por lo general los datos útiles van entre los 2 bytes de conmutación y los 16 bytes de transmisión de texto. Los datos específicos siempre ocupan 7 bytes:

1. 1 byte con el campo de control.
2. 2 bytes con la dirección de origen.
3. 2 bytes con la dirección de destino.
4. 1 byte de comprobación.
5. 1 byte con la longitud de datos útiles, con el contador de ruta y con el tipo de dirección de destino.

"*La tecnología KNX está respaldada por la KNX Association, una agrupación de empresas líderes activas en diversos campos relacionados son el control de viviendas y edificios. Actualmente, la KNX Association tiene más de 370 empresas miembros, representando más del 80 % de los dispositivos para el control de viviendas y edificios vendidos en Europa.*"

La meta de estas empresas es promover el desarrollo e implementación de sistemas de control en viviendas o inclusive edificios.

Tanto X10 como KNX siguen formando parte del mercado actual, fomentando e impulsando el mundo de la integración digital aun en el ambiente del hogar.

La domótica, continúa en constante cambio, donde antes solo se buscaba prender una lámpara, el estéreo o televisor, ahora contamos con sistemas de seguridad, vigilancia, accesos, etc. Es impresionante el avance que se tiene al volver tu hogar un hogar digital.

3.4 Robótica

El tema de la robótica es sumamente extenso, no sólo hace uso de electrónica y programación, aquí se llevan a cabo un sinfín de cálculos en cuanto a la mecánica y cinemática.

El término Robot es originario del checo ROBOTA empleado por primera vez en la obra teatral de Karel Čapek "Rossum's Universal Robots", señalando con esta a un humanoide mecánico, el argumento de la obra fue acerca de una empresa que construía humanos artificiales orgánicos con el fin de reducir el trabajo para el resto de las personas, a partir de ese momento la palabra fue empleada en prácticamente todos los idiomas.

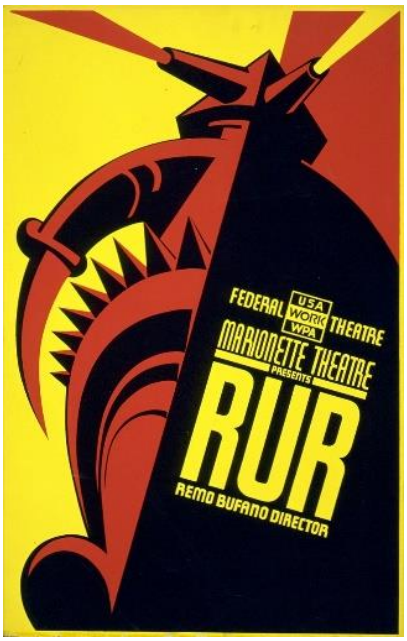


Imagen 3.11 Cartel promocional de la obra teatral. [B31]

En Ruso la palabra Robot significa “trabajo” lo que no esta tan alejado de su origen checo, lo cual significaba “trabajo forzado”.

El primer brazo manipulador fue desarrollado por Harold Roselund en colaboración con Willard Pollard dentro de la empresa “DeVilbiss” en el año de 1938. Fue utilizada para pintar con spray, debido a la insuficiencia tecnología y la falta de cómputo digital la tarea de programar dicho brazo era muy complicada, por lo tanto, no tuvo el éxito esperado.

Durante la segunda guerra mundial se incursiono más en el concepto de telepresencia, una idea diseñada para aparentar que una persona se encuentra presente en un punto indicado, cosa que es falsa pues se encuentra en otro lugar (hoy en día se hace mediante videoconferencias). El ingeniero Estadounidense Raymond C. Goertz teniendo este concepto presente diseño un manipulador maestro-esclavo conocido como “teleoperador”, consiste en un par de manos mecánicas controladas a distancia haciendo uso de encadenamientos mecánicos.

Imagen 3.12 Teleoperador. [B32]

Un gran escritor de ciencia ficción, que en lo personal admiro mucho, fue quien nos entregó el término robotics (robótica), Isaac Asimov no sólo nos entregó un nuevo tema de estudio sino que también publicó las tres leyes de la robótica:

“1- Un robot no puede hacer daño a un ser humano o, por inacción, permitir que un ser humano sufra daño.

2- Un robot debe obedecer las órdenes de un humano, salvo que alguna de éstas ente en conflicto con la primera Ley.

3- Un robot debe proteger su propia existencia, salvo que esto viole la primera o la segunda Ley.” (Zabala, 2007).

Fue en 1952 que George Devol patentó el primer brazo manipulador con memoria, capaz de mover herramientas de trabajo desde un punto a otro. Devol es considerado el “padre de los robots”. En 1960 Devol vende la patente de su invento a una empresa llamada Condec la cual comenzó inmediatamente la producción del robot Unimate en la subsidiaria Unimation. Ya para el año 1962 General Motors instalo su primer robot Unimation.

Desde entonces el campo de la robótica fue creciendo impresionantemente, tanto que para el año de 1974 se contaba con alrededor de 3500 robots en el mundo, obviamente todos aplicados a la industria, aun no se tomaba en cuenta la posibilidad de tener robots de servicio.

Grandes empresas comienzan a desarrollar robots industriales, pero es la NASA quien en septiembre de 1976 logra aterrizar en Marte el robot VIKING II, cuenta con un brazo robótico articulado.

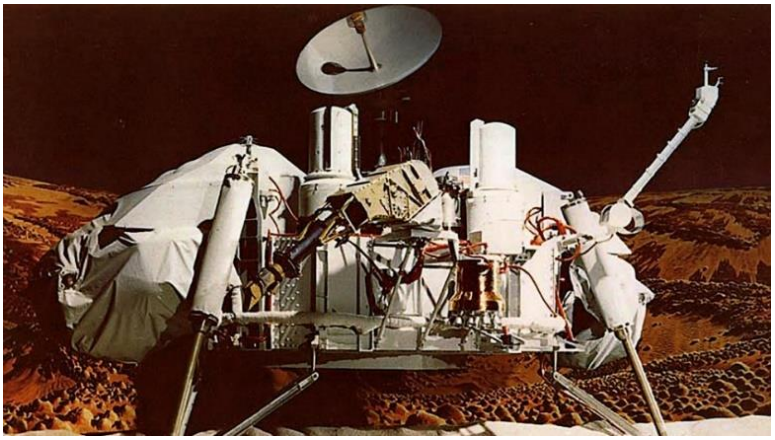


Imagen 3.13 Viking II. [B33]

Más ambiciosa aún se vio la empresa Japonesa Honda que en 1986 comenzó su proyecto para crear un robot humanoide, mostrando desde 1993 la conocida serie P que incluye el modelo P1, P2, P3 y P4 así como el conocido ASIMO, creado en el año 2001 y aun es actualizado constantemente.

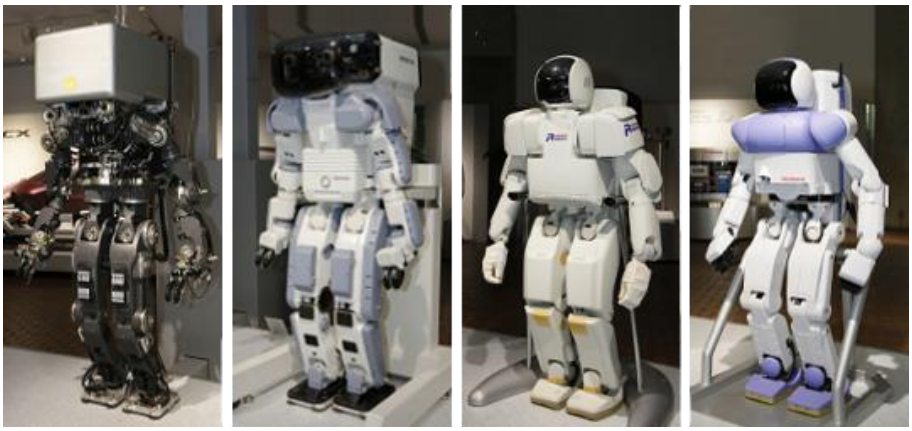


Imagen 3.14 Serie P de Honda (de izquierda a derecha P1 a P4). [B34]

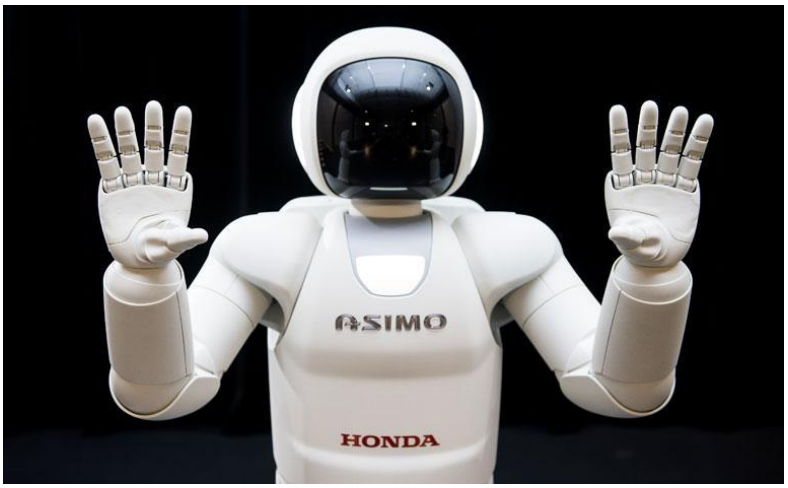


Imagen 3.15 Asimo de Honda. [B35]

La idea de la robótica se ha ido modificando, pues antes los robots eran únicamente aplicados a la industria, hoy en día es más aceptable la idea de tener robots para servicio. De hecho es más común ver la robótica aplicada aun en los juguetes. Cozmo es un ejemplo.

Creado por la empresa Anki, Cozmo es un robot capaz de expresar emociones haciendo uso de la inteligencia artificial programada en él. Posee una serie de motores para su desplazamiento y su interacción con el medio físico, reconoce rostros y nombres (semejante a un perro). Tiene expectativas muy altas en el mercado, pues es fácilmente comparado con el personaje Wall-E de la película con el mismo nombre dirigida por Andrew Stanton producida por Walt Disney Pictures y Pixar Animation Studios, además de su curiosa forma de hablar parecida a R2-D2 de la película StarWars. Cozmo también interactúa jugando, incluye una serie de cubos con los cuales el robot puede divertirse e inclusive competir contra una persona. La robótica puede encontrarse ya en prácticamente cualquier lugar.

Imagen 3.16 Cozmo de la empresa Anki. [B36]

Actualmente en el mundo se llevan a cabo torneos de robótica, donde niños de 8-13 años participan con sus propios robots, tuve la fortuna de participar en la Robocup del año 2012 efectuada en México, y me impresionó todo el potencial a nivel mundial en cuanto a la robótica, sobre todo en el campo de humanoides. En este torneo podemos apreciar partidos de futbol donde los jugadores son robots.

Fue aquí donde vi por primera vez un partido de robots NAO.

Aldebaran Robotics una subsidiaria de la empresa Softbank, es la responsable del diseño y creación del robot que lleva como nombre NAO. Capaz de reconocer rostros (al igual que Cozmo), sonidos, reconoce hasta 20 idiomas, identificar su entorno, reconocer formas, interactuar con personas. Es simplemente un robot humanoide capaz de realizar cualquier actividad con sus 25º de libertad en movimiento con una fluidez casi humana. Tiene acceso a internet mediante WiFi o Ethernet.

Imagen 3.17 Robot NAO de la empresa Softbank. [B37]

NAO es capaz de hacer prácticamente cualquier cosa según sea programado, gracias a sus dos cámaras, cuatro micrófonos, nueve sensores táctiles, dos sensores de ultrasonidos, 8 sensores de presión, un acelerómetro y un giróscopo.

Es programado si se desea usando NAOqi OS, un sistema operativo basado en Gentoo (LINUX). Cuenta con una guía rápida y manuales para su instalación disponibles en línea. Con sus múltiples bibliotecas, tenemos el control completo de NAO. Soporta los lenguajes, Java, Java Script, C++ y Python. O también tenemos la opción de utilizar Choreographe, un software instalable en casi todos los sistemas operativos, este software nos da la facilidad de crear animaciones, diálogos y comportamientos; sin mencionar el control y monitoreo del robot. Además del Sistema Operativo, el software Choreographe se puede hacer uso de Python SDK, C++ SDK, Java SDK, entre otros. Para más información se puede consultar la guía completa*, desde el desempaquetado del robot, hasta la programación de él.

Los sistemas arduino pueden ser ocupados para el diseño de prototipos, gracias a su fácil manejo y bajo costo.

Galileo es empleado para sistemas de IoT por su puerto de Ethernet ya incluido y al igual que arduino, su facilidad al momento de programar.

Raspberry es una gran herramienta de aprendizaje, basta con cargar el SO, (como lo veremos en el capítulo 5) para hacer uso de él. También es económico para todas las funcionalidades que nos ofrece, requiere de más tiempo para aprender a usarlo, pues no solo es cragar el programa y ejecutarlo, es instalar los servicios a ocupar, ejecutarlos y aplicar prueba y error.

Como podemos ver, la era digital ha ido evolucionando a grandes pasos y a una velocidad impresionante. Lo que antes era un simple sueño, ahora lo vemos tan común en el día, dando por hecho que vivimos en la era de la integración digital, recordando un poco, la integración digital nos habla sobre la manipulación de los datos a través de uno o más elementos, un elemento recibe los datos, los envía mediante algún protocolo a otro dispositivo y este los manipula para obtener un resultado o los envía a otro, y así sucesivamente hasta obtener el resultado deseado. Un robot hace básicamente eso; detecta haciendo uso de algún sensor, genera una decisión (empleando un programa) y realiza una acción.

Con esto podemos ver cada vez más cerca ciudades futurísticas como las imaginaba Isaac Asimov o Julio Verne.

4) Definición del problema.

La razón por la cual se buscó el desarrollo de este proyecto, fue principalmente para reducir los costos del sistema tanto de hardware como de software. El proyecto se dividió en tres equipos, compañeros de ingeniería mecánica, ingeniería eléctrica y mi respectiva área, ingeniería en Computación. Originalmente se contaba con un sistema basado en tecnología de la marca LabVIEW por parte de la compañía National Instruments.

Imagen 4.1 Logo oficial de LabVIEW. [B38]

LabVIEW es una herramienta de software que permite hacer la integración con hardware teniendo así la facilidad para realizar pruebas, monitoreo, control e inclusive realizar un análisis de datos de dicho hardware. LabVIEW nos facilita la programación con una modalidad de “clic y arrastre”, con esta utilidad se pueden generar interfaces y así visualizar lo obtenido por todo el hardware conectado. Esta herramienta de software utiliza dos interfaces, la primera es para generar la interfaz mostrada al usuario y la segunda para realizar todo el diseño del algoritmo a programar. LabVIEW utiliza como forma de programación los diagramas a bloques, facilitando así el diseño lógico del algoritmo, simplemente hay que buscar los elementos necesarios en el área adecuada para el diseño del programa, y utilizar el modo “clic y arrastre”, para así tener un programa completamente funcional.

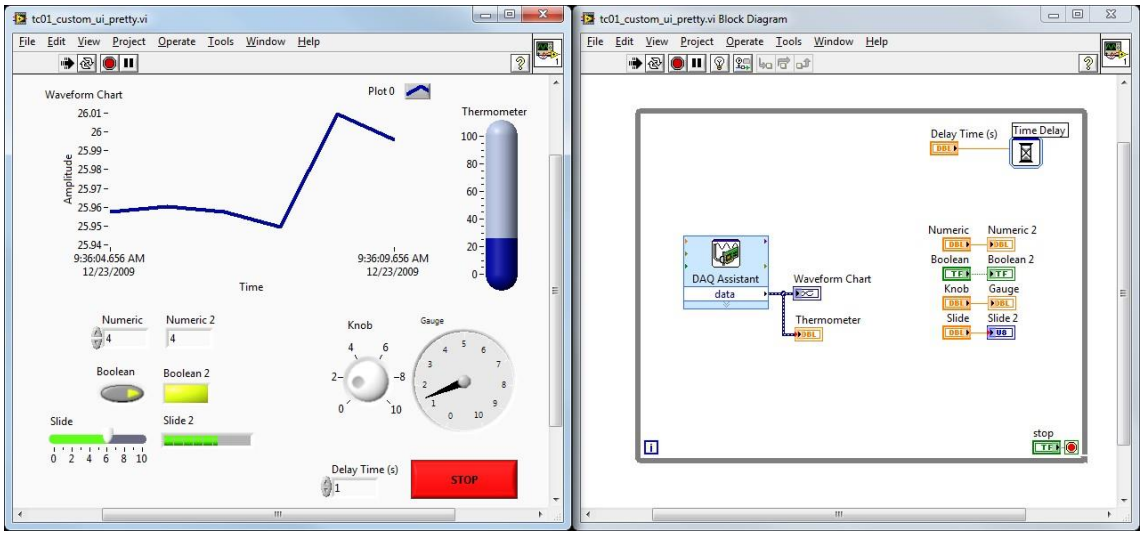


Imagen 4.2 Ejemplo de LabVIEW. [B39]

LabVIEW es un software con una gran cantidad de utilidades, pero en caso de no tener alguna en específico, es posible descargar de la red o de su página oficial las bibliotecas requeridas para un proyecto en específico.

National Instruments nos provee de diferentes versiones de LabVIEW:

- Base: Productividad al instante en desarrollo gráfico para DAQ y control de instrumentos.
- Completo: Un extenso juego de rutinas para matemáticas, procesamiento de señales y generación de señales.
- Profesional: La solución para ingenieros que necesitan desarrollo o validación de código.
- Paquetes de LabVIEW: LabVIEW Profesional más software adicional para industrias específicas.

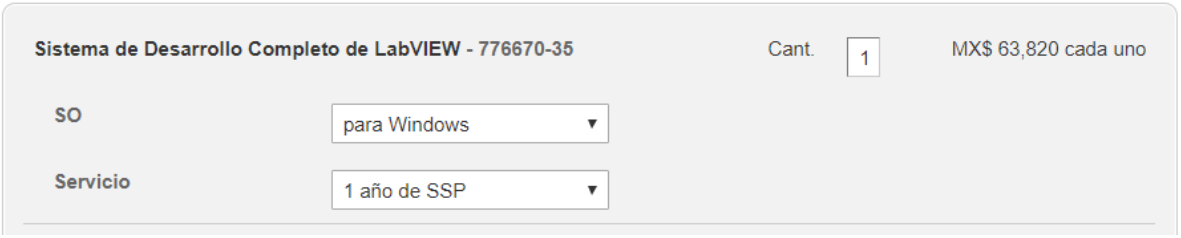
Donde la página oficial nos da la información detallada de cada una de sus versiones.

Sistema de Desarrollo Completo de LabVIEW:

- Integra hardware con una variedad de dispositivos hechos a medida y buses.
- Sintaxis de programación gráfica e intuitiva.
- Desarrollo de interfaces de usuario personalizadas para aplicaciones de ingeniería.
- Algoritmos integrados para procesamiento de señales, análisis, matemática y control PID.
- Un año de membresía de SSP para soporte técnico, formación y capacitación en línea y actualizaciones de software.

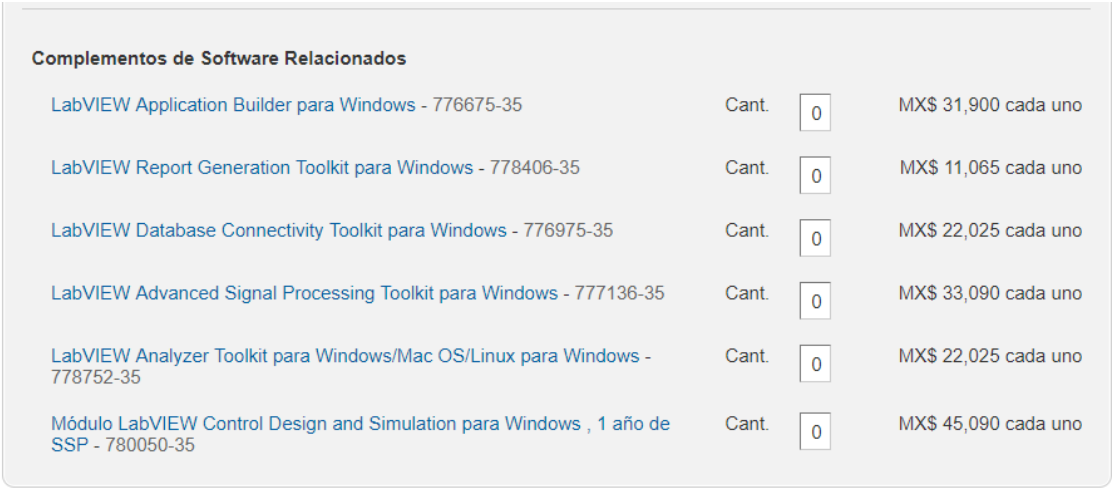
Nos provee de manuales, una versión de evaluación para los usuarios que buscan primero probar el producto antes de su respectiva compra, la lista de requerimientos y cierta ayuda para su correcta instalación.

La misma página oficial de LabVIEW nos muestra su respectiva tabla de costos:



[Consultado en Septiembre, 2017. Tienda oficial de National Instruments]

Y nos da la posibilidad de adquirir algunos complementos para su funcionamiento. Estos complementos no son obligatorios pero pueden ser de gran utilidad en algunos casos.

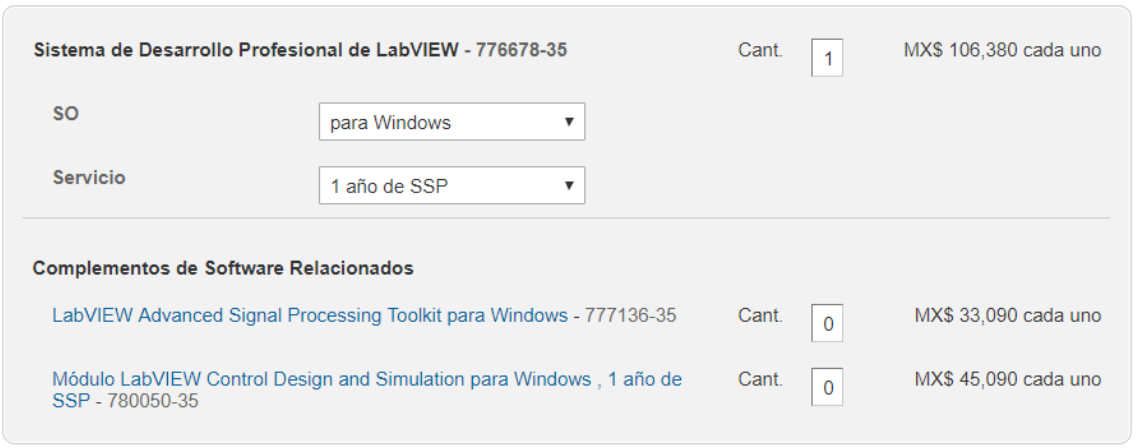


[Consultado en Septiembre, 2017. Tienda oficial de National Instruments]

Sistema de Desarrollo Profesional de LabVIEW:

- Software de diseño gráfico de sistemas completamente integrado.
- Soporte para una amplia gama de hardware de medida, E/S y buses.
- Interfaces de usuario personalizadas y guiadas por evento para medidas y control.
- Extensa funcionalidad para procesamiento de señales, análisis y matemáticas.
- Compilador avanzado para asegurar ejecución de alto rendimiento y optimización de código.
- Desarrollo profesional de software con revisión de la calidad del código, pruebas de unidad y creación de ejecutables.

De igual forma contamos con manuales de usuario, requerimientos del sistema y su respectiva tabla de costos y complementos.



[Consultado en Septiembre, 2017. Tienda oficial de National Instruments]

Sistema de Desarrollo Base de NI LabVIEW para Windows:

- Integra hardware desde una variedad de dispositivos e instrumentos de medida.
- Sintaxis de programación gráfica e intuitiva para simplificar la automatización.
- Desarrollo de interfaces de usuario personalizadas para aplicaciones de ingeniería.
- Un año de membresía de SSP para soporte técnico, formación y capacitación en línea y actualizaciones de software.

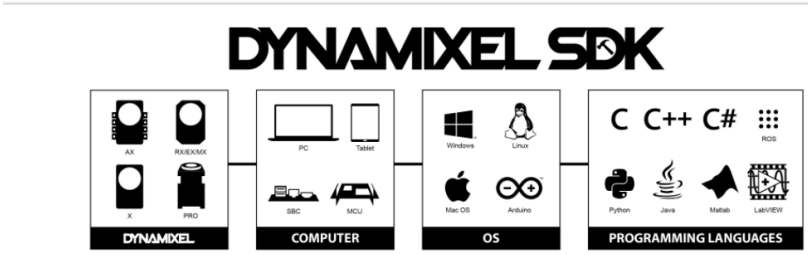
[Consultado en Septiembre, 2017. Tienda oficial de National Instruments]

Se debe tener en cuenta que el costo por versión es anual, así que si se requiere del software por más de un año se debe realizar la compra de la licencia cada año.

En cuanto a las bibliotecas para el control y monitoreo del brazo robótico, se emplea el "DYNAMIXEL SDK MAESTER" el cual está compuesto por una serie de programas que nos proveen de funciones para el control de los servomotores Dynamixel. Esta API* está diseñada para actuadores Dynamixel y plataformas basadas en Dynamixel. Provee de un manual para su correcto uso.

*: Interfaz de programación de aplicaciones (Application Programming Interface) se trata de un conjunto de funciones o métodos contenidos en una librería para ser utilizadas en otro sistema de software.

Esta API es completamente gratuita, se baja directamente de internet. Desde la página de Github podemos realizar la descarga, además nos provee de manuales y ayudas sobre la API.



1. Quick Overview

Introduction to the Dynamixel SDK and the Dynamixel SDK Wiki

2. Hardware Setup

How to set up the necessary hardware to use Dynamixel with the Dynamixel SDK

3. Software Preparation

How to set up the software environment to begin developing with the Dynamixel SDK

4. SDK Example

Descriptions of the provided SDK examples and instructions on how to execute these examples in different software environments

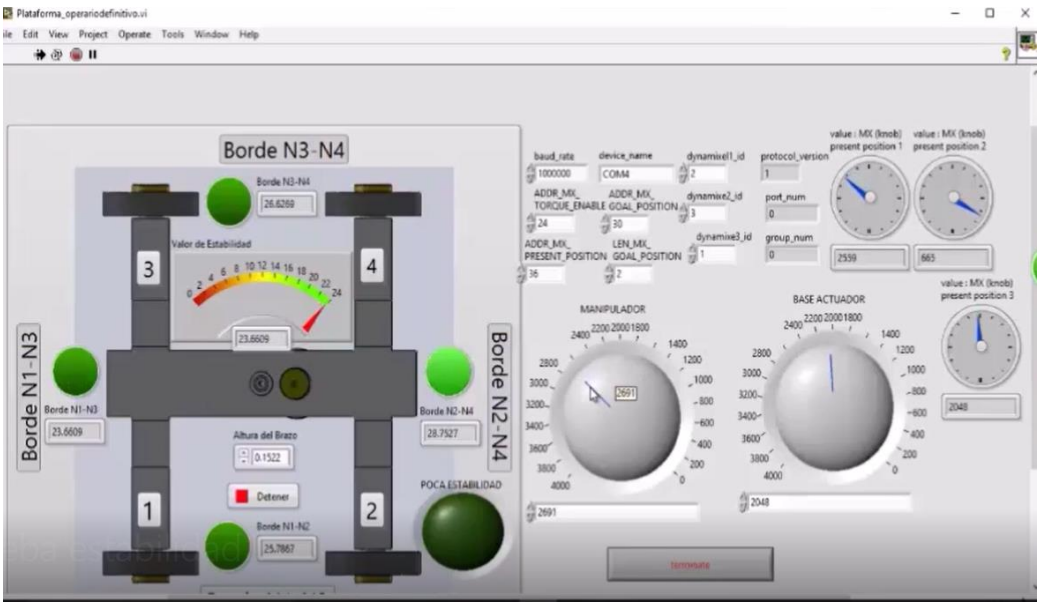
5. API Reference

Dynamixel SDK API Documentation

[A22]

A pesar que la biblioteca no requeriría un gasto, LabVIEW si lo necesita.

A continuación se muestra una captura de pantalla del diseño actual del sistema a replicar:



[A23]

Podemos visualizar tres interfaces, una de control, otra para el monitoreo y una parte gráfica en la cual observamos el comportamiento del brazo según su posición en una base especialmente diseñada con cuatro sensores, los sensores miden el peso inducido en ellos, mandan una señal a un módulo cuyo fin es procesar dicha señal para entregarla al programa y de esta forma el programa nos muestra en el apartado gráfico, la posición del brazo, el segmento de control únicamente nos permite darle el movimiento al brazo y por último la interfaz de monitoreo, este nos permite saber la posición de cada motor que conforma al brazo, el puerto COM* en el que se ubica, entre otras señales que el mismo motor manda.

Para el aparado de sensores se ocupan 4 galgas extensiométricas**, y el módulo NI-9237, el cual es fabricado por National Instruments, al igual que LabVIEW, una vez más encontramos la problemática del costo.

NI-9237

Módulo de Entrada de Puente/Tensión de la Serie C

Desde MX$ 27,010.00

VEA DETALLES DEL PRODUCTO

[Consultado en Septiembre, 2017. Tienda oficial de National Instruments]

El costo del producto es de $27,010.00 M/N, además de la inversión realizada por el LabVIEW se hizo la inversión para este módulo, el cual posee todo lo necesario para alimentar y medir hasta cuatro sensores, lo necesario para la base del brazo. Para resolver este problema, se empleó la ayuda de otro equipo, encargado de diseñar un sistema más económico y compatible con la solución planteada por la sección de software.

LabView siempre ha sido una herramienta eficaz pero altamente cara, agregando los complementos y herramientas a utilizar, los costos aumentan de sobremanera.

Elementos	LabVIEW Básico	LabVIEW Completo	LabVIEW Profesional
Licencia Vitalicia	$21,260	$63,820	$106,380
NI-9237	$27,010	$27,010	$27,010
Total a invertir	$48,270	$90,830	$133,390

Tabla 4.1 Costos y total.

La **Tabla 4.1** solo muestra el costo del software y el Módulo de Entrada de puente, pero el precio aumentara si se utilizan complementos para LabVIEW, además del tiempo que se tomara desarrollar el software. Para ello, presentamos una solución más factible haciendo uso de diversas tecnologías, como son, Raspberry Pi 3 y QT Creator los cuales se explicaran a detalle en el siguiente capítulo.

*: Puerto de comunicación serial, en este caso USB. Se asignan las siglas COM y un número. Esto es dado por el sistema operativo.

**: Sensores. Varían su resistencia respecto a la fuerza aplicada en ellas.

5) Propuesta de solución.

Como propuesta de solución se buscó un dispositivo con las interfaces necesarias para el control del brazo, desafortunadamente se llegó a la conclusión que lo más óptimo sería el empleo de una computadora, ya que con esta se puede hacer la interfaz correspondiente. Una laptop no podría ser, pues a pesar de contar con los requerimientos para el control, faltaría el apartado de monitoreo, por lo tanto se descartó esa idea. La solución fue la Raspberry Pi 3 el cual es considerado uno de los elementos dentro de los kits de electrónica, este dispositivo nos brinda la facilidad de una computadora pero en un espacio mucho más reducido, además de contar con interfaces de comunicación física como lo son los puertos seriales y el PINOUT* característico de estos dispositivos.

5.1 Elementos a usar en Hardware.

Los elementos principales en cuanto a hardware constan del dispositivo Raspberry Pi 3, servomotores Dynamixel (AX-18A, MX-64, MX-28), sistema de comunicación TTL Dynamixel (USB2DYnamixel) y su respectiva fuente de poder con su conexión. Del lado de sensores, se cuenta con 4 galgas.

Raspberry Pi 3:

Sus especificaciones son las siguientes:

- Procesador Quad Core 1.2GHz Broadcom BCM2837 64bit
- 1GB de memoria RAM
- BCM43438 wireless LAN y Bluetooth Low Energy (BLE) integrados en la tarjeta.
- Puerto de Ethernet
- 40-pins
- 4 puertos USB 2
- 4 Pole stereo output and composite video port
- Full size HDMI
- Puerto CSI para cámara
- DSI para coneccion de display tactil
- Puerto Micro SD
- Puerto Micro USB para alimentación

*: Interfaz física para conexión con elementos externos a la placa. Serie de pines integrados en la placa. Cada uno independiente de otro.

Imagen 5.1 Raspberri pi 3 Modelo B [A24]

Raspberry cuenta con su manual de usuario, haciendo más entendible el funcionamiento, además la página principal cuenta con tutoriales para la correcta instalación del sistema operativo.

De igual manera podemos encontrar en internet, cientos de tutoriales, guías, ejercicios, prácticas, y ejemplos de sistemas que se han desarrollado con ayuda de Raspberry Pi.

Robotis es la empresa encargada de Dynamixel, los cuales son pequeños actuadores específicamente creados para ser utilizados en el área de la robótica y la mecánica. En este caso y como ya se mencionó se hace uso de los modelos AX y MX, para ello Robotis nos proporciona los manuales de cada actuador en la red con un formato pdf, además de video tutoriales y asistencia telefónica.

A continuación se muestran las características de cada actuador involucrado en el brazo.

AX-18A:

- Peso: 54.5 g (AX-18F), 55.9 g (AX-18A)
- Dimensión: 32 mm * 50 mm * 40 mm
- Resolución: 0.29 °
- Relación de reducción de engranajes: 254: 1
- Par de Paro: 1.8N.m (a 12.0V, 2.2A)
- Velocidad sin carga: 97rpm (a 12V)
- Running Degree
  0 ° ~ 300 °
  Endless Turn
- Temperatura de funcionamiento: -5 ℃ ~ + 75 ℃
- Voltaje: 9 ~ 12V (voltaje recomendado 11.1V)
- Señal de comando: paquete digital

- Tipo de protocolo: comunicación serie asíncrona semidúplex (8 bits, 1 parada, sin paridad)
- Enlace (Físico): TTL Nivel Multi Drop (Conector tipo margarita)
- ID: 254 ID (0 ~ 253)
- Velocidad de comunicación: 7343bps ~ 1 Mbps
- Comentarios: posición, temperatura, carga, voltaje de entrada, etc.
- Material: Plástico de ingeniería

Imagen 5.2 Actuador/Servomotor Dynamixel AX-18A con complementos. [A25]

MX-64:

- MCU: ST CORTEX-M3 (STM32F103C8 @ 72MHZ, 32BIT)
- SENSOR DE POSICIÓN: codificador absoluto sin contacto (12BIT, 360 GRADOS)
  Maker: ams (www.ams.com), Part No: AS5045
- MOTOR: Maxon
- TASA DE BAUDIOS: 8000 bps ~ 4.5 Mbps
- ALGORITMO DE CONTROL: CONTROL PID
- Resolución: 0.088
- Running Degree
  ° ~ 360 °
  Endless Turn
- Peso: 135g
- Dimensión: 40.2 mm x 61.1 mm x 41 mm
- Relación de reducción de engranajes: 200: 1
- Par de par
  5.5N.m (a 11.1V, 3.9A),
  6.0N.m (a 12V, 4.1A)
  7.3N.m (a 14.8V, 5.2A)
- Sin velocidad de carga

58rpm (a 11.1V)
63rpm (a 12V)
78rpm (a 14.8V)

- Temperatura de funcionamiento: -5 ℃ ~ + 80 ℃
- Voltaje: 10 ~ 14.8V (voltaje recomendado 12V)
- Señal de comando: paquete digital
- Tipo de protocolo
  MX-64T / MX-64AT (comunicación serie asíncrona semidúplex (8 bits, 1 parada, sin paridad))
  MX-64R / MX-64AR (comunicación serie asíncrona RS485 (8 bits, 1 parada, sin paridad))
- Enlace (Físico)
  MX-64T / MX-64AT (TTL Level Multi Drop Bus)
  MX-64R / MX-64AR (RS485 Multi Drop Bus)
- ID: 254 ID (0 ~ 253)
- Comentarios: posición, temperatura, carga, voltaje de entrada, etc.
- Material: Full Metal Gear
  MX-64AR / MX-64AT (Cuerpo de Metal (Delantero), Cuerpo de Plástico de Ingeniería (Medio, Atrás))
  MX-64R / MX-64T (cuerpo de plástico de ingeniería (delantero, medio, trasero))
- Corriente de espera: 100 mA

Imagen 5.3 Actuador/Servomotor Dynamixel MX-64 con complementos. [A26]

MX-28:

- MCU: ST CORTEX-M3 (STM32F103C8 @ 72MHZ, 32BIT)
- SENSOR DE POSICION: codificador absoluto sin contacto (12BIT, 360 GRADOS)
  Maker: ams (www.ams.com), Part No: AS5045
- MOTOR: Maxon
- TASA DE BAUDIOS: 8000 bps ~ 4.5 Mbps
- ALGORITMO DE CONTROL: CONTROL PID
- Resolución: 0.088 °

- Running Degree
  - ° ~ 360 °
  - Endless Turn
- Peso: 77g
- Dimensión: 35.6 mm x 50.6 mm x 35.5 mm
- Relación de reducción de engranajes: 193: 1
- Par de par
  - 2.3N.m (a 11.1 V, 1.3 A),
  - 2.5N.m (a 12V, 1.4A)
  - 3.1N.m (a 14.8V, 1.7A)
- Sin velocidad de carga
  - 50rpm (a 11.1V)
  - 55rpm (a 12V)
  - 67rpm (a 14.8V)
- Temperatura de funcionamiento: -5 ℃ ~ + 80 ℃
- Voltaje: 10 ~ 14.8V (voltaje recomendado 12V)
- Señal de comando: paquete digital
- Tipo de protocolo
- MX-28T / MX-28AT (comunicación serie asíncrona semidúplex (8 bits, 1 parada, sin paridad))
- MX-28R / MX-28AR (comunicación serie asíncrona RS485 (8 bits, 1 parada, sin paridad))
- Enlace (Físico)
  - MX-28T / MX-28AT (TTL Level Multi Drop Bus)
  - MX-28R / MX-28AR (RS485 Multi Drop Bus)
- ID: 254 ID (0 ~ 253)
- Comentarios: posición, temperatura, carga, voltaje de entrada, etc.
- Material: Full Metal Gear
  - MX-28AR / MX-28AT (Cuerpo de metal (delantero), cuerpo de plástico de ingeniería (medio, trasero))
  - MX-28R / MX-28T (cuerpo de plástico de ingeniería (delantero, medio, trasero))
- Corriente de espera: 100 M.

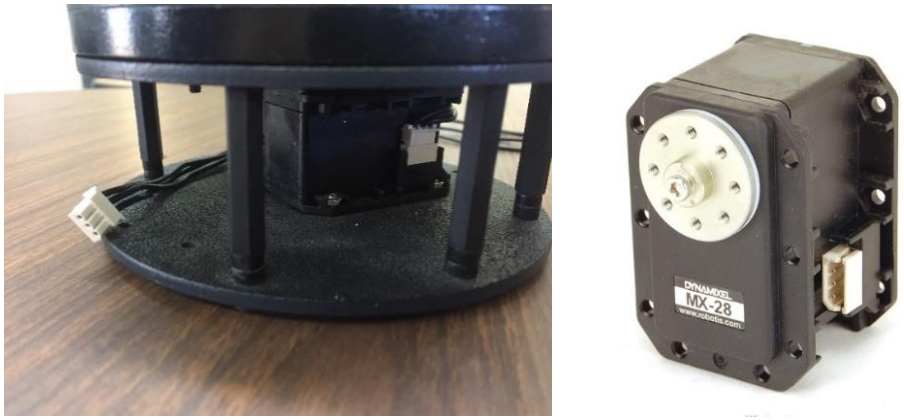


Imagen 5.4 Actuador/Servomotor Dynamixel MX-64 con complementos. [A27] [B40]

Cada actuador posee una memoria RAM y una memoria EEPROM y su respectiva tabla de control. A continuación se mostraran las tablas correspondientes a cada actuador, divididas entre los registros de la EEPROM y la RAM.

5.1.1 Tablas de registros internos de los actuadores Dynamixel.

AX-18A:

Area	Address (Hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
EEPROM	0 (0X00)	Model Number(L)	Lowest byte of model number	R	18(0X12)
	1 (0X01)	Model Number(H)	Highest byte of model number	R	0 (0X00)
	2 (0X02)	Version of Firmware	Information on the version of firmware	R	-
	3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
	4 (0X04)	Baud Rate	Baud Rate of Dynamixel	RW	1 (0X01)
	5 (0X05)	Return Delay Time	Return Delay Time	RW	250 (0XFA)
	6 (0X06)	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	Highest byte of clockwise Angle Limit	RW	0 (0X00)
	8 (0X08)	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit	RW	3 (0X03)
	11 (0X0B)	the Highest Limit Temperature	Internal Limit Temperature	RW	75 (0X46)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 (0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	140 (0X8C)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	215 (0XD7)
	15 (0X0F)	Max Torque(H)	Highest byte of Max. Torque	RW	3 (0X03)
	16 (0X10)	Status Return Level	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	LED for Alarm	RW	36(0x24)
	18 (0X12)	Alarm Shutdown	Shutdown for Alarm	RW	36(0x24)
	24 (0X18)	Torque Enable	Torque On/Off	RW	0 (0X00)
	25 (0X19)	LED	LED On/Off	RW	0 (0X00)

RAM	26 (0X1A)	CW Compliance Margin	CW Compliance margin	RW	1 (0X01)
	27 (0X1B)	CCW Compliance Margin	CCW Compliance margin	RW	1 (0X01)
	28 (0X1C)	CW Compliance Slope	CW Compliance slope	RW	32 (0X20)
	29 (0X1D)	CCW Compliance Slope	CCW Compliance slope	RW	32 (0X20)
	30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
	31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	-
	32 (0X20)	Moving Speed(L)	Lowest byte of Moving Speed (Moving Velocitiy)	RW	-
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed (Moving Velocitiy)	RW	-
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit (Goal Torque)	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit (Goal Torque)	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position (Present Velocity)	R	-
	37 (0X25)	Present Position(H)	Highest byte of Current Position (Present Velocity)	R	-
	38 (0X26)	Present Speed(L)	Lowest byte of Current Speed	R	-
	39 (0X27)	Present Speed(H)	Highest byte of Current Speed	R	-
	40 (0X28)	Present Load(L)	Lowest byte of Current Load	R	-
	41 (0X29)	Present Load(H)	Highest byte of Current Load	R	-
	42 (0X2A)	Present Voltage	Current Voltage	R	-
	43 (0X2B)	Present Temperature	Current Temperature	R	-
	44 (0X2C)	Registered	Means if Instruction is registered	R	0 (0X00)
	46 (0X2E)	Moving	Means if there is any movement	R	0 (0X00)
	47 (0X2F)	Lock	Locking EEPROM	RW	0 (0X00)
	48 (0X30)	Punch(L)	Lowest byte of Punch	RW	32 (0X20)
	49 (0X31)	Punch(H)	Highest byte of Punch	RW	0 (0X00)

[C3]

MX-64:

Area	Address (Hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
EEPROM	0 (0X00)	Model Number(L)	Lowest byte of model number	R	54 (0X36)
	1 (0X01)	Model Number(H)	Highest byte of model number	R	1 (0X01)
	2 (0X02)	Version of Firmware	Information on the version of firmware	R	-
	3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
	4 (0X04)	Baud Rate	Baud Rate of Dynamixel	RW	34 (0X22)
	5 (0X05)	Return Delay Time	Return Delay Time	RW	250 (0XFA)
	6 (0X06)	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	Highest byte of clockwise Angle Limit	RW	0 (0X00)
	8 (0X08)	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit	RW	15 (0X0F)
	11 (0X0B)	the Highest Limit Temperature	Internal Limit Temperature	RW	80 (0X50)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 (0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	160 (0XA0)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	255 (0XFF)
	15 (0X0F)	Max Torque(H)	Highest byte of Max. Torque	RW	3 (0X03)
	16 (0X10)	Status Return Level	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	LED for Alarm	RW	36 (0X24)
	18 (0X12)	Alarm Shutdown	Shutdown for Alarm	RW	36 (0X24)
	20 (0X14)	Multi Turn Offset(L)	multi-turn offset least significant byte (LSB)	RW	0 (0X00)
	21 (0X15)	Multi Turn Offset(H)	multi-turn offset most significant byte (MSB)	RW	0 (0X00)
	22 (0X16)	Resolution Divider	Resolution divider	RW	1 (0X01)
	24 (0X18)	Torque Enable	Torque On/Off	RW	0 (0X00)
	25 (0X19)	LED	LED On/Off	RW	0 (0X00)

RAM	26 (0X1A)	D Gain	Derivative Gain	RW	0 (0X00)
	27 (0X1B)	I Gain	Integral Gain	RW	0 (0X00)
	28 (0X1C)	P Gain	Proportional Gain	RW	32 (0X20)
	30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
	31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	-
	32 (0X20)	Moving Speed(L)	Lowest byte of Moving Speed (Moving Velocity)	RW	-
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed (Moving Velocity)	RW	-
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit (Goal Torque)	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit (Goal Torque)	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position (Present Velocity)	R	-
	37 (0X25)	Present Position(H)	Highest byte of Current Position (Present Velocity)	R	-
	38 (0X26)	Present Speed(L)	Lowest byte of Current Speed	R	-
	39 (0X27)	Present Speed(H)	Highest byte of Current Speed	R	-
	40 (0X28)	Present Load(L)	Lowest byte of Current Load	R	-
	41 (0X29)	Present Load(H)	Highest byte of Current Load	R	-
	42 (0X2A)	Present Voltage	Current Voltage	R	-
	43 (0X2B)	Present Temperature	Current Temperature	R	-
	44 (0X2C)	Registered	Means if Instruction is registered	R	0 (0X00)
	46 (0X2E)	Moving	Means if there is any movement	R	0 (0X00)
	47 (0X2F)	Lock	Locking EEPROM	RW	0 (0X00)
	48 (0X30)	Punch(L)	Lowest byte of Punch	RW	0 (0X00)
	49 (0X31)	Punch(H)	Highest byte of Punch	RW	0 (0X00)
	68 (0X44)	Current(L)	Lowest byte of Consuming Current	RW	0 (0X00)
	69 (0X45)	Current(H)	Highest byte of Consuming Current	RW	0 (0X00)
	70 (0X46)	Torque Control Mode Enable	Torque control mode on/off	RW	0 (0X00)

	71 (0X47)	Goal Torque(L)	Lowest byte of goal torque value	RW	0 (0X00)
	72 (0X48)	Goal Torque(H)	Highest byte of goal torque value	RW	0 (0X00)
	73 (0X49)	Goal Acceleration	Goal Acceleration	RW	0 (0X00)

[C4]

MX-28:

Area	Address (Hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
EEPROM	0 (0X00)	Model Number(L)	Lowest byte of model number	R	29 (0X1D)
	1 (0X01)	Model Number(H)	Highest byte of model number	R	0 (0X00)
	2 (0X02)	Version of Firmware	Information on the version of firmware	R	-
	3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
	4 (0X04)	Baud Rate	Baud Rate of Dynamixel	RW	34 (0X22)
	5 (0X05)	Return Delay Time	Return Delay Time	RW	250 (0XFA)
	6 (0X06)	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	Highest byte of clockwise Angle Limit	RW	0 (0X00)
	8 (0X08)	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit	RW	15 (0X0F)
	11 (0X0B)	the Highest Limit Temperature	Internal Limit Temperature	RW	80 (0X50)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 (0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	160 (0XA0)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	255 (0XFF)
	15 (0X0F)	Max Torque(H)	Highest byte of Max. Torque	RW	3 (0X03)
	16 (0X10)	Status Return Level	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	LED for Alarm	RW	36 (0X24)
	18 (0X12)	Alarm Shutdown	Shutdown for Alarm	RW	36 (0X24)

	20 (0X14)	Multi Turn Offset(L)	multi-turn offset least significant byte (LSB)	RW	0 (0X00)
	21 (0X15)	Multi Turn Offset(H)	multi-turn offset most significant byte (MSB)	RW	0 (0X00)
	22 (0X16)	Resolution Divider	Resolution divider	RW	1 (0X01)
R A M	24 (0X18)	Torque Enable	Torque On/Off	RW	0 (0X00)
	25 (0X19)	LED	LED On/Off	RW	0 (0X00)
	26 (0X1A)	D Gain	Derivative Gain	RW	0 (0X00)
	27 (0X1B)	I Gain	Integral Gain	RW	0 (0X00)
	28 (0X1C)	P Gain	Proportional Gain	RW	32 (0X20)
	30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
	31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	-
	32 (0X20)	Moving Speed(L)	Lowest byte of Moving Speed (Moving Velocity)	RW	-
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed (Moving Velocity)	RW	-
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit (Goal Torque)	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit (Goal Torque)	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position (Present Velocity)	R	-
	37 (0X25)	Present Position(H)	Highest byte of Current Position (Present Velocity)	R	-
	38 (0X26)	Present Speed(L)	Lowest byte of Current Speed	R	-
	39 (0X27)	Present Speed(H)	Highest byte of Current Speed	R	-
	40 (0X28)	Present Load(L)	Lowest byte of Current Load	R	-
	41 (0X29)	Present Load(H)	Highest byte of Current Load	R	-
	42 (0X2A)	Present Voltage	Current Voltage	R	-
	43 (0X2B)	Present Temperature	Current Temperature	R	-
	44 (0X2C)	Registered	Means if Instruction is registered	R	0 (0X00)
	46 (0X2E)	Moving	Means if there is any movement	R	0 (0X00)
	47 (0X2F)	Lock	Locking EEPROM	RW	0 (0X00)
	48 (0X30)	Punch(L)	Lowest byte of Punch	RW	0 (0X00)

	49 (0X31)	Punch(H)	Highest byte of Punch	RW	0 (0X00)
	73 (0X49)	Goal Acceleration	Goal Acceleration	RW	0 (0X00)

[C5]

Los registros internos (Address) son muy parecidos en los tres modelos, esto hace más fácil su manipulación a nivel de software, más adelante hablaremos sobre ello.

El controlador USB2Dynamixel es un dispositivo empleado para controlar los actuadores Dynamixel directamente desde una computadora, posee un puerto serial RS-232, un conector de 4 pins para comunicación RS-485, un conector de 3 pins para comunicación TTL y un switch para seleccionar la forma de comunicación a utilizar.

Imagen 5.5 Controlador USB2 para comunicación por puerto USB. [A28]

Dynamixel, también pone a nuestra disposición el manual en internet para el correcto uso y manejo de este dispositivo.

Para la fase de poder, se utiliza una fuente de alimentación muy parecida a un cargador de laptop, este posee las siguientes características:

- INPUT: 100-240v~50/60 Hz 1.6A
- OUTPUT: 12v CD 5A

Con su respectiva protección y adaptador para no dañar ningún dispositivo y tener buena comunicación con el USB2Dynamixel.

Para establecer la comunicación entre todos los actuadores, el controlador USB2Dynamixel y la fuente de poder, se hace uso de cabes especiales provistos por Dynamixel, son cables de 3 líneas (VCC, GND y DATA) específicamente diseñados para TTL.

Imagen 5.6 Cable de comunicación. [A29]

Imagen 5.7 Regulador en el paso de corriente entre el USB2 y los actuadores/servomotores. [A30]

Imagen 5.8 Alimentación (vista superior). [A31]

Imagen 5.9 Alimentación (vista inferior). [A32]

5.1.2 Sensores utilizados.

Los sensores que actualmente se están ocupando son galgas extensiométricas, las cuales ya fueron mencionadas en el capítulo anterior junto con el módulo que se encarga de administrar las señales, estas fueron colocadas en la base del brazo, en total son 4 sensores.
Las galgas extensiométricas miden la deformación sufrida en ellos, cuando se ejerce una presión o bien un peso, estos cambian su resistencia eléctrica y es aquí donde debido a la variación eléctrica, podemos calcular que existe presencia o ausencia de peso.

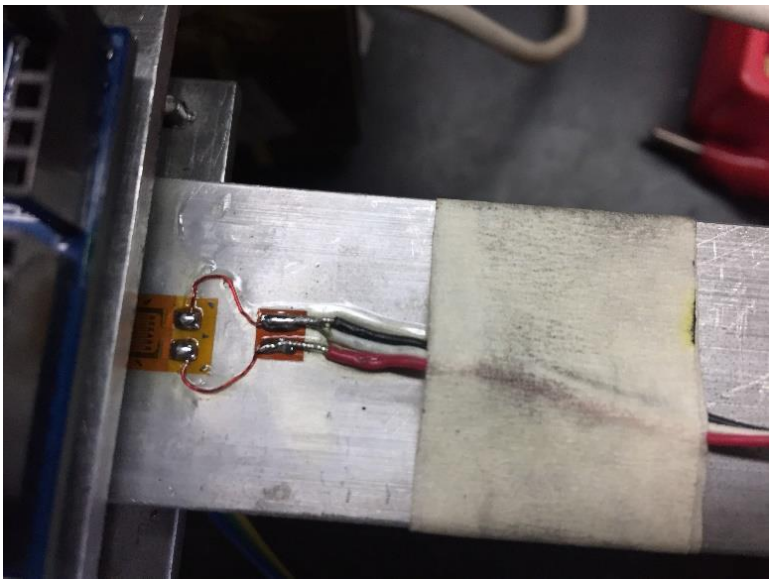
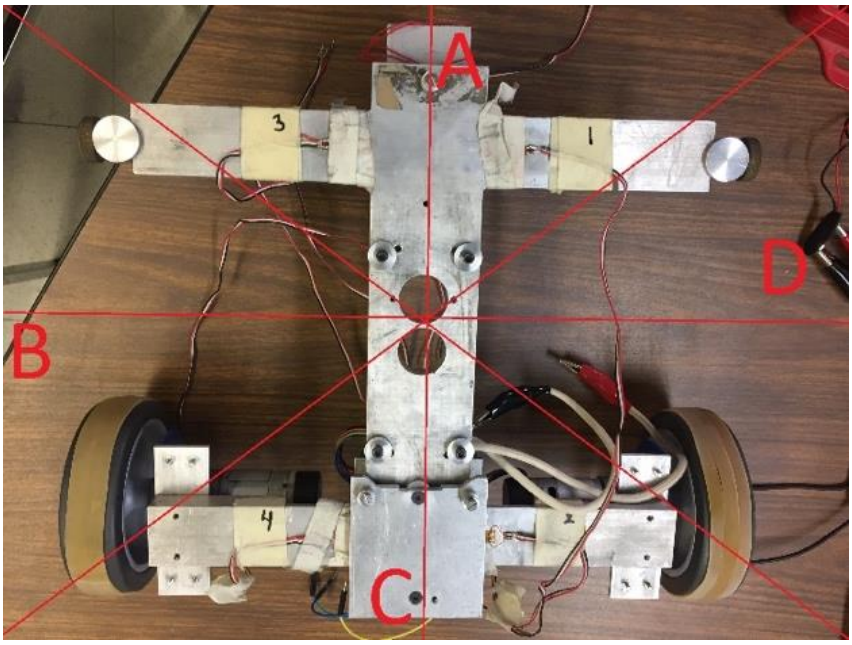


Imagen 5.10 Galga Extensiométrica en la base. [A33]

Explicado gráficamente, en la siguiente imagen podemos ver la base utilizada, señalando el lugar donde están posicionados los sensores.

Dividiremos la base en 8 partes.



[A34]

Si el brazo es colocado apuntando hacia la sección A los sensores 1 y 3 marcaran un ligero cambio de presión, a lo cual el programa reaccionara indicando gráficamente en la interfaz la variación de ambos sensores. Claro para pasar la señal generada por los sensores al programa, se debe ocupar una interfaz intermedia, la cual es desarrollada por otro equipo especializado en el apartado de electrónica.

En cambio, si el brazo se encuentra señalando el punto B, los sensores 3 y 4 mostraran el cambio efectuado en su interior y lo podremos visualizar en la interfaz del programa.

Así sucesivamente, en cualquier punto donde se ejerza una presión, se debe tener el monitoreo de los sensores y así el programa sabrá cómo actuar, si permite al usuario continuar con la manipulación del brazo o si bloquea la comunicación para evitar una volcadura y regresa el brazo a una posición estándar.

5.1.3 Ventajas en el uso de Raspberry.

Raspberry además de ser económico, nos provee de varias interfaces para comunicación con las cuales podemos efectuar el control y monitoreo del brazo. Lo fundamental del proyecto es el control para un "libre" movimiento (Considerando que el movimiento del brazo es limitado por la mecánica aplicada en él) y el monitoreo para saber dónde se encuentra posicionado el brazo con respecto a su base. Gracias a la arquitectura interna de la Raspberry (ARM) podemos cargar cualquier sistema operativo compatible con esa arquitectura, como lo es RASPBIAN o inclusive una versión especial de Windows 10.

Existen dispositivos como Arduino, el cual es una placa programable, de fácil uso y de software libre. Un elemento bastante poderoso dentro de los kits de electrónica, pero no resulto factible en este proyecto, ya que nos veíamos limitados por su cantidad de puertos físicos para conectar los sensores, además nos obligaba a hacer uso de una computadora normal, pues Arduino se encuentra más orientado al hardware que al software, seria básicamente como usar una PIC, pues podemos tener el monitoreo de los sensores pero para el control de los actuadores necesitamos de la interfaz de usuario, cosa que en Arduino no es factible.

Raspberry posee todas las interfaces de comunicación necesarias, además del PINOUT posee puertos como el de Bluetooth y la antena de WIFI interna, es sumamente portable, se alimenta con 5 volts, tiene 4 puertos USB 2.0 más que suficiente para los periféricos necesarios y el controlador de los actuadores, jack de 3.5 para entrada de audífonos o bocinas, salida HDMI para conectar con cualquier pantalla, bus para conexión con pantalla táctil y cámara. Simplemente es una computadora en miniatura.

5.2 Elementos a usar en Software.

Entrando al apartado del software utilizado, se hace uso principalmente del sistema operativo RASPBIAN JESSIE, el cual está basado en el sistema operativo DEBIAN LINUX. RASBIAN JESSIE es completamente gratuito y la descarga se realiza en la página principal de Raspberry, tenemos para seleccionar dos versiones. La versión RASPBIAN STRETCH LITE y la versión RASPBIAN STRETCH WITH DESKTOP, la utilizada en el diseño de este sistema es RASPBIAN STRETCH WITH DESKTOP ya que nos provee desde el primer arranque de la interfaz gráfica para facilitar la relación usuario-máquina.

Rasbian nos otorga un tutorial completo de como instalar cualquiera de las dos versiones de RASBIAN JESSIE que se desee.

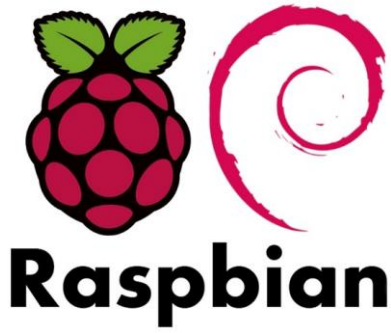


Imagen 5.12 Logo Raspbain. [B41]

Otro software a utilizar es QT CREATOR, se trata de un IDE (Integrated Development Environment) de desarrollo para aplicaciones en C++ y QML, forma parte de QT PROJECT. QT CREATOR fue elegido debido a su simple e intuitiva interfaz, además provee de autocompletado que nos ahorra tiempo de escritura y una gran facilidad al momento de crear las interfaces gráficas, ya que cuenta con la modalidad toma y arrastra, para ir armando de esta forma aquello que será mostrado al usuario. De igual forma, QT CREATOR posee sus manuales de instalación, guías y ejemplos. Además de su facilidad de uso existen múltiples bibliotecas que podemos ocupar para un mejor desarrollo de programas y generar códigos más "limpios".QT5 y QT CREATOR son completamente gratuitos y se encuentran disponibles para sistemas Linux, Windows y macOS.

Imagen 5.13 Logo Qt Creator. [B42]

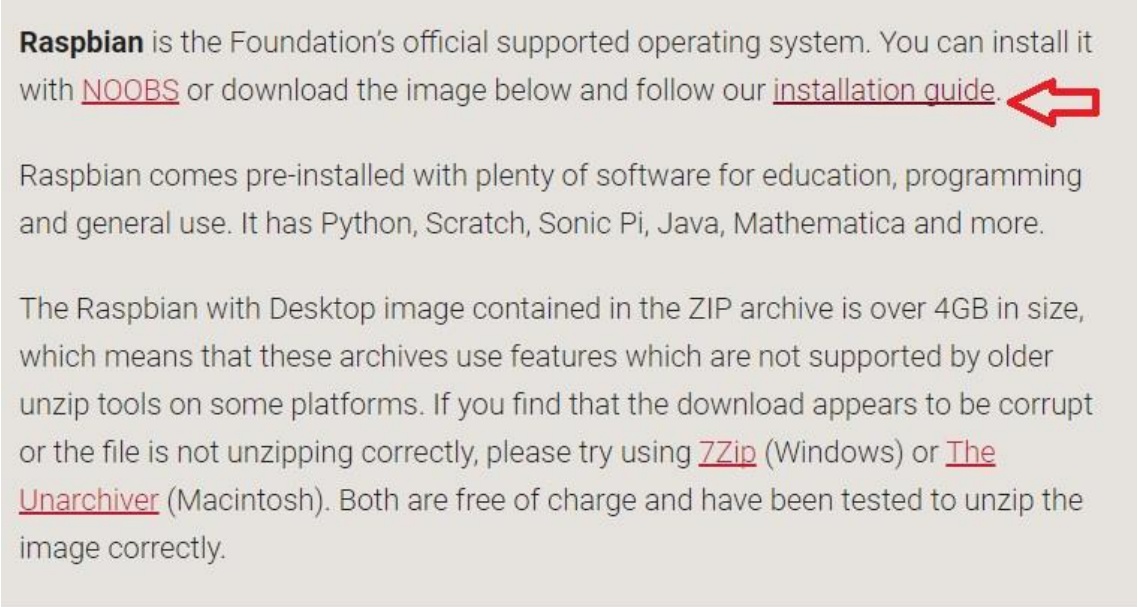
Para la manipulación del brazo por medio de la programación, se hizo uso de Dynamixel SDK, parecido al utilizado en LabVIEW pero esta vez orientado a C++. Dynamixel SDK nos provee de las bibliotecas adecuadas para el control de los actuadores, estas bibliotecas nos facilitan muchos aspectos de la programación, desde la comunicación con los actuadores hasta él envió de paquetes para que estos comiencen su funcionamiento. Encontrar este conjunto de herramientas fue complicado y se carece de información, muchos ejercicios con lo que llegue a encontrarme hacían uso de Arduino, pero en Raspberry casi no hay información. Una vez encontrado Dynamixel SDK se facilitó todo el proceso, pues igualmente cuenta con su guía de instalación y algunos ejemplos para su correcto uso.

Imagen 5.14 Dynamixel SDK. [B43]

A continuación se muestran los pasos a seguir según la guía de instalación para el grabado del sistema operativo, los pasos serán realizados desde una máquina con sistema operativo WINDOWS 10 de 64bits y la instalación de QT5, QT CREATOR y Dynamixel SDK se realizaran directamente en la Raspberry.

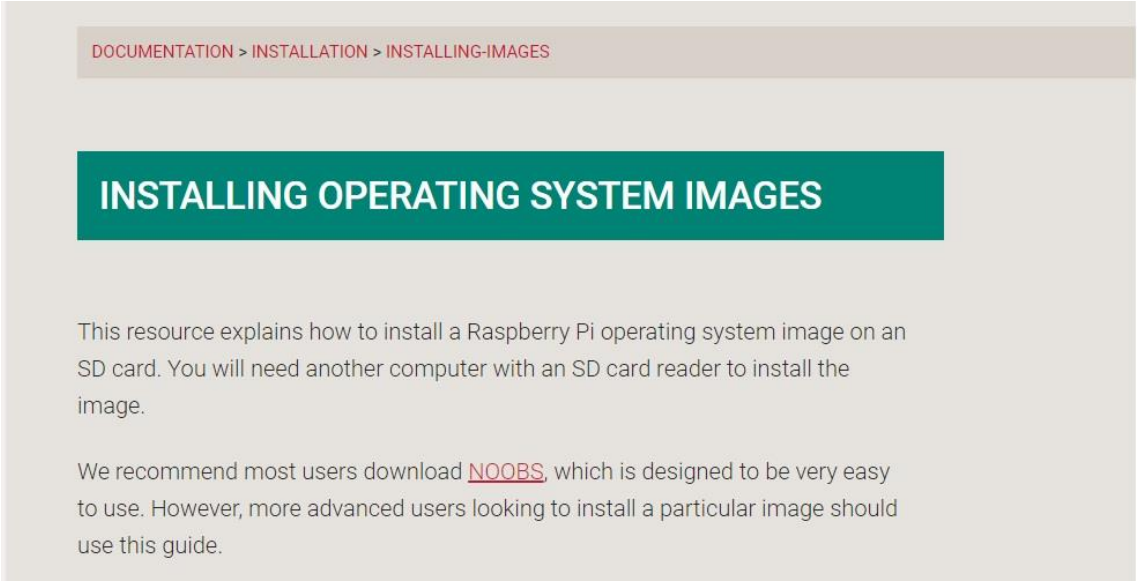
5.2.1 Instalación de los componentes a usar.

La página a la cual debemos acceder es https://www.raspberrypi.org/downloads/raspbian/ (Página consultada en Octure del 2017).
Ahí veremos el siguiente apartado:



[Captura. Consultado en Octubre del 2017]

Damos clic donde dice “installation guide”, y veremos la siguiente página.
https://www.raspberrypi.org/documentation/installation/installing-images/README.md
Ahí podemos visualizar de primera instancia la siguiente información.



[Captura. Consultado en Octubre del 2017]

Explorando un poco más en esa misma página, podemos encontrar lo siguiente:

[Captura. Consultado en Octubre del 2017]

La flecha azul señala un link donde podremos realizar la descarga del RASPBIAN que deseemos. Las flechas roja, morado y gris señalan el software que podemos ocupar para la descompresión del RASPBIAN descargado, en mi caso lo ideal sería usar 7-Zip, pero si se tiene WINRAR (como yo), puede realizarse la descompresión del SO con ayuda de ese programa.
En mi caso el archivo descargado se ve de la siguiente forma:

[A35]

Y una vez descomprimido tenemos un archivo como el siguiente:

[A36]

Esta es el archivo que se grabara en la memoria SD para así dar arranque desde las Raspberry. A continuación se indica la forma de grabar este archivo en la memoria SD.

Writing an image to the SD card

You will need to use an image writing tool to install the image you have downloaded on your SD card.

Etcher is a graphical SD card writing tool that works on Mac OS, Linux and Windows, and is the easiest option for most users. Etcher also supports writing images directly from the zip file, without any unzipping required. To write your image with Etcher:

- Download Etcher and install it.
- Connect an SD card reader with the SD card inside.
- Open Etcher and select from your hard drive the Raspberry Pi `.img` or `.zip` file you wish to write to the SD card.
- Select the SD card you wish to write your image to.
- Review your selections and click 'Flash!' to begin writing data to the SD card.

For more advanced control of this process, see our system-specific guides:

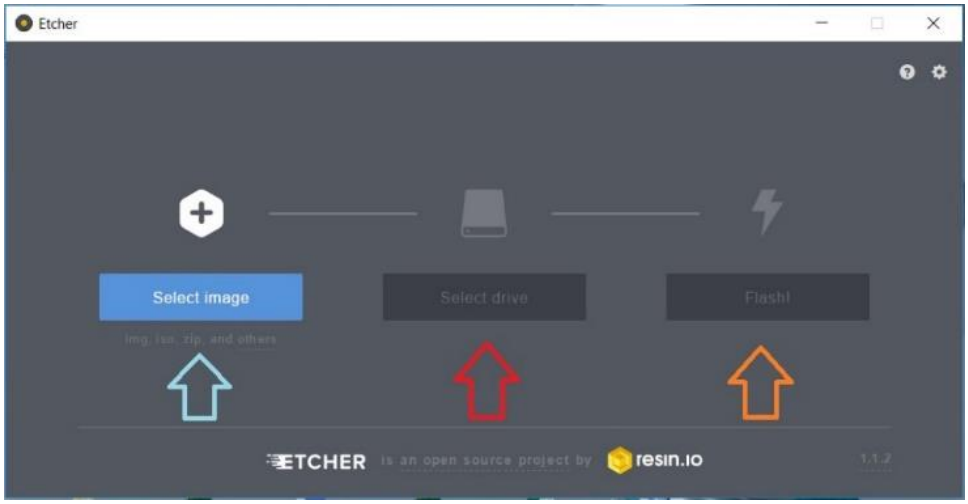
- Linux
- Mac OS
- Windows

[Captura. Consultado en Octubre del 2017]

Como se muestra en esta imagen, es necesario descargar un programa llamado Etcher* para grabar el SO** descargado en la Micro memoria SD. La flecha roja nos indica la liga de descarga del programa.
Una vez descargado e instalado el programa se tendrá algo como esto:

[A37]

Continuando con los pasos de la guía de instalación solo hace falta grabar el SO en el la Micro memoria SD para poder comenzar a usar la Raspberry.



[A38]

En la flecha azul, colocaremos (haciendo clic y seleccionando) el archivo ya sea en ZIP o en IMG del Sistema Operativo RASPBIAN JESSIE (en nuestro caso), preferentemente se hace la selección del archivo IMG.

La flecha roja indica en que dispositivo realizaremos el grabado del SO.

Y por último la flecha naranja muestra la opción de “Flash!” lo cual hace referencia a grabar inmediatamente, una vez seleccionados los otros dos campos, podremos hacer uso de este botón para comenzar el grabado.

Esto es únicamente para grabar el SO en la Micro memoria SD. Procederemos a ver como luce una vez iniciado dentro de la Raspberry.

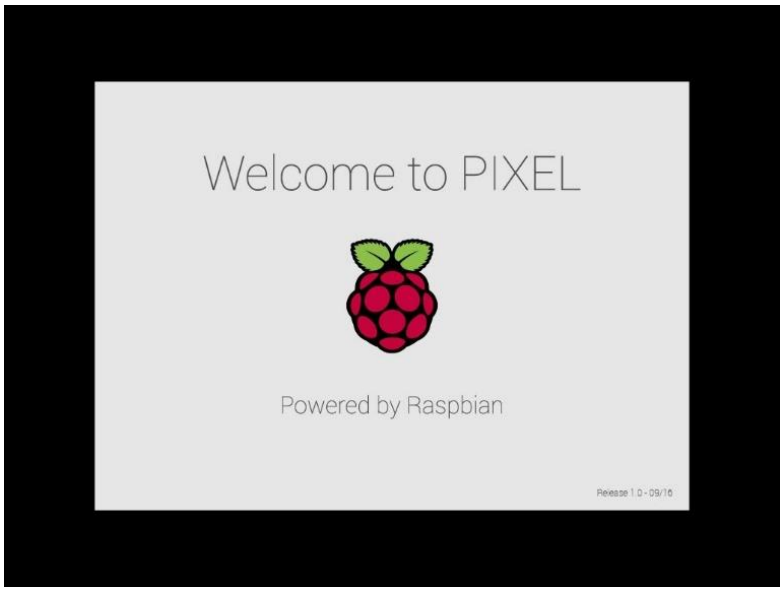
Teniendo ya iniciada la Raspberry, comenzaremos con la instalación de QT5 y QT CREATOR.

Para la instalación de QT5 y QT CREATOR, primero y antes que nada debemos hacer la conexión de los periféricos principales a la Raspberry y colocar la SD en el puerto adecuado.

*: Programa especializado para montar imágenes especiales en tarjetas SD o memorias FLASH USB

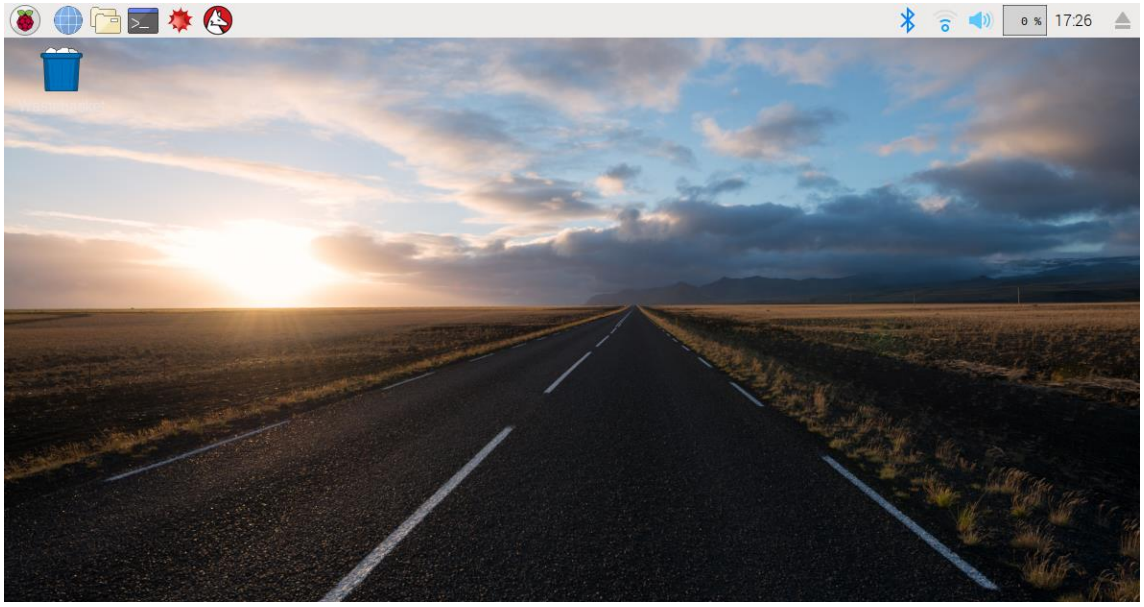
**: Sistema Operativo

Una vez conectado el teclado, mouse, HDMI y teniendo colocada la tarjeta SD en su slot correspondiente procedemos a darle poder con el cargador.
El arranque del sistema operativo se vera de la siguiente forma:



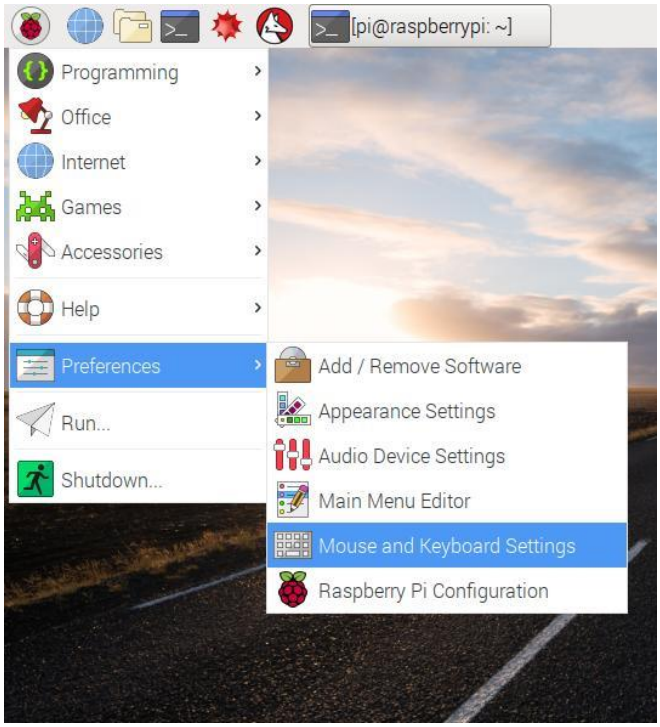
[A39]

Una vez iniciado podremos ver el escritorio, como en cualquier otro sistema operativo:

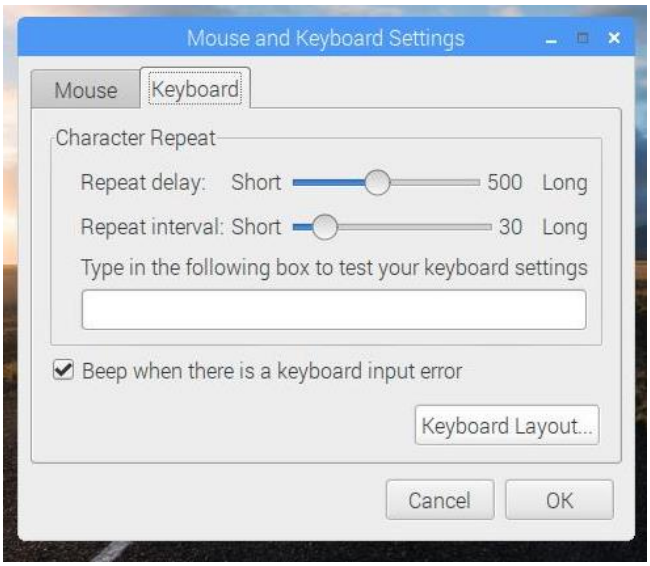


[A40]

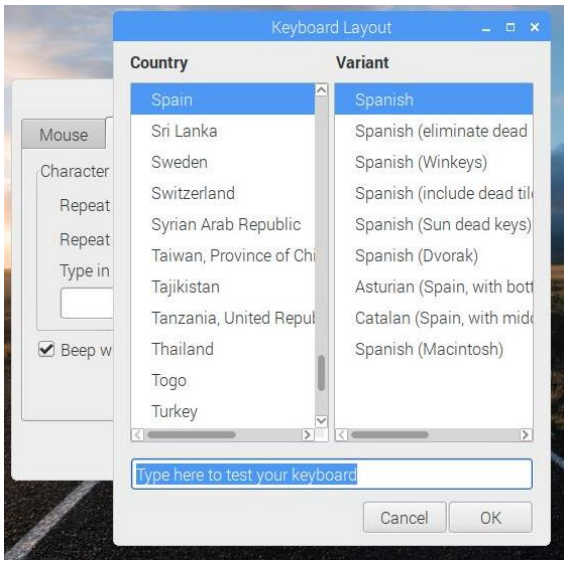
Comenzando a interactuar con este nuevo sistema operativo, realizaremos el cambio de teclado y si se quiere de idioma. Localizamos en la “barra de tareas” el logo de Raspberry, daremos clic y seleccionaremos la opción “Preferences” seguido de “Mouse and Keyboard Settings”. Ya seleccionado procedemos a dirigirnos a la pestaña de “Keyboard”, hacer clic en el botón de “Keyboard Layout…”, buscamos Spain y Spanish y terminamos haciendo clic en “OK”, esto es únicamente para el cambio de teclado a español. El cambio de idioma requeriría de un reinicio para efectuar los cambios y eso no se hará en este tutorial ya que no es fundamental.


[pi@raspberrypi: ~]
Programming
Office
Internet
Games
Accessories
Help
Preferences
Run...
Shutdown...
Add / Remove Software
Appearance Settings
Audio Device Settings
Main Menu Editor
Mouse and Keyboard Settings
Raspberry Pi Configuration

[A41]


Mouse and Keyboard Settings
Mouse
Keyboard
Character Repeat
Repeat delay: Short 500 Long
Repeat interval: Short 30 Long
Type in the following box to test your keyboard settings
Beep when there is a keyboard input error
Keyboard Layout...
Cancel
OK

[A42]



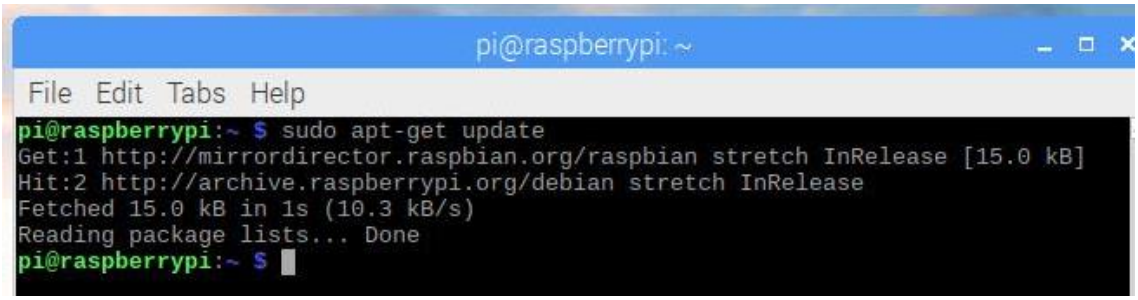
[A43]

Habiendo hecho los cambios en el teclado, procederemos a abrir una terminal, en el logo de Raspberry seguido de “Accessories” y “Terminal”, o bien con las teclas CTRL+ALT+T directamente (en este punto se debe haber configurado el Internet por WIFI o cable de red).

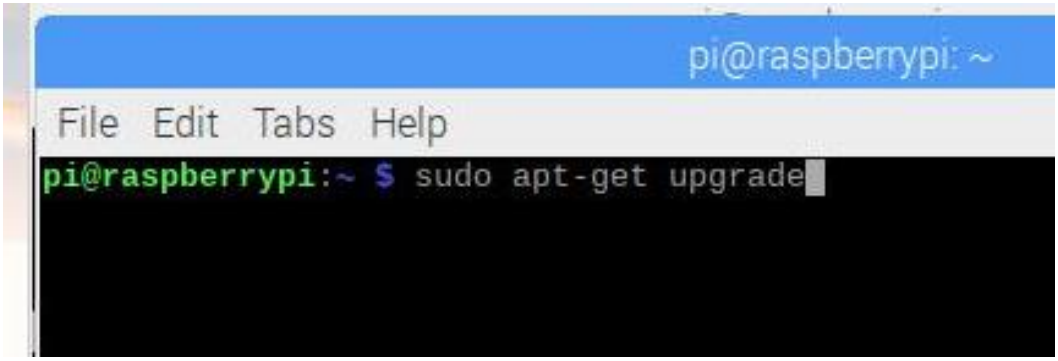
Lo que comúnmente se hace en todo sistema Linux, es buscar actualizaciones y poner el sistema al 100 para eso usaremos los siguientes comandos:

$ sudo apt-get update

$ sudo apt-get upgrade



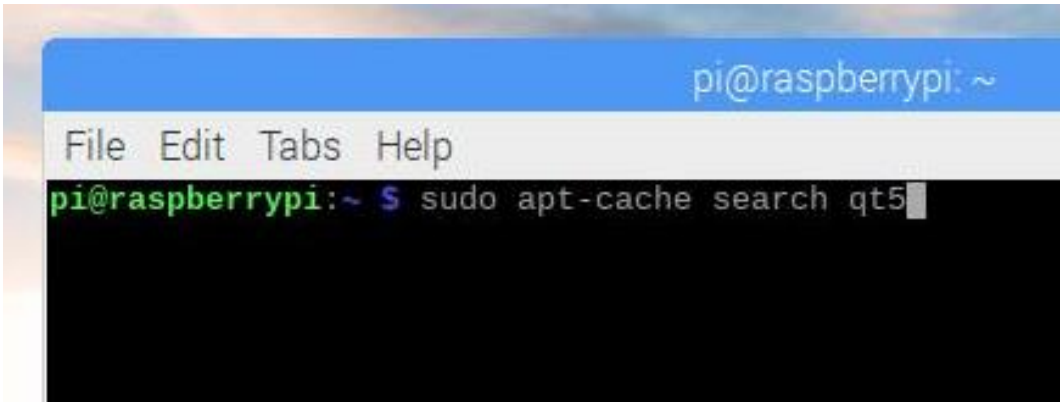
[A44]



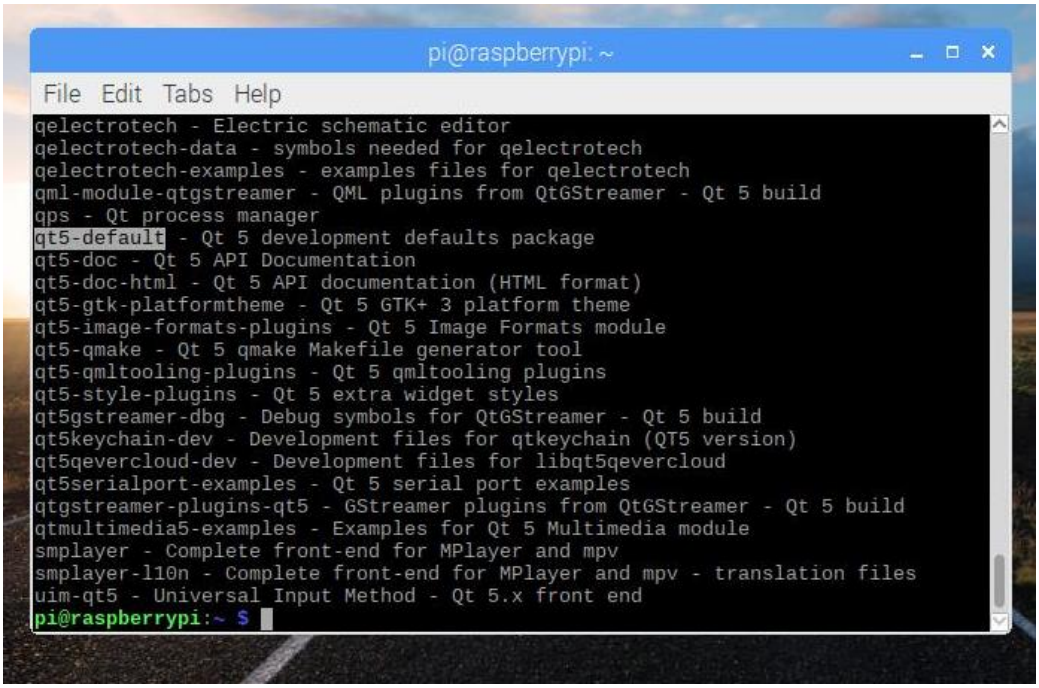
[A45]

Al no tener demasiadas cosas almacenadas y ser el primer arranque, los comandos los ejecutara sin problemas, en caso de existir algún detalle hacer las recomendaciones que se nos muestran. Continuando con la instalación de QT CREATOR, buscaremos dentro de APT si tenemos en lista a QT5 (el cual es requerido para el funcionamiento de QT CREATOR) con el siguiente comando:

$apt-cache search qt5



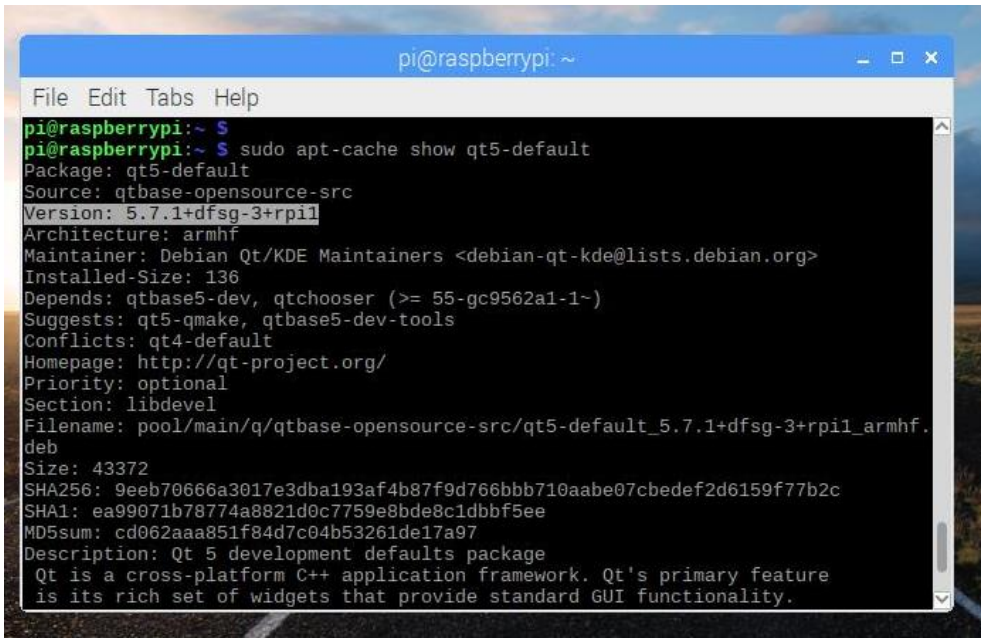
[A46]



[A47]

Verificaremos la versión del paquete qt5-default haciendo uso del siguiente comando:

$apt-cache show qt5-dedault

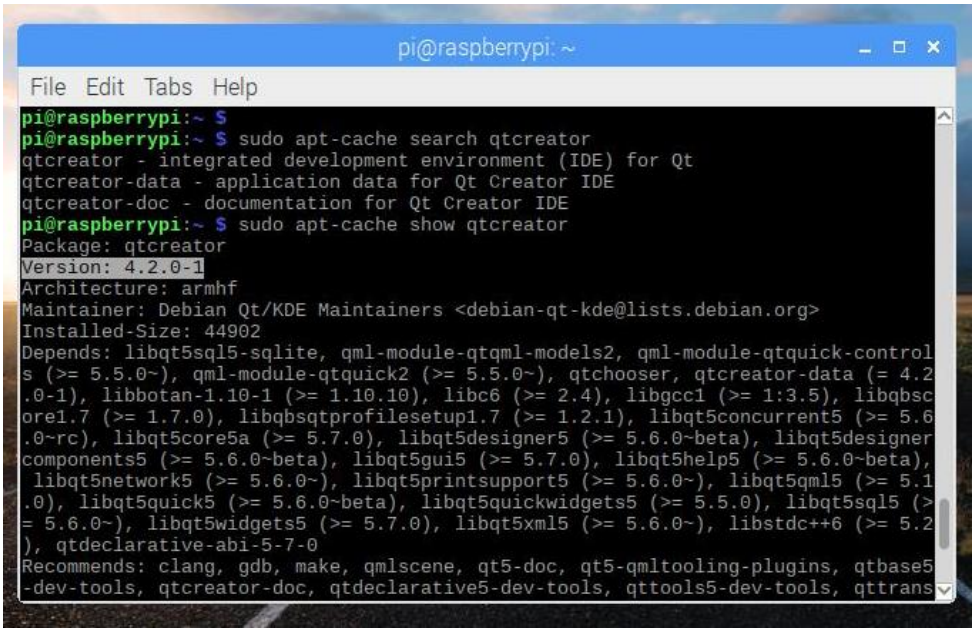


[A48]

Aplicaremos los dos comandos pasador pero esta vez directamente con QT CREATOR.

$apt-cache search qtcreator

$apt-cache show qtcreator

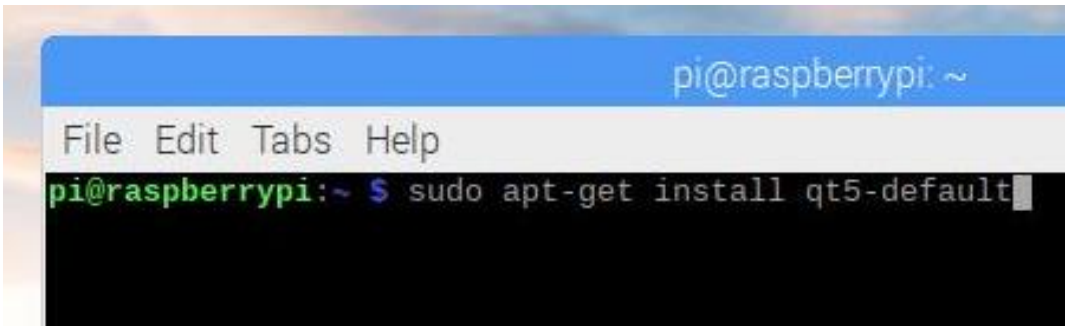


[A49]

Una vez teniendo la información de las versiones y sabiendo que podemos acceder a ellos desde APT procederemos a descargar e instalar QT5 y QT CREATOR.

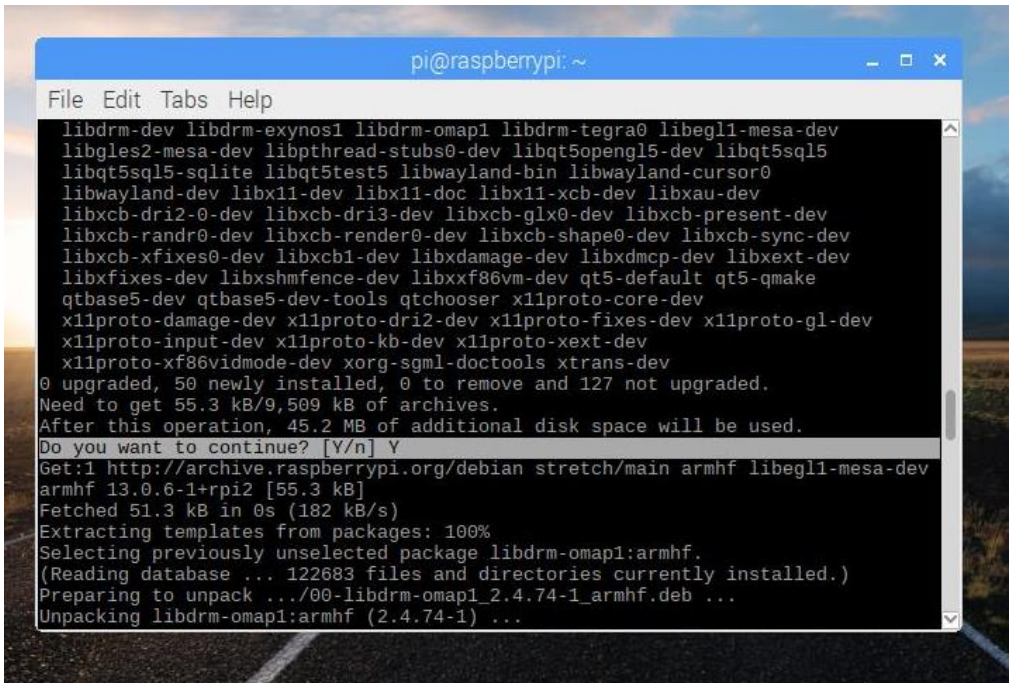
Para QT5 usaremos el comando:

$sudo apt-get install qt5-default



[A50]

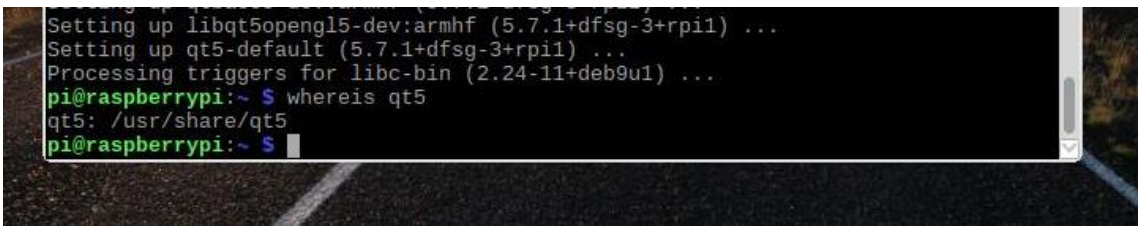
En el momento que pide confirmación tecleamos “Y” seguido de “ENTER” y esperamos a que termine la instalación. En caso de presentar algún problema hacer caso a las indicaciones que nos arroje, de no encontrar solución buscar en la guía de instalación de qt5.



[A51]

Ya terminada la instalación, personalmente recomiendo verificar que el archivo descargado exista dentro de la Raspberry, con el siguiente comando:

$whereis qt5

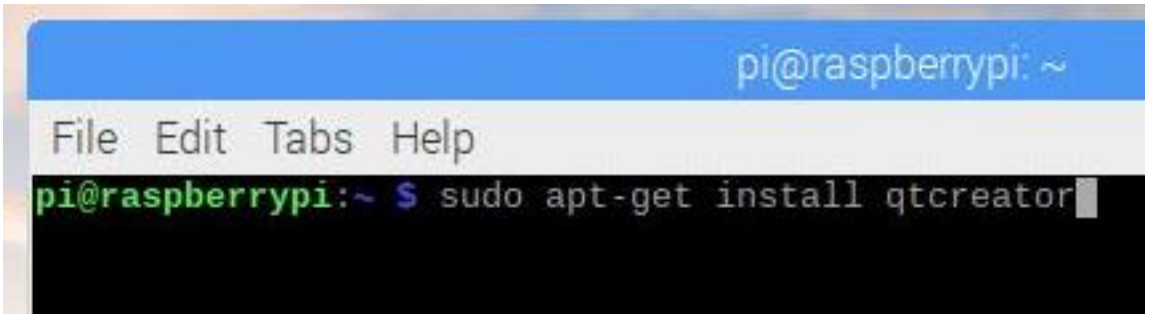


[A52]

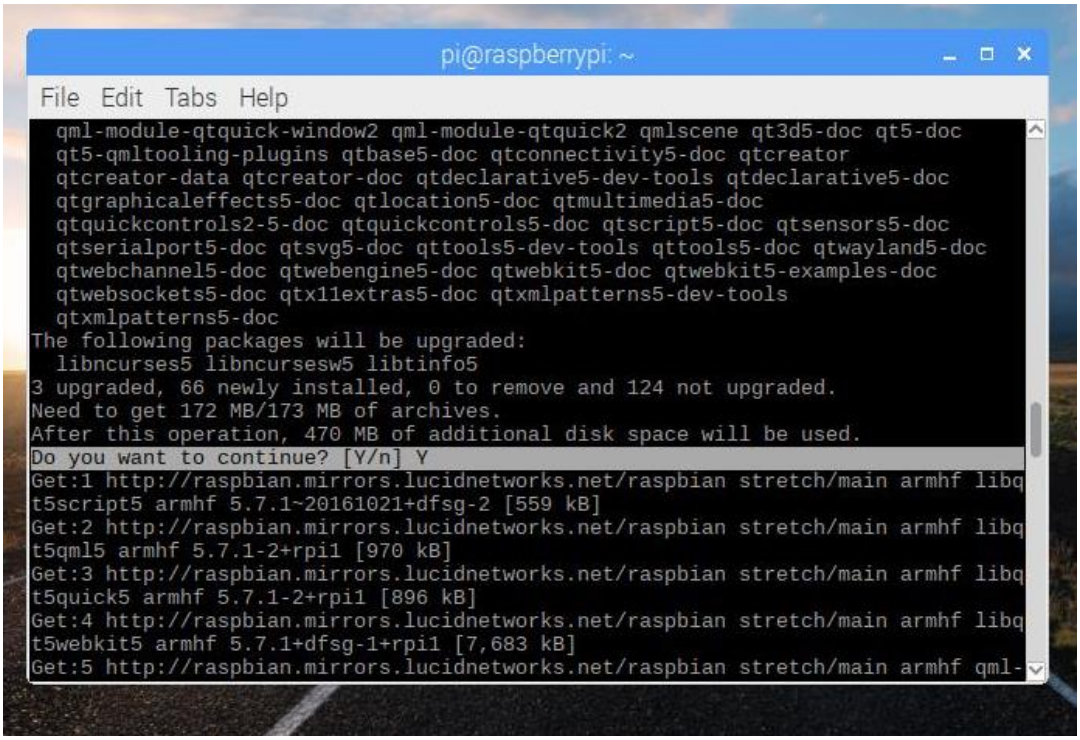
Volveremos a correr los mismos comandos pero esta vez con qtcreator:

$sudo apt-get install qtcreator

$whereis qtcreator



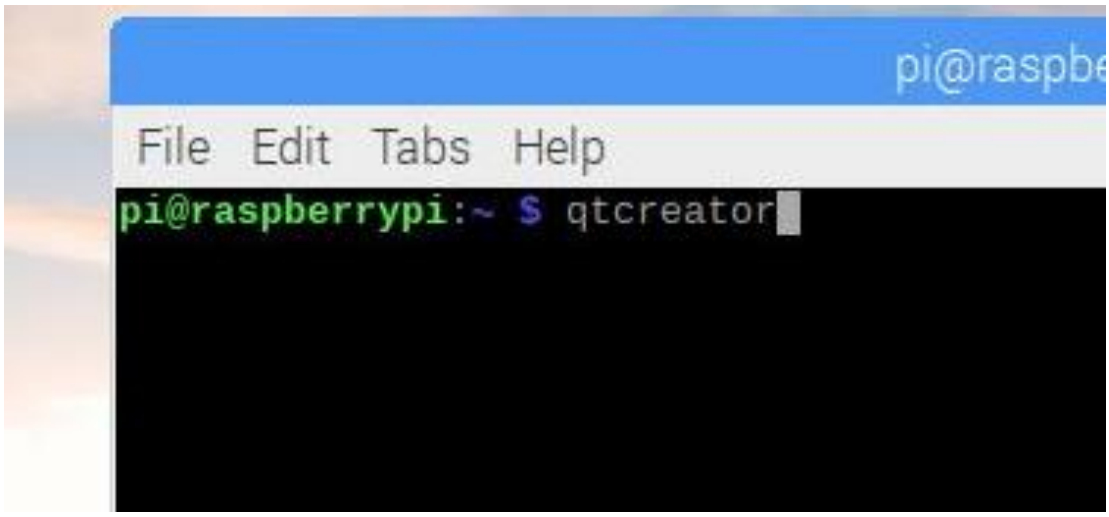
[A53]



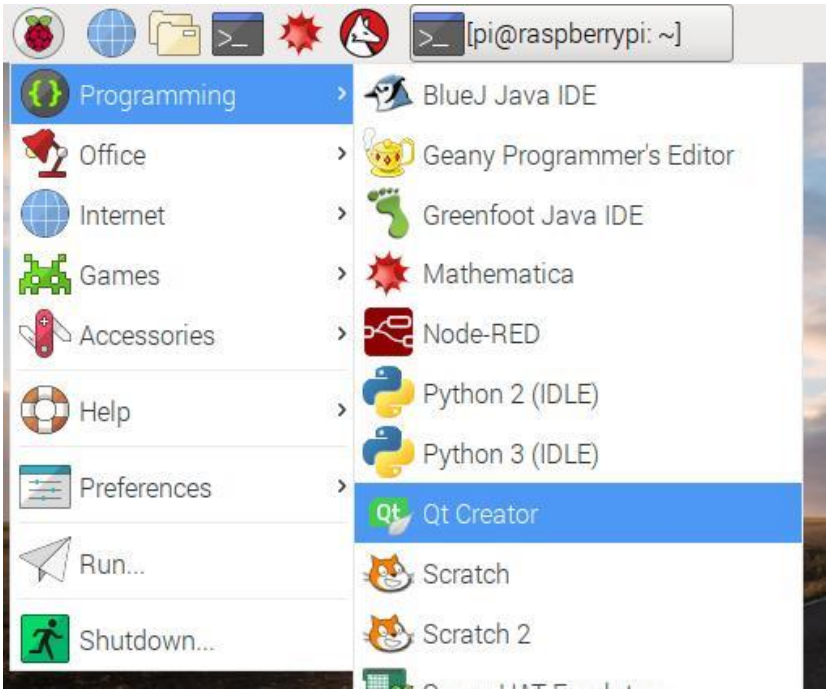
[A54]

[A55]

Para ejecutar QT CREATOR, solo basta con escribir qtcreator en la consola o buscarlo directamente en el inicio de Raspberry.



[A56]



[A57]

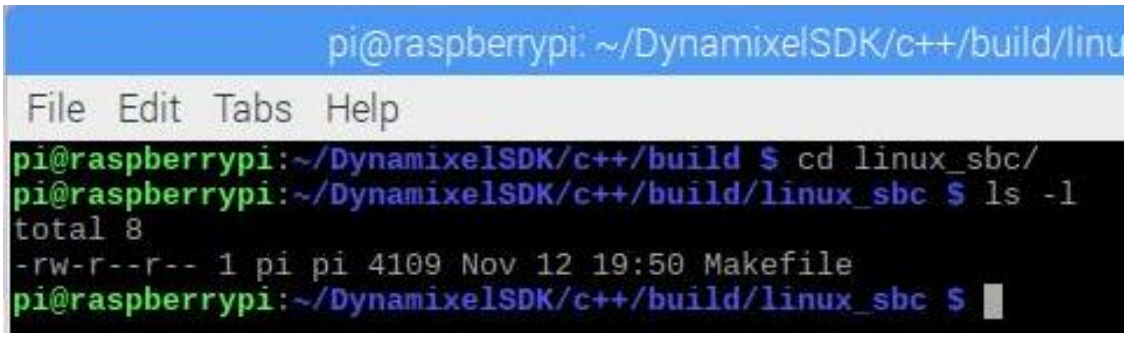
En este proyecto se hace uso de las bibliotecas Dynamixel SDK 3.4.2 y la instalación para que funcionen con QT CREATOR es como se muestra a continuación.

El tutorial a realizar es oficial de Dynamixel, comenzaremos clonando la carpeta contenedora de los archivos necesarios con el siguiente comando desde la terminal:

$git clone https://github.com/ROBOTIS-GIT/DynamixelSDK.git

Ahora vamos a compilar la librería de C++, pues es el lenguaje que maneja QT CREATOR. Esto lo haremos accediendo a la carpeta /DynamixelSDK/c++/build y entrando a la carpeta de “linux sbc”

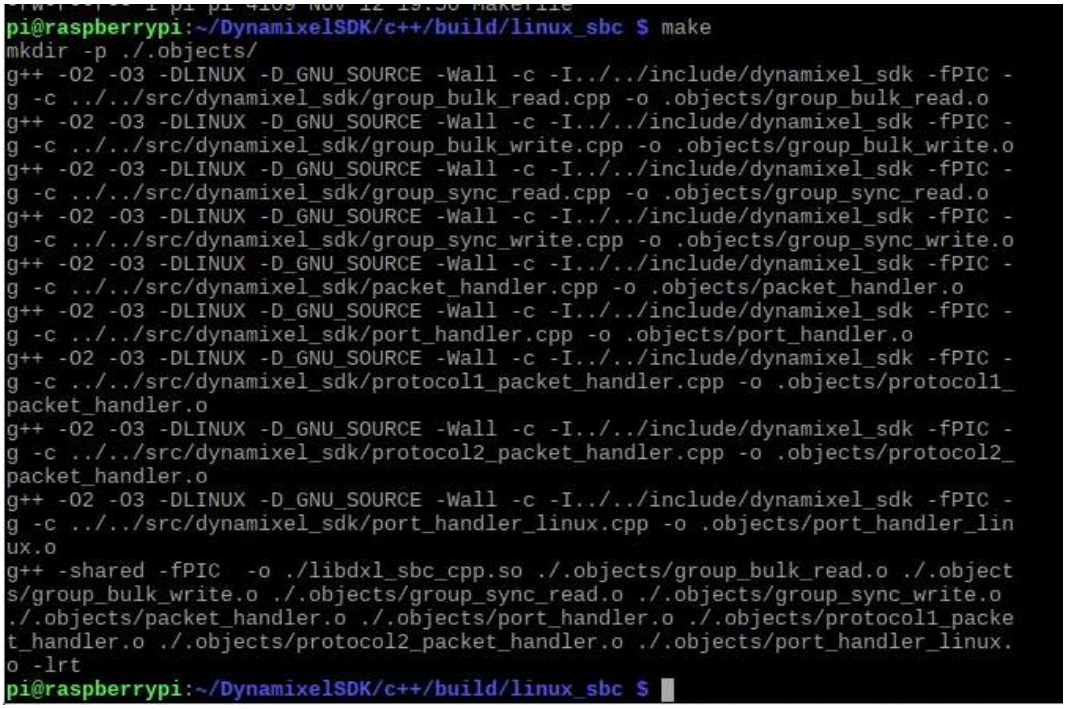
Ahora vamos a compilar la librería de C++, pues es el lenguaje que maneja QT CREATOR. Esto lo haremos accediendo a la carpeta /DynamixelSDK/c++/build y entrando a “linux sbc”



[A58]

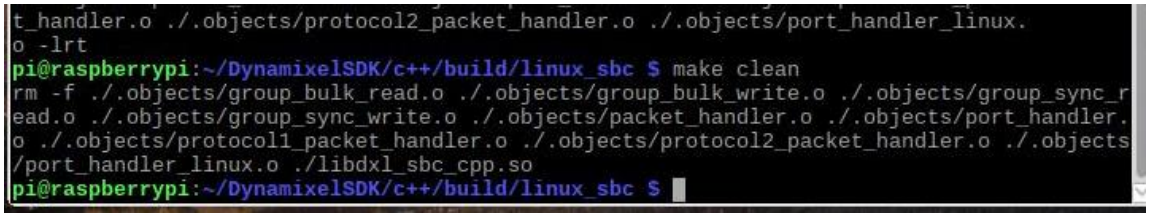
Una vez dentro vamos a hacer uso del comando make para compilar el archivo “Makefile” que contiene la carpeta, lo siguiente puede resultar algo repetitivo pero es para asegurarse que todo sea compilado de manera correcta.

$make



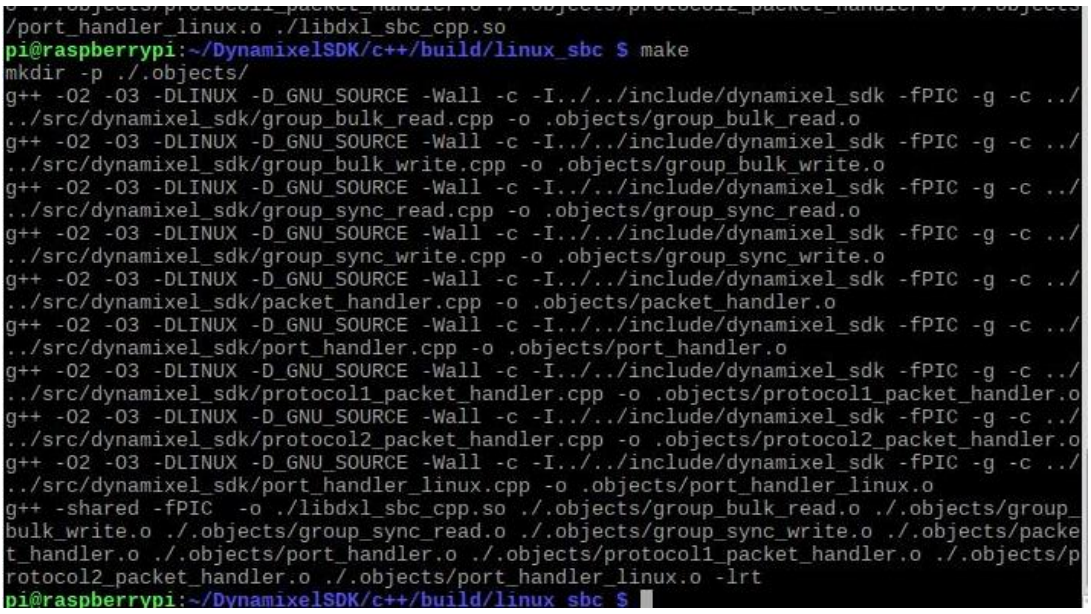
[A59]

$make clean



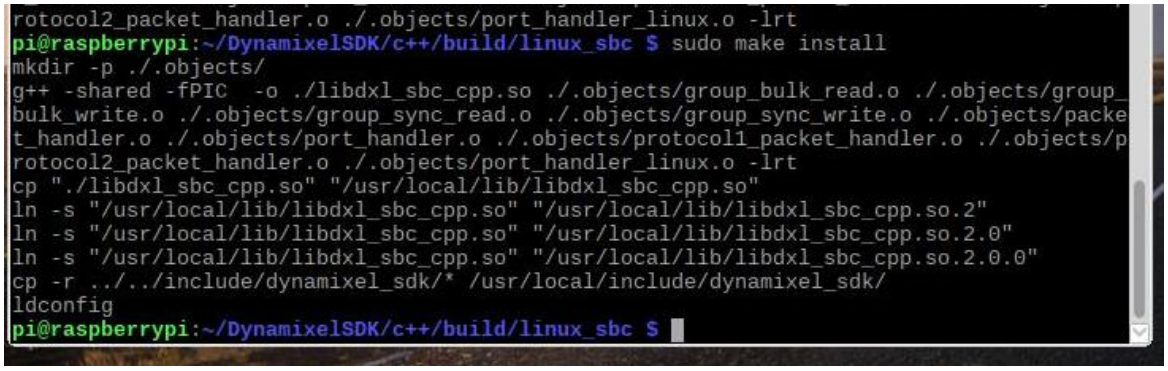
[A60]

$make



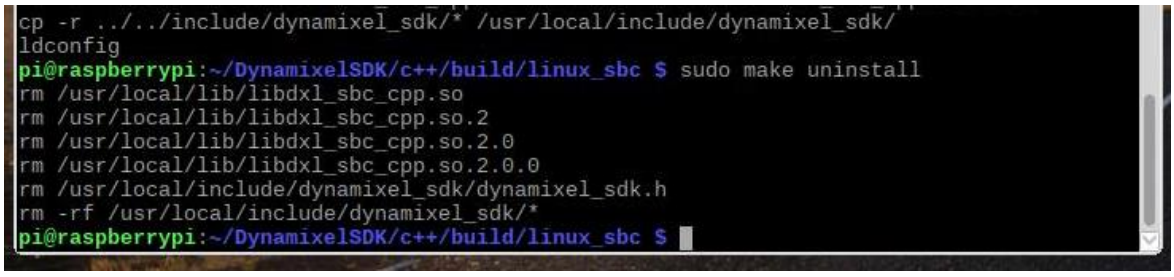
[A61]

$sudo make install



[A62]

$sudo make uninstall



[A63]

$sudo make install

```
rm -rf /usr/local/include/dynamixel_sdk/*
pi@raspberrypi:~/DynamixelSDK/c++/build/linux_sbc $ sudo make install
mkdir -p ./.objects/
g++ -shared -fPIC  -o ./libdxl_sbc_cpp.so ./.objects/group_bulk_read.o ./.objects/group_
bulk_write.o ./.objects/group_sync_read.o ./.objects/group_sync_write.o ./.objects/packe
t_handler.o ./.objects/port_handler.o ./.objects/protocol1_packet_handler.o ./.objects/p
rotocol2_packet_handler.o ./.objects/port_handler_linux.o -lrt
cp "./libdxl_sbc_cpp.so" "/usr/local/lib/libdxl_sbc_cpp.so"
ln -s "/usr/local/lib/libdxl_sbc_cpp.so" "/usr/local/lib/libdxl_sbc_cpp.so.2"
ln -s "/usr/local/lib/libdxl_sbc_cpp.so" "/usr/local/lib/libdxl_sbc_cpp.so.2.0"
ln -s "/usr/local/lib/libdxl_sbc_cpp.so" "/usr/local/lib/libdxl_sbc_cpp.so.2.0.0"
cp -r ../../include/dynamixel_sdk/* /usr/local/include/dynamixel_sdk/
ldconfig
pi@raspberrypi:~/DynamixelSDK/c++/build/linux_sbc $
```

[A64]

Y verificamos que exista ahora el archivo compilado.

```
pi@raspberrypi: ~/DynamixelSDK/c++/build/linux_
File  Edit  Tabs  Help
pi@raspberrypi:~/DynamixelSDK/c++/build/linux_sbc $ ls -l
total 704
-rwxr-xr-x 1 root root 711796 Nov 12 20:10 libdxl_sbc_cpp.so
-rw-r--r-- 1 pi   pi     4109 Nov 12 19:50 Makefile
pi@raspberrypi:~/DynamixelSDK/c++/build/linux_sbc $
```

[A65]

Ahora estamos listos para hacer uso de Dynamixel en nuestro proyecto de QT CREATOR con la siguiente instrucción dentro del proyecto (en el archivo .pro).

La forma de hacer la llamada a las bibliotecas de Dynamixel es con los comandos LIBS e INCLUDEPATH

LIBS += -ldxl_sbs_cpp

INCLUDEPATH += /home/pi/DynamixelSDK/c++/include/

```
25
26 SOURCES += main.cpp\
27         inicial.cpp
28
29 HEADERS  += inicial.h
30
31 FORMS    += inicial.ui
32
33 LIBS  += -ldxl_sbc_cpp
34 INCLUDEPATH += /home/pi/DynamixelSDK/c++/include/
35
```

[A66]

5.2.2 QT CREATOR durante el desarrollo.

Durante el desarrollo del proyecto, realicé varias pruebas previas al control de Dynamixel, por ejemplo el clásico “Hola Mundo”, para esto únicamente hice la impresión en pantalla de la frase “Hola Mundo”, todo fue programado en C++. QT CREATOR nos facilita múltiples formas de visualizar un programa, ya sea en la consola común y con una interfaz gráfica. Primeramente aprendí a desarrollar aplicaciones en consola, simples ciclos FOR, condiciones IF, WHILE entre otras palabras reservadas. Fueron programas muy simples pero quede satisfecho con los resultados arrojados por QT CREATOR, únicamente encontré un inconveniente, el hecho de estar agregando el compilador a utilizar cada vez que se abría nuevamente el programa. No representaba un gran inconveniente en el desarrollo de las aplicaciones pero suele ser algo molesto después de algún tiempo. En cuanto a la interfaz inicial de QT CREATOR es sumamente amigable, fácil de entender, y si ya se tiene algo de experiencia en IDE’s de este estilo, es completamente igual a manejar cualquiera. Personalmente, me ha gustado mucho manejar interfaces de desarrollo como esta, y por ello decidí comenzar a utilizar la herramienta para creación de Interfaces de Usuario. Me acomode muchísimo a la modalidad de arrastra y pega, bastante simple y dinámica, con todas las opciones en la paleta de opciones de cada elemento seleccionado e incrustado. La forma de generar eventos a los objetos colocados en la UI es simple, basta con un clic derecho y la selección de la acción a realizar, como puede ser un Clic a un botón o el cambio de un texto en un Label. Me sorprendió bastante la simpleza de desarrollo con este QT.

Poco a poco fui desarrollando más y más programas, cada vez más complejos, hasta llegar a hacer uso de una librería conocida como WIRING PI, con esta podemos utilizar el PINOUT de la Raspberry, y así comenzar a integrar software y hardware, con una resistencia y un led, pude programar el encendido y apagado de este, haciendo uso de QT. Poco después hice la lectura de un switch físico, con un arreglo simple de resistores y un botón, utilizando el PINOUT, la librería WIRING PI y QT CREATOR, desarrollando una interfaz con un Label y el texto “Sin Presionar” que al momento de recibir la lectura del botón mediante el PINOUT de la Raspberry cambiaba a “Presionado”. Puede parecer simple, pero fue un gran avance para continuar aprendiendo de la lectura de sensores y manipulación de elementos físicos con la programación realizada en QT.

Siguiendo con las investigaciones fue que comencé a probar formas de comunicación con los servomotores y motores de CD, displays y muchos más artefactos de electrónica. Simplemente me maravillé con el uso de QT CREATOR, WIRING PI, RASPBERRY y la gran cantidad de elementos de electrónica que se tienen.

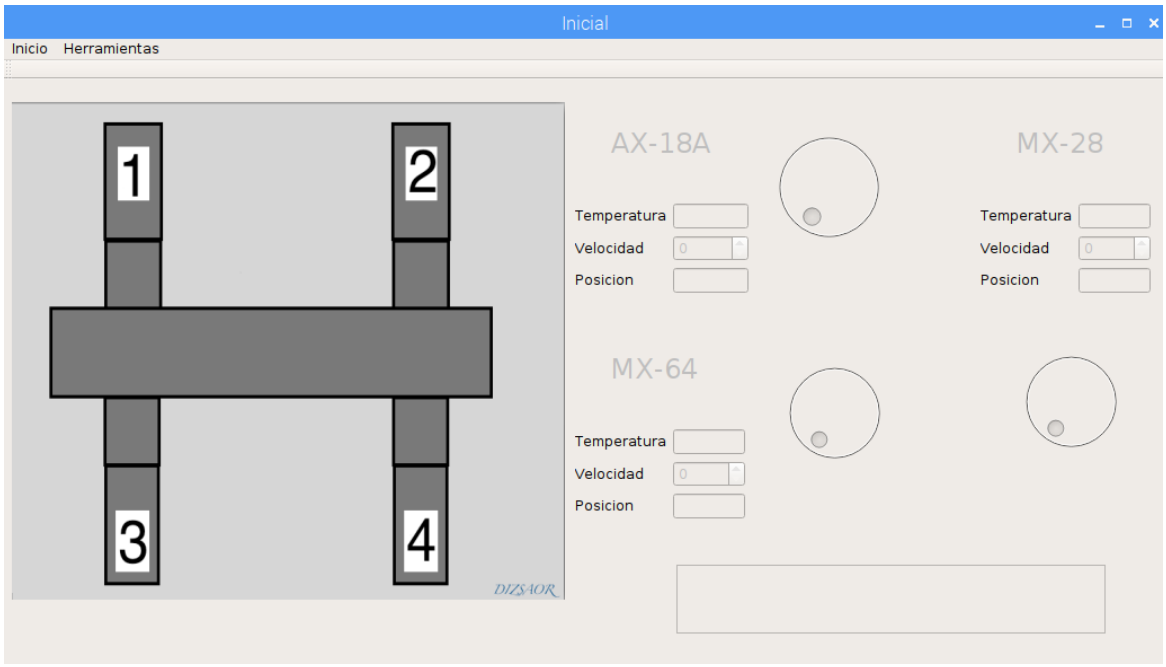
Aprendí a manejar los protocolos de I2C y SPI con la ayuda de WIRING PI pero fue cuando entendí cómo funcionan los actuadores de Dynamixel que me vi en un problema.

Dynamixel funciona con TTL y al inicio intenté hacer él envió de datos con el PINOUT, pero resultó en un gran problema pues no pasaba absolutamente nada, y recibir datos así como enviarlos se complicaba mucho en la programación, fue cuando descubrí Dynamixel SDK, el cual facilito mucho el envío y recepción de datos ya que no era necesario hacerlo mediante el PINOUT si no con el puerto USB de la Raspberry y con ayuda del USB2DYNAMIXEL.

Esto ayudaba en dos aspectos, tanto a nivel de software como a nivel de hardware. En software, las bibliotecas que contiene DynamixelSDK nos otorga una serie de métodos para el envío de los datos y de igual forma contiene métodos para la recepción, de esta forma tenemos el control y monitoreo de los actuadores que conforman el brazo. En hardware, nos brinda seguridad pues al hacer uso de USB2DYNAMIXEL no corremos el riesgo por la manipulación del PINOUT de la Raspberry, personalmente esto me tranquilizó mucho, pues ya había cometido el error de una mala conexión con los actuadores y termine dañando una Raspberry.

5.2.3 Ejecución.

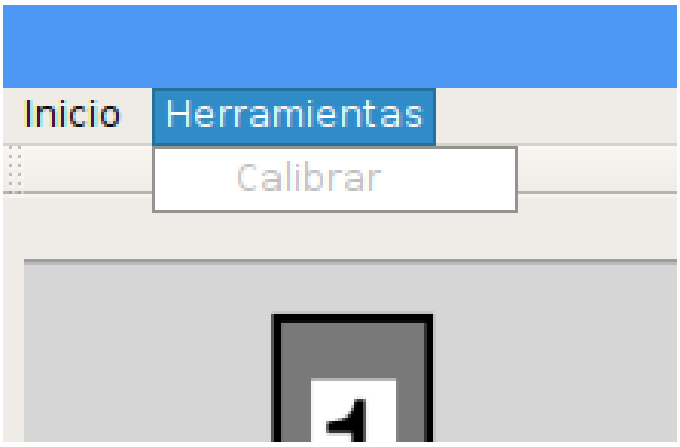
Inicialmente, el programa nos muestra la siguiente interfaz:



[A67]

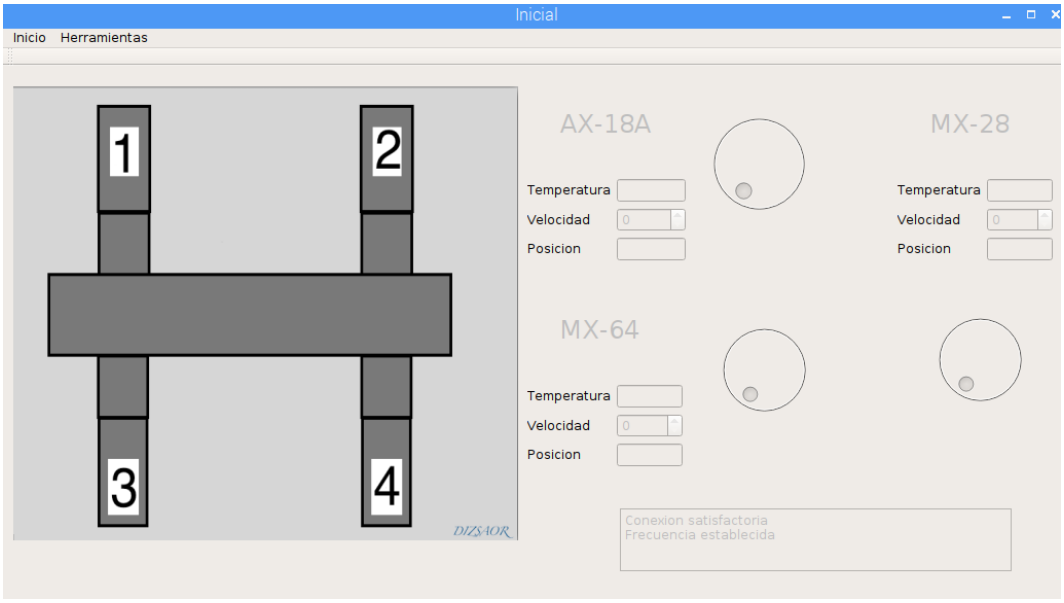
Donde existen elementos que no pueden ser utilizados hasta seleccionar el evento “Conectar”.

[A68]



[A69]

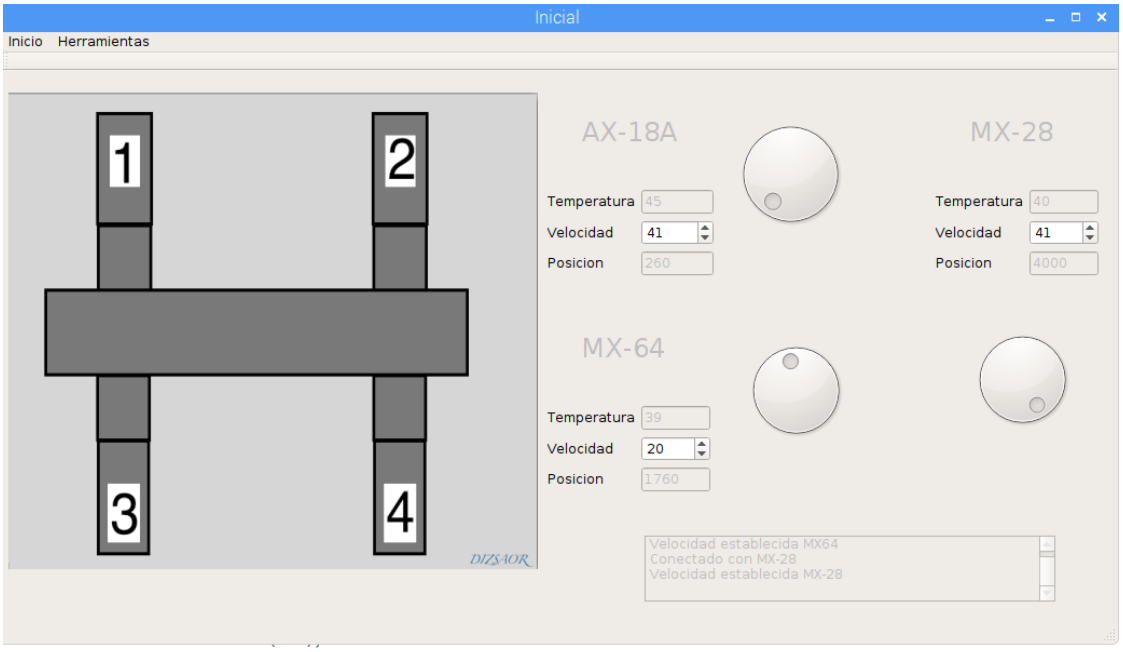
Una vez conectado el sistema nos mostrara el siguiente mensaje:



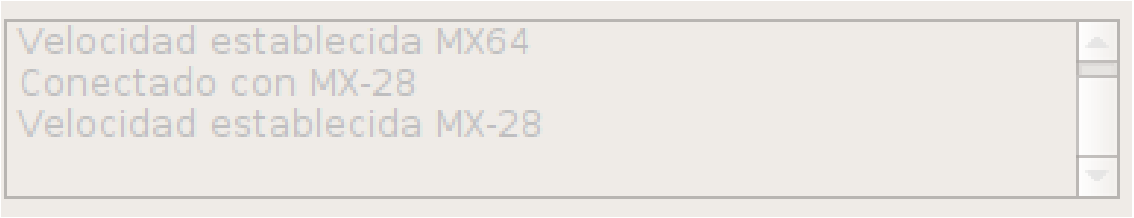
[A70]

[A71]

Ya conectado, procedemos a calibrar:



[A72]



[A73]

En la imagen anterior, podemos observar la conexión satisfactoria con los actuadores.

Para calibrar cada uno de los actuadores (AX18A, MX64, MX28), se modifica su velocidad en valores que el actuador entiende (41,20 y 41 respectivamente). De igual forma se modifica su posición en valores que cada actuador entiende (260, 1760 y 4000 respectivamente), dichos valores fueron calculados a partir de las limitaciones físicas que poseen los elementos en conjunto. Cada actuador tiene un rango máximo y mínimo tanto en velocidad como en posición.

Para el caso del actuado AX18A su velocidad puede ser manejada de 10 a 50 y su posición de 255 hasta 765.

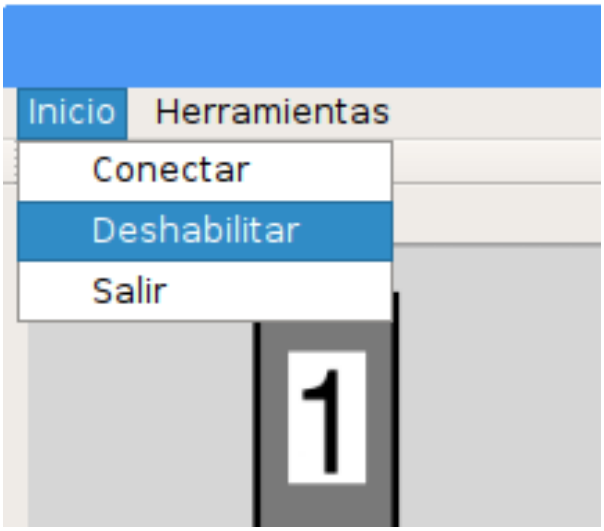
En el actuador MX64 tenemos valores en velocidad desde 10 hasta 60 y posiciones desde 886 hasta 2767.

Para el actuador MX28 manejamos valores de 10 a 60 (como MX64) y su posición maneja el rango de 1100 a 4000.

Todos estos valores han sido sometidos a pruebas y considero son los más adecuados. Hasta el momento se maneja todo en lenguaje de actuadores, considerando en las siguientes versiones del programa, manejarlo por grados y porcentajes ambos campos.

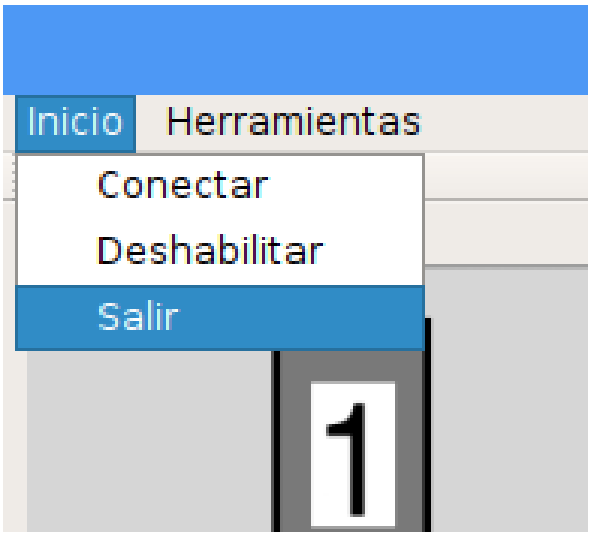
El monitoreo de la temperatura se lleva a cabo al realizar el cambio en la posición de algún actuador. Independientemente de cual actuador sea el manipulado, todos serán consultados para obtener su temperatura.

Cuando se desee concluir con la manipulación de los actuadores, bastará con ir a la pestaña de "Inicio" y seleccionar "Deshabilitar". Esto deshabilita el torque en todos los actuadores.



[A74]

Una vez deshabilitado el torque, podemos proceder a cerrar el programa en "Inicio" y seleccionando "Salir".



[A75]

Ahora se presentará el código del programa (solo aquellas partes consideradas importantes).

5.2.3.1 Servomotor.h

Este archivo es la cabecera a ocupar en inicial.h. Contiene las referencias a métodos y atributos a ocupar en los objetos.

```
#ifndef SERVOMOTOR_H
#define SERVOMOTOR_H
```

```
#include "dynamixel_sdk.h"

class Servomotor
{
public:
    Servomotor();
    Servomotor(dynamixel::PortHandler         *portHandler,dynamixel::PacketHandler
*packetHandler,int identificador);
    void set_Torque(int enable);
    int get_Torque();
    int get_Posicion();
    void set_Posicion(int p);
    int get_Temperatura();
    int get_Velocidad();
    void set_Velocidad(int v);
    ~Servomotor();

private:
    int id=0;
    uint16_t temperatura=0;
    int torque;
    int velocidad=0;
    int posicion=0;
    int dxl_comm_result = COMM_TX_FAIL;
    uint8_t dxl_error = 0;
    uint16_t dxl_present_position = 0;
    dynamixel::PortHandler *portHandler;
    dynamixel::PacketHandler *packetHandler;
};

#endif // SERVOMOTOR_H
```

5.2.3.2 Inicial.h

En este apartado se hace la declaración de las variables a utilizar (objetos).

```
private:
    Ui::Inicial *ui;
    dynamixel::PortHandler                      *portHandler                      =
dynamixel::PortHandler::getPortHandler(DEVICEFREC);
    dynamixel::PacketHandler                    *packetHandler                    =
dynamixel::PacketHandler::getPacketHandler(PROTOCOL_VERSION);
    Servomotor *ax18a;
    Servomotor *mx64;
    Servomotor *mx28;
```

5.2.3.3 Servomotor.cpp

Ahora veremos el código completo, los métodos a utilizar por los objetos “Servomotor”.

```
#include "servomotor.h"
#include "ui_inicial.h"
#include "inicial.h"

#define ADDR_MX_TORQUE_ENABLE           24
#define ADDR_MX_GOAL_POSITION           30
```

```
#define ADDR_MX_PRESENT_POSITION        36
#define ADDR_VEL                        32
#define ADDR_TEMP                       43


    Servomotor::Servomotor(){
    }
    Servomotor::Servomotor(dynamixel::PortHandler
*portHandler,dynamixel::PacketHandler *packetHandler,int identificador){
        this->packetHandler=packetHandler;
        this->portHandler=portHandler;
        this->id=identificador;
    }
    void Servomotor::set_Torque(int enable){
        try{
            dxl_comm_result    =    packetHandler->write1ByteTxRx(portHandler,    id,
ADDR_MX_TORQUE_ENABLE, enable, &dxl_error);
            if (dxl_comm_result != COMM_SUCCESS){
              throw 0;
            }
            else if (dxl_error != 0){
              throw 0;
            }
            else{
              torque=1;
            }
        }catch(int e){
            torque=e;
        }
    }

    int Servomotor::get_Torque(){
        return torque;
    }

    void Servomotor::set_Velocidad(int v){
        try{
            dxl_comm_result     =     packetHandler->write2ByteTxRx(portHandler,id,
ADDR_VEL, v, &dxl_error);
            if (dxl_comm_result != COMM_SUCCESS){
              packetHandler->printTxRxResult(dxl_comm_result);
              throw 1;
            }
            else if (dxl_error != 0)
            {
              packetHandler->printRxPacketError(dxl_error);
              throw 1;
            }else{
              velocidad=v;
            }
        }catch(int e){
            velocidad=e;
        }
    }

    int Servomotor::get_Velocidad(){
        return velocidad;
    }

    void Servomotor::set_Posicion(int p){
        try{
            dxl_comm_result     =     packetHandler->write2ByteTxRx(portHandler,id,
ADDR_MX_GOAL_POSITION, p, &dxl_error);
            if (dxl_comm_result != COMM_SUCCESS){
```

```
                packetHandler->printTxRxResult(dxl_comm_result);
                throw 0;
            }
            else if (dxl_error != 0){
                packetHandler->printRxPacketError(dxl_error);
                throw 0;
            }else{
                posicion=p;
            }
        }catch(int e){
            posicion=e;        }

    }
    int Servomotor::get_Temperatura(){
        try{
            dxl_comm_result    =    packetHandler->read2ByteTxRx(portHandler,    id,
ADDR_TEMP, &temperatura, &dxl_error);
            if (dxl_comm_result != COMM_SUCCESS)
            {
              packetHandler->printTxRxResult(dxl_comm_result);
              throw 0;
            }
            else if (dxl_error != 0)
            {
              packetHandler->printRxPacketError(dxl_error);
              throw 0;

            }else return temperatura;
        }catch(int e){
            return e;
        }
    }
    int Servomotor::get_Posicion(){
        return posicion;
    }
    Servomotor::~Servomotor(){
    }
```

5.2.3.4 Inicial.cpp

En este archivo tenemos las direcciones de memoria pertenecientes a los ID de cada actuador Dynamixel, así como la frecuencia para envío de datos los actuadores.

```
#define DXL_ID_AX18A                    3               // ID DEL DINAMIXEL AX-18A
#define DXL_ID_MX64                     2               // ID DEL DYNAMIXEL MX64
#define DXL_ID_MX28                     1               // ID DEL DYNAMIXEL MX64
#define BAUDRATE                        1000000         //Frecuencia serial para
envio de datos
```

Aquí verifico que el dispositivo USB2 y establezco comunicación con él.

```
void Inicial::on_actionConectar_triggered(){

    //Elementos para comunicacion
    if (portHandler->openPort()){
      ui->consola->appendPlainText("Conexion satisfactoria");
      if (portHandler->setBaudRate(BAUDRATE)){
```

```
        ui->consola->appendPlainText("Frecuencia establecida");
        ui->actionCalibrar->setEnabled(true);
        ui->actionDeshabilitar->setEnabled(true);
      }else{
        ui->consola->appendPlainText("Error al establecer frecuencia");
      }
    }else{
      ui->consola->appendPlainText("Error al encontrar Dynamixel USB2");
    }
}
```

En este apartado del programa se instancian los objetos Servomotor, y se procede a calibrar cada uno.

```
void Inicial::on_actionCalibrar_triggered(){

    //Objetos Servomotor
    ax18a= new Servomotor(portHandler,packetHandler,DXL_ID_AX18A);
    mx64= new Servomotor(portHandler,packetHandler,DXL_ID_MX64);
    mx28 = new Servomotor(portHandler,packetHandler,DXL_ID_MX28);

    ax18a->set_Torque(1);
    if(ax18a->get_Torque()==1){
        ui->consola->appendPlainText("Conectado con AX-18A");
        ax18a->set_Velocidad(velEstandar);
        if(ax18a->get_Velocidad()==velEstandar){
            ui->consola->appendPlainText("Velocidad establecida AX-18A");
            ui->spinVel_18a->setValue(ax18a->get_Velocidad());
            ui->spinVel_18a->setEnabled(true);
            ax18a->set_Posicion(260);
            if(ax18a->get_Posicion()<=260){
                ui->linePos_18a->setText(QString::number(ax18a->get_Posicion()));
                ui->dial_18a->setEnabled(true);
                ui->dial_18a->setValue(ax18a->get_Posicion());
            }else{
                ui->linePos_18a->setText("Error");
            }
        }else{
            ui->consola->appendPlainText("Error al cambiar Velocidad");
        }
    }else{
        ui->consola->appendPlainText("Error al conectar con AX-18A");
    }

    mx64->set_Torque(1);
    if(mx64->get_Torque()==1){
        ui->consola->appendPlainText("Conectado con MX-64");
        mx64->set_Velocidad(20);
        if(mx64->get_Velocidad()==20){
            ui->consola->appendPlainText("Velocidad establecida MX64");
            ui->spinVel_64->setValue(mx64->get_Velocidad());
            ui->spinVel_64->setEnabled(true);
            mx64->set_Posicion(1760);
            if(mx64->get_Posicion()<=1760){
                ui->linePos_64->setText(QString::number(mx64->get_Posicion()));
                ui->dial_64->setEnabled(true);
                ui->dial_64->setValue(mx64->get_Posicion());
            }else{
                ui->linePos_64->setText("Error");
            }
        }else{
            ui->consola->appendPlainText("Error al cambiar Velocidad MX-64");
```

```
        }
    }else{
        ui->consola->appendPlainText("Error al conectar con MX-64");
    }
    mx28->set_Torque(1);
    if(mx28->get_Torque()==1){
        ui->consola->appendPlainText("Conectado con MX-28");
        mx28->set_Velocidad(velEstandar);
        if(mx28->get_Velocidad()==velEstandar){
            ui->consola->appendPlainText("Velocidad establecida MX-28");
            ui->spinVel_mx28->setValue(mx28->get_Velocidad());
            ui->spinVel_mx28->setEnabled(true);
            mx28->set_Posicion(4000);
            if(mx28->get_Posicion()<=4000){
                ui->linePos_28->setText(QString::number(mx28->get_Posicion()));
                ui->dial_28->setEnabled(true);
                ui->dial_28->setValue(mx28->get_Posicion());
            }else{
                ui->linePos_28->setText("Error");
            }
        }else{
            ui->consola->appendPlainText("Error al cambiar Velocidad MX-28");
        }
    }else{
        ui->consola->appendPlainText("Error al conectar con MX-28");
    }

}
```

Cuando se hace un cambio de valor en el dial (rueda en la interfaz gráfica). El actuador correspondiente al dial cambiara su posición. El código encargado de esto es (el código solo afecta al actuador AX18A):

```
void Inicial::on_dial_18a_valueChanged(int value){
    if(ax18a->get_Velocidad()>=10  &&  ax18a->get_Velocidad()<=50){  //verifico  la
velocidad que este iniciada y sea menor a 50
        ax18a->set_Posicion(value);
        if(ax18a->get_Posicion()==value){
            ui->linePos_18a->setText(QString::number(ax18a->get_Posicion()));
        }else{
            ui->linePos_18a->setText("Error");
        }
    }else{
        ui->consola->appendPlainText("Verificar la velocidad de AX-18A (10-50)");
    }
    sensar_temp();
}
```

Al realizar el cambio de velocidad desde la interfaz gráfica se ejecuta el siguiente código (el código solo afecta al actuador AX18A).

```
void Inicial::on_spinVel_18a_valueChanged(int arg1){
    ax18a->set_Velocidad(arg1);
}
```

Cuando seleccionamos “Deshabilitar” en la interfaz gráfica.

```
void Inicial::on_actionDeshabilitar_triggered(){
    ax18a->set_Torque(0);
    if(ax18a->get_Torque()==1){
        ui->consola->appendPlainText("Torque deshabilitado AX18A");
        delete ax18a;
    }else{
        ui->consola->appendPlainText("Error al deshabilitar torque AX18A");
    }

    mx64->set_Torque(0);
    if(mx64->get_Torque()==1){
        ui->consola->appendPlainText("Torque deshabilitado MX-64");
        delete mx64;
    }else{
        ui->consola->appendPlainText("Error al deshabilitar torque MX-64");
    }
    mx28->set_Torque(0);
    if(mx28->get_Torque()==1){
        ui->consola->appendPlainText("Torque deshabilitado MX-28");
        delete mx28;
    }else{
        ui->consola->appendPlainText("Error al deshabilitar torque MX-28");
    }

}
```

Este código es utilizado para censar la temperatura de cada actuador y mostrarlo en la interfaz gráfica.

```
void Inicial::sensar_temp()
{
  ui->lineTemp_18a->setText(QString::number(ax18a->get_Temperatura()));
  ui->lineTemp_64->setText(QString::number(mx64->get_Temperatura()));
  ui->lineTemp_28->setText(QString::number(mx28->get_Temperatura()));
}
```

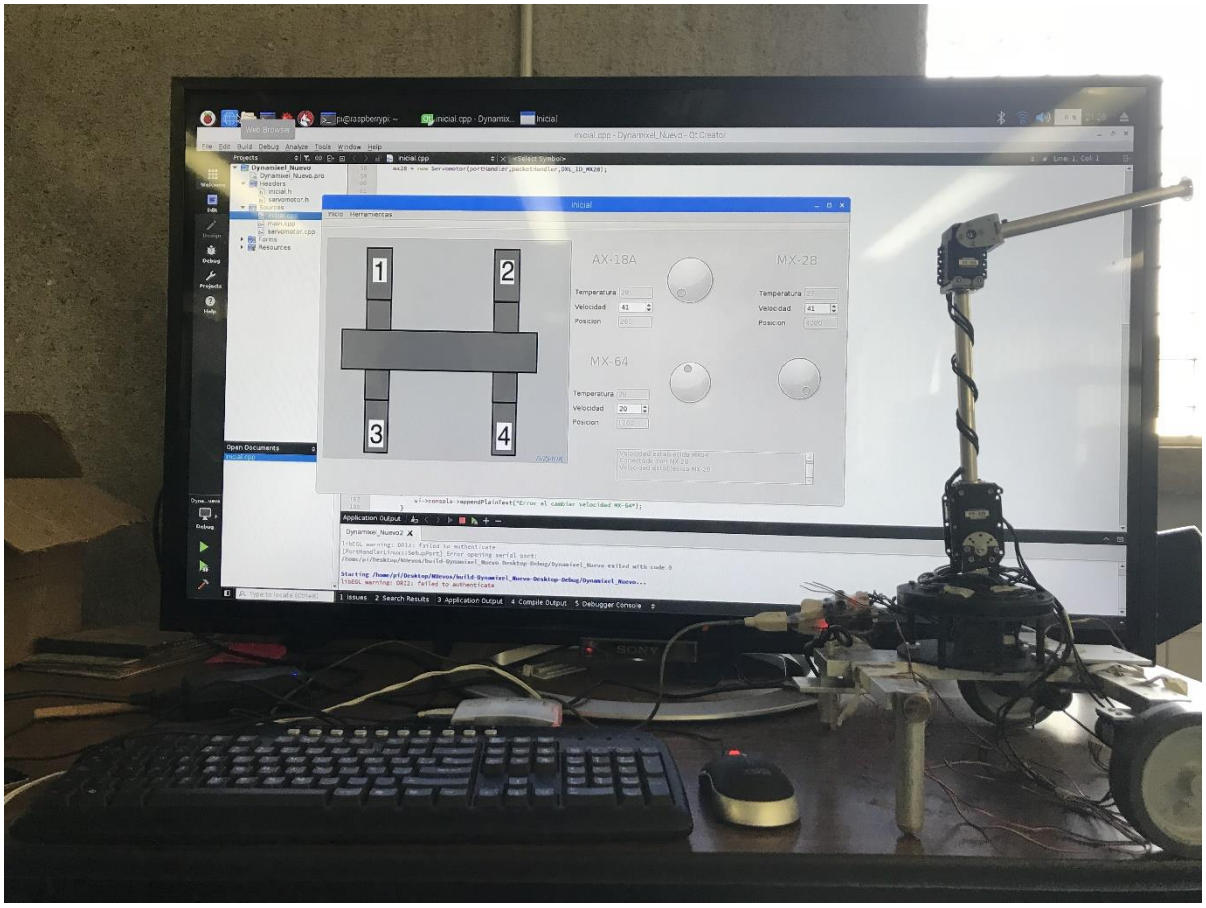
Para concluir se presentan algunas imágenes del sistema en funcionamiento:

[A76]

En las imágenes anteriores se muestra la conectividad entre Raspberry Pi, el controlador USB2 y el regulador de corriente entre USB2 y los actuadores.

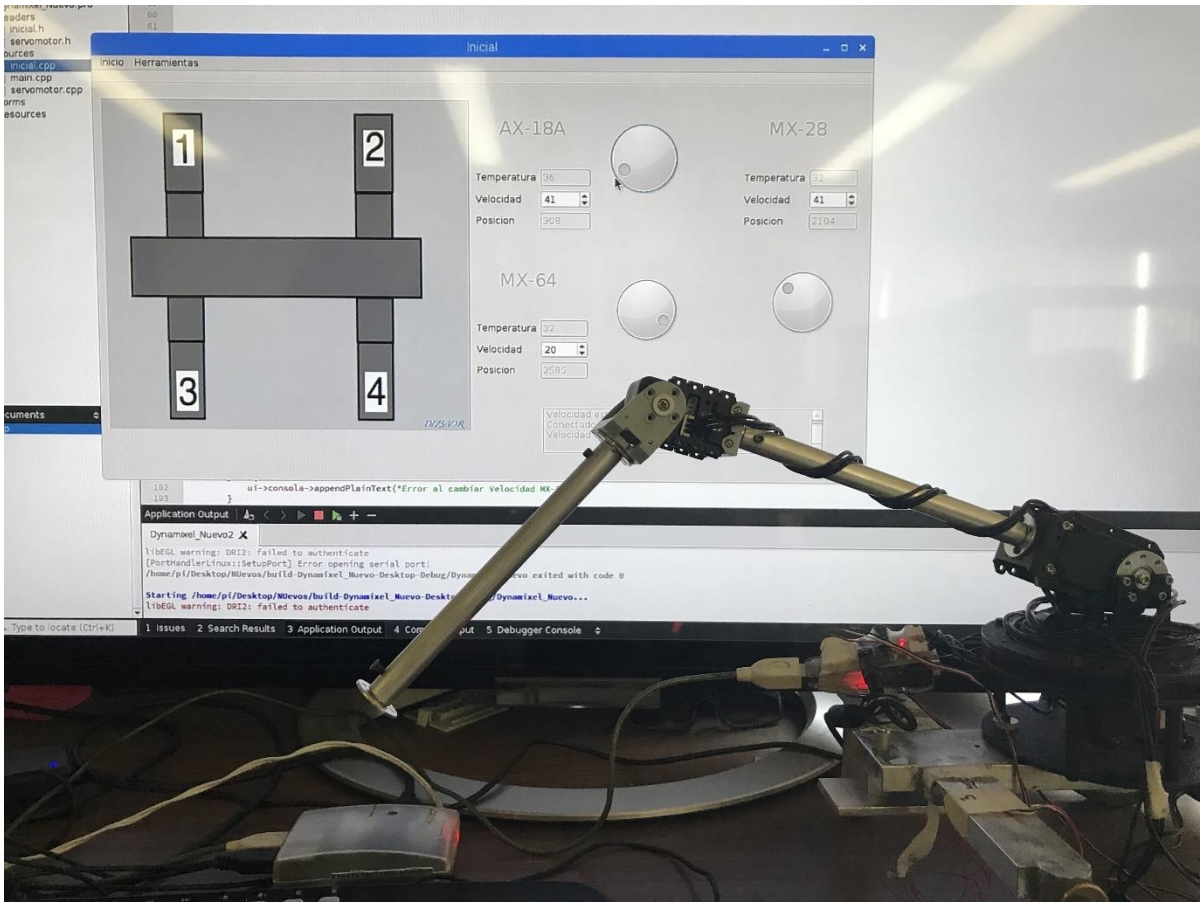
USB2 es un poco más grande que los puertos machos comunes de USB, por lo tanto es necesario emplear una extensión. Tanto USB2 como el regulador de corriente se encuentran unidos por cinta tipo masking tape.

Con ayuda de la interfaz gráfica se logró tener el control del brazo, en la punta se tiene un movimiento aproximado de 180º (actuador AX18A). La parte central tiene libertades de aproximadamente 3º hasta 177º, para no tener contacto con la base, pues esto puede generar daños a la estructura que soporta al actuador MX64. Y por último, la base, la cual cuenta con un movimiento cercano a los 270º aproximadamente.



[A77]

En la imagen previa obcervamos la interfaz grafica y el brazo robótico una vez calibrado.



[A78]

Aquí se muestra la interacción con el brazo, modificando los valores resultantes de haber calibrado, hasta tener los valores deseados. En este caso AX18A, MX64 y MX28 cambian sus valores de posición a 308, 2585 y 2104 respectivamente. Aproximadamente, AX18A a 73º, MX64 a 165º y MX28 a 142º.

Este proyecto tomo poco más de un año en ser realizado y aun no se concluye, falta la adquisición de datos por parte de las galgas. Desafortunadamente, eso ya no estaba en mis manos.

6) Resultados y trabajo futuro.

Los resultados obtenidos con el sistema han sido satisfactorios, tanto para software como para hardware, se tiene el control del brazo mecánico, así como el monitoreo tanto de la temperatura, su posicionamiento y la velocidad a la cual se mueve cada actuador Dynamixel. Sin embargo un punto a mejorar es la comunicación con las galgas. Momentáneamente la comunicación con la base del brazo es nula, por lo tanto la volcadura de este solo podrá ser evitada manualmente, limitando la velocidad y posición del brazo.

Como parte de los trabajos futuros del presente proyecto se puede dividir en tres fases en la primera requiere de una limpieza de código, agregar otras funciones para monitorear diferentes aspectos de los actuadores. Se planea utilizar el display táctil de Raspberry, para ello adecuaré la interfaz haciéndola adaptable con dicho hardware.

La segunda fase consta del monitoreo de las galgas extensiométricas, teniendo el dispositivo especializado para la interacción intermedia entre las Raspberry y las galgas, buscaré las herramientas adecuadas para interactuar con él para posteriormente implementar en la interfaz. Esto pude tomar más tiempo, pues el conocimiento por mi parte de estos artefactos (galgas) es muy limitado.

En la tercera fase, pienso generar la interfaz de QTCREATOR a un dispositivo móvil, compartir las señales a través del Bluetooth de Raspberry y a su vez este podrá controlar el brazo sin ningún problema.

Más allá de eso me gustaría en algún momento diseñar sistemas inteligentes para hogares o edificios completos, enfocándome a la domótica, así como desarrollar robótica en el área de la salud, crear robots ayudantes como enfermeros, o inclusive prótesis para quienes lo necesiten. Me gustaría expandir más el conocimiento de la ingeniería en computación, no solo déjalo en la parte de la programación, alimentarlo con microcontroladores y microprocesadores, electrónica, entro otros dispositivos para ayudar a mejorar la calidad de vida de todas las personas posibles.

7) Conclusiones.

El trabajo me llevo más allá de lo que pensaba; al comienzo cuando creí haber limitado los temas me di cuenta que se requería de más información para llegar a transmitir el mensaje deseado. Siempre he sido una persona a quien le gusta buscar y entender el porqué de las cosas, sus orígenes, como iniciaron, como funcionan y el cómo se encuentra en la actualidad. Es por eso que este trabajo lo lleve hasta los inicios del cómputo, dando una vista general, quizá no tan detallada como me hubiera encantado, pero la información adquirida fue impresionante. En la carrera de Ingeniería en Computación nos enseñan una parte de la historia del cómputo, más no todo. Considero que conocer los inicios del cómputo es fundamental para entender el funcionamiento de las máquinas actuales, en materias como Introducción a la Ingeniería en Computación debería existir un apartado dedicado a la historia del cómputo y una comparativa con las máquinas actuales (incluyendo SBC). Con el motivo de enriquecer este trabajo también aparte una sección para la electrónica, pues gracias a ella tenemos los kits de electrónica, pero para enfocarme a ese tema como tal, tuve que retroceder a los inicios de la electricidad. Para mí que no soy Ingeniero en Electrónica se complicó un poco el tema, a pesar de llevar materias de electricidad y electrónica, no es lo mismo que estar investigando. De igual forma propondría más laboratorios de electrónica en la carrera con profesores ya experimentados en el tema. No solo limitar a los estudiantes en el uso de microcontroladores antiguos tipo PIC, fomentar el uso de sistemas SBC, crear sus propios circuitos e integrarlos a estos.

La sección de Domótica me gustó mucho pues ya tuve la oportunidad de participar en un pequeño proyecto referente a esto y no conocía mucho del tema, mucho menos de los inicios de la domótica, los protocolos, etc. Y para el apartado de Robótica fue cuando me emocione más, pues este tema es de mis favoritos, igualmente investigar la historia de ella, me encanto. Aunque me faltó redactar más información sobre estos temas, me siento cómodo del como quedo ese capítulo.

El participar en la elaboración de este sistema ha sido de las mejores oportunidades que he tenido hasta este momento, el lograr manipular un sistema mecánico/electrónico con un dispositivo que jamás había utilizado.

Este proyecto me ayudo a mejorar como ingeniero pues adquirí conocimiento no solo aplicable al cómputo, también a la mecánica y a la electrónica. Aprender más sobre la programación orientada a objetos con el lenguaje C++, en interfaces que no conocía ha sido una muy grata experiencia. Conocer mucho más sobre microcontroladores, y sobre computadoras de una sola placa (SBC) aumento más mi interés sobre la electrónica y la robótica sin mencionar la domótica.

Como Ingenieros en Computación debemos saber cómo resolver los problemas referentes al cómputo, pero eso no debe limitarnos simplemente al aspecto del software o cambiar ciertas partes de la computadora para hacerla funcionar, debemos ver más allá e innovar en el hardware para hacer más eficiente el software. Entender la relación entre ambos y como integrarlos de una forma más efectiva.

Con este trabajo, busqué enseñar un poco acerca de la relación electrónica con elementos de software, adquirir datos a partir de la electrónica y manipularlos con un programa, para tener una respuesta o acción a realizar.

8) Referencia de imágenes y tablas.

A1: Imagen 1.1 Diagrama del uso de tarjetas perforadas. Elaboración Propia.

A2: Imagen 1.2 Diagrama a bloques del funcionamiento de una computadora de segunda generación. Elaboración Propia.

A3: Imagen 1.3 Flujo ordenado de electrones. Elaboración Propia.

A4: Imagen 1.4 Circuito eléctrico. Elaboración Propia.

A5: Imagen 1.10 Representación en circuito de una compuerta AND. Elaboración Propia.

A6: Imagen 1.12 Representación en circuito de una compuerta OR. Elaboración Propia.

A7: Imagen 1.14 Representación en circuito de una compuerta NOT. Elaboración Propia.

A8: Imagen 1.16 Representación en circuito de una compuerta NAND. Elaboración Propia.

A9: Imagen 1.18 Representación en circuito de una compuerta NOR. Elaboración Propia

A10: Imagen 1.23 Símbolo de la compuerta XNOR. Elaboración Propia.

A11: Imagen 1.36 Grabador PICkit 2. Elaboración Propia.

A12: Imagen 3.2 Arduino UNO, vista superior. Elaboración Propia.

A13: Fotografía. Elaboración Propia.

A14: Fotografía. Elaboración Propia.

A15: Fotografía. Elaboración Propia.

A16: Fotografía. Elaboración Propia.

A17: Imagen 3.4 Jumpers. Elaboración Propia.

A18: Imagen 3.5 Arduino Mega. Elaboración Propia.

A19: Imagen 3.6 Zocalo y procesador de Galileo. Elaboración Propia.

A20: Imagen 3.7 JTAG de Galileo. Elaboración Propia.

A21: Imagen 3.8 Galileo vista superior (lado izquierdo), vista inferior (lado derecho). Elaboración Propia.

A22: Captura de pantalla tomada de: https://github.com/ROBOTIS-GIT/DynamixelSDK

A23: Captura de pantalla. Sistema actual.

A24: Imagen 5.1 Raspberri pi 3 Modelo B Elaboración Propia.

A25: Imagen 5.2 Actuador/Servomotor Dynamixel AX-18A con complementos. Elaboración Propia.

A26: Imagen 5.3 Actuador/Servomotor Dynamixel MX-64 con complementos. Elaboración Propia.

A27: Imagen 5.4 Actuador/Servomotor Dynamixel MX-64 con complementos. Elaboración Propia.

A28: Imagen 5.5 Controlador USB2 para comunicación por puerto USB. Elaboración Propia.

A29: Imagen 5.6 Cable de comunicación. Elaboración Propia.

A30: Imagen 5.7 Regulador en el paso de corriente entre el USB2 y los actuadores/servomotores. Elaboración Propia.

A31: Imagen 5.8 Alimentación (vista superior). Elaboración Propia.

A32: Imagen 5.9 Alimentación (vista inferior). Elaboración Propia.

A33: Imagen 5.10 Galga Extensiométrica en la base. Elaboración Propia.

A34: Base del brazo robótico dividido en secciones. Elaboración Propia

A35 - A37: Captura de pantalla. Elaboración Propia.

A38: Captura de pantalla del programa Etcher en ejecución. Elaboración Propia.

A39: Captura de pantalla Raspberry arranque. Elaboración Propia.

A40 – A75: Captura de pantalla. Elaboración Propia.

A76 – A78: Fotografías del sistema en funcionamiento. Elaboración Propia.

B1: AreaTecnologia. (s.f). TRANSISTOR. [Fotografía]. Recuperado de http://www.areatecnologia.com/TUTORIALES/EL%20TRANSISTOR.htm

B2.1: McAllister, Willy. (s.f). El diodo como un elemento de circuito [Dibujo]. Recuperado de https://es.khanacademy.org/science/electrical-engineering/ee-semiconductor-devices/ee-diode/a/ee-diode-circuit-element

B2.2: McAllister, Willy. (s.f). El diodo como un elemento de circuito [Dibujo]. Recuperado de https://es.khanacademy.org/science/electrical-engineering/ee-semiconductor-devices/ee-diode/a/ee-diode-circuit-element

B3: Robotics Plus. (mayo 19, 2017). Resistencia eléctrica [Dibujo]. Recuperado de https://robotics.plus/resistencia-electrica/

B4: Agarwal, T. (s.f). Know all About a Capacitor – Working of a Capacitor [Fotografía]. Recuperado de https://www.elprocus.com/construction-of-capacitor-with-working/

B5.1: Zazzle. (s.f). Regalos Inductor. [Dibujo]. Recuperado de https://www.zazzle.com/inductor+regalos?lang=es

B5.2: Alibaba.com. (s.f). High precision miniature inductor coil for TV and camera. [Fotografía]. Recuperado de https://www.alibaba.com/product-detail/High-precision-miniature-inductor-coil-for_427040439.html

B6: Google Sites. (s.f). Compuertas Lógicas. [Dibujo]. Recuperado de: https://sites.google.com/site/electronicadigitalmegatec/home/compuertas-logicas

B7: Google Sites. (s.f). Compuertas Lógicas. [Dibujo]. Recuperado de: https://sites.google.com/site/electronicadigitalmegatec/home/compuertas-logicas

B8: sistemasumma. (septiembre 9, 2012). Diagrama de circuitos lógicos. [Dibujo]. Recuperado de: https://sistemasumma.com/2012/09/09/diagrama-de-circuitos-logicos/

B9: G36c's Blog. (s.f). Entradas NAND y NOR. [Dibujo]. Recuperado de: https://g36c.wordpress.com/2011/10/06/entradas-nand-y-nor/

B10: Wikipedia. (s.f). Puerta NOR. [Dibujo]. Recuperado de: http://www.wikiwand.com/es/Puerta_NOR

B11: J. (febrero 1, 2012). COMPUERTAS LÓGICAS [Dibujo]. Recuperado de:

http://electronicadepartamentoindustrial.blogspot.mx/2012/02/compuertas-logicas.html

B12: jjbeard. (junio 2, 2006). Porta Lógica Ou-Exclusivo. [Dibujo]. Recuperado de:

https://pt.wikipedia.org/wiki/Porta_L%C3%B3gica_Ou-Exclusivo

B13: J. (febrero 1, 2012). COMPUERTAS LÓGICAS [Dibujo]. Recuperado de:

http://electronicadepartamentoindustrial.blogspot.mx/2012/02/compuertas-logicas.html

B14: Roboprosl. (s.f). Automatización Industrial [Fotografía]. Recuperado de:

https://www.roboprosl.com/cuando-nace-la-automatizacion-industrial-del-telar-autonomo-a-los-automatas-programables/

B15: Chavez, J. (julio 27, 2014). La maquina de Turing. [Dibujo]. Recuperado de:

http://tecnoinfomtia.blogspot.mx/2014/07/la-maquina-de-turing.html

B16: strigoi, D. (septiembre 29, 2009). Arquitectura de von Neumann [Dibujo]. Recuperado de:

https://es.wikipedia.org/wiki/Arquitectura_de_von_Neumann

B17: ENRIMALAGON21. (septiembre 9, 2017). Arquitectura Harvard. [Dibujo]. Recuperado de:

https://www.emaze.com/@AORFQLICI/arquitectura-harvard

B18: lillq. (junio 27, 2012). Borrado de EPROMs con la luz del sol. [Fotografía]. Recuperado de:

https://www.i-ciencias.com/pregunta/20557/borrado-de-eproms-con-la-luz-del-sol

B19: galeon.com. (s.f). 8. MEMORIA ROM, PROM (MEMORIA ROM PROGRAMABLE), EPROM Y EEPROM. [Fotografía]. Recuperado de: http://arqcompunisangil.galeon.com/aficiones2686189.html

B20: Plus Electronics. (s.f). Memoria 27C4001 OTP. [Fotografía]. Recuperado de: http://electronica.com.ve/new/catalog/product_info.php?products_id=1148

B21: culturacion. (s.f). Qué es una memoria Flash. [Fotografía]. Recuperado de:

http://culturacion.com/que-es-una-memoria-flash/

B22: Amidata. (s.f). Memoria NVRAM Maxim DS1230Y-70+, 28 pines, EDIP, 256kbit, 70ns, Montaje en orificio pasante, 4,5 V a 5,5 V. [Fotografía]. Recuperado de: https://es.rs-online.com/web/p/chips-de-memoria-nvram/0132801/

B23: Lanzet, K. (abril 20, 2009). Intel MCS-48. [Fotografía]. Recuperado de: https://es.wikipedia.org/wiki/Intel_MCS-48

B24: Developers-Club. (s.f). Intel 8051. 30 years in devices, instruments and … soft toys. [Fotografía]. Recuperado de: http://developers-club.com/posts/205486/

B25: RS Components Ltd. (s.f). Microchip PIC16LF877A-I/PT, 8bit PIC Microcontroller, 20MHz, 14.3 kB, 256 B Flash, 44-Pin TQFP. [Fotografía]. Recuperado de: https://uk.rs-online.com/web/p/microcontrollers/6533966/

B26: 14030354. (octubre 4, 2016). Procesador. [Dibujo]. Recuperado de: https://www.emaze.com/@AZQRCRRL/Procesador

B27: Chacón, M. (febrero 23, 2015). Convergencia Digital y su impacto frente a la comunicación en las empresas. [Dibujo]. Recuperado de:http://nuevaasignatura2015.blogspot.mx/2015/02/convergencia-digital-y-su-impacto.html

B28: Arduino. (s.f). ARDUINO PRO MINI. [Fotografía]. Recuperado de: https://store.arduino.cc/usa/arduino-pro-mini

B29: S. (agosto 28, 2017). Raspberry Pi 2 & 3 Pin Mappings. [Dibujo]. Recuperado de: https://docs.microsoft.com/en-us/windows/iot-core/learn-about-hardware/pinmappings/pinmappingsrpi

B30: KNX Association. (2014). ¿Qué es KNX?. [Dibujo]. Recuperado de: https://www.knx.org/mx/knx/associacion/que-es-knx/index.php

B31: Work Projects Administration Federal Art Project. (2017). R.U.R. by Karel Čapek 1939.jpg [Dibujo]. Recuperado de https://commons.wikimedia.org/wiki/File:R.U.R._by_Karel_%C4%8Capek_1939.jpg

B32: Hoggett, R. (septiembre 18, 2014). 1954 – ElectroMechanical Manipulator – Ray Goertz (American). [Fotografía]. Recuperado de: http://cyberneticzoo.com/tag/raymond-c-goertz/

B33: NASA Jet Propulsion Laboratory. (s.f). MISSION TO MARS Viking 2. [Fotografía]. Recuperado de: https://www.jpl.nasa.gov/missions/viking-2/

B34: Morio. (mayo 23, 2010). Honda prototype robots Honda Collection Hall.jpg. [Fotografía]. Recuperado de:

https://commons.wikimedia.org/wiki/File:Honda_prototype_robots_Honda_Collection_Hall.jpg

B35: Bell, J. (julio 29, 2014). Next generation: as Honda's Asimo robot grows up, we quiz engineer Satoshi Shigemi about its future. [Fotografía]. Recuperado de: https://www.wallpaper.com/lifestyle/next-generation-as-hondas-asimo-robot-grows-up-we-quiz-engineer-satoshi-shigemi-about-its-future

B36: target brands. (s.f). Anki Cozmo Robot. [Fotografía]. Recuperado de: https://intl.target.com/p/anki-cozmo-robot/-/A-52631303

B37: CMO Innovation editors. (febrero 26, 2017). SmarTone deploying NAO "smart robots" at its retail stores. [Fotografía]. Recuperado de: https://www.cw.com.hk/it-hk/smartone-deploying-nao-smart-robots-at-its-retail-stores

B38: Tinchorton. (s.f). LabView Con Arduino. [Dibujo]. Recuperado de: http://saber.patagoniatec.com/labview-con-arduino-arduino-argentina-ptec/

B39: National Instruments Corporation. (marzo 18, 2010). Diseñar Interfaces de Usuario Personalizadas en NI LabVIEW con su NI USB-TC01. [Captura]. Recuperado de: http://www.ni.com/tutorial/10728/es/

B40: Trossen Robotics. (s.f). Dynamixel MX-28T Robot Actuator. [Fotografía]. Recuperado de: http://www.trossenrobotics.com/dynamixel-mx-28-robot-actuator.aspx

B41: Horsey, J. (2015). Raspbian Jessie Lite Operating System Now Available. [Dibujo] Recuperado de Geeky Gadgets Sitio web: https://www.geeky-gadgets.com/raspbian-jessie-lite-operating-system-now-available-03-12-2015/

B42: BILIB. (2017). QT Creator. [Dibujo]. Recuperado de: https://www.bilib.es/recursos/catalogo-de-aplicaciones/ficha-de-aplicacion/app/qt-creator/

B43: ROBOTS. (s.f). Coding&Software. [Dibujo]. Recuperado de: http://en.robotis.com/model/page.php?co_id=codingsoftware#

C1: Maestre, J. (2015). X-10. En Domótica para ingenieros (p. 8). España: Ediciones Paraninfo.

C2: Maestre, J. (2015). X-10. En Domótica para ingenieros (p. 8). España: Ediciones Paraninfo.

C3: ROBOTIS. (2010). AX-18F/ AX-18A. octubre, 2017, de ROBOTIS Sitio web: http://support.robotis.com/en/product/actuator/dynamixel/ax_series/ax-18f.htm

C4: ROBOTIS. (2010). MX-64T / MX-64R / MX-64AT / MX-64AR. octubre, 2017, de ROBOTIS Sitio web: http://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-64at_ar.htm

C5: ROBOTIS. (2010). MX-28T / MX-28R / MX-28AT / MX-28AR. octubre, 2017, de ROBOTIS Sitio web: http://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-28at_ar.htm

9) Bibliografía.

A. (septiembre, 2014). Ejemplo de Inferencia lógica. Revista Ejemplode.com. De http://www.ejemplode.com/29-logica/3945-ejemplo_de_inferencia_logica.html

Aliverobots. (s.f). NAO Los robots del futuro son ya una realidad. noviembre, 2017, de ROBOTRONICA Sitio web: http://aliverobots.com/nao/

anki. (s.f). Cozmo. Big brain. Bigger personality. noviembre, 2017, de ANKI Sitio web: https://www.anki.com/en-us/cozmo

Arduino. (s.f). Intel Galileo. octubre, 2017, de Arduino Sitio web: https://www.arduino.cc/en/ArduinoCertified/IntelGalileo

Arduino. (s.f). ARDUINO PRO MINI. octubre, 2017, de Arduino Sitio web: https://store.arduino.cc/sa/arduino-pro-mini

Asus. (s.f). Tinker Board. noviembre, 2017, de ASUSTeK Computer Inc. Sitio web: https://www.asus.com/mx/Single-Board-Computer/Tinker-Board/

Benchimol,D. (2011). Sistemas Embebidos. En Proyectos con microcontroladores aprenda a desarrollar sus propias aplicaciones (p.76). Argentina: USERSHOP.

Blog Historia de la Informática. (diciembre 18, 2013). RASPBERRY PI. octubre, 2017, de museo informática Sitio web: http://histinf.blogs.upv.es/2013/12/18/raspberry-pi/

Chacón, F. (2008). Tensión o diferencia de potencial. Patrones. En Medidas Eléctricas para Ingenieros (p. 52). España: Universidad Pontificia Comillas.

chuidiang. (s.f). Conceptos básicos de make y los Makefile. octubre, 2017, de chuidiang Sitio web: http://www.chuidiang.org/clinux/herramientas/makefile.php

Fernández, R., & Ontiveros, M. (2008). Los padres fundadores. En Historias de la historia del cómputo en México (p.15). México, D.F.: Valor Agregado.

Filmaffinity. (2008). WALL•E. octubre 2017, de Filmaffinity Sitio web: https://www.filmaffinity.com/mx/film744679.html

Gallegos, G. (noviembre 24, 2015). End-to-End Digital Integration: The Next Manufacturing Revolution. octubre, 2017, de Advantage Business Media Sitio web: https://www.mbtmag.com/article/2015/11/end-end-digital-integration-next-manufacturing-revolution

García, J. (noviembre 3, 2014). ¿Qué es una placa SBC?. octubre, 2017, de AB Internet Networks Sitio web: https://www.hwlibre.com/que-es-una-placa-sbc/

Grob, B. (1989). Electrónica Básica. Estado de México: McGRAW-HILL.

Gumstix. (s.f). Cobalt MC. noviembre 2017, de Gumstix, Inc. Sitio web: https://store.gumstix.com/computers/cobalt-mc.html

Intel Corporation. (octubre 12, 2017). Introducción a las Placas Intel® Galileo. noviembre, 2017, de Intel Corporation Sitio web: https://www.intel.la/content/www/xl/es/support/articles/000005912/boards-and-kits/intel-galileo-boards.html

Intel Corporation. (s.f). Board Intel® Galileo. octubre, 2017, de Intel Corporation Sitio web: https://ark.intel.com/es/products/78919/Intel-Galileo-Board

Inter-American Institute for Cooperation on Agriculture. (1985). La organización funcional de la computadora. En Compendio de agronomía tropical, Volumen 1 (p. 166). Costa Rica: Bib. Orton IICA / CATIE.

Iñigo, R.,&Vidal, E. (2002). Introducción. En Robots Industriales Manipuladores (169). Univ. Politèc. de Catalunya: ilustrada.

Jones, C. (2013). Starting the ascent of digital computers and software. En The Technical and Social History of Software Engineering (p. 88). Estados Unidos de América: Addison-Wesley.

Junestrand, S.,Passaret, X., & Vázquez, V. (2004). Introducción. En Domótica y hogar digital (p. 4). España: Editorial Paraninfo.

Krol, J. (2016). El robot Cozmo de Anki tiene inteligencia artificial — y emociones. octubre, 2017, de CBS INTERACTIVE INC Sitio web: https://www.cnet.com/es/analisis/anki-cozmo/primer-vistazo/

Maestre, J. (2015). La Vivienda. En Domótica para ingenieros (318). Madrid, España: Ediciones Paraninfo, S.A.

Morris, R. (junio 10, 2015). What is a comparison between Arduino, Intel Galileo and the Raspberry Pi?. octubre, 2017, de Quora Sitio web: https://www.quora.com/What-is-a-comparison-between-Arduino-Intel-Galileo-and-the-Raspberry-Pi

National Instruments Corporation.. (s.f). NI-9237 Módulo de Entrada de Puente/Tensión de la Serie C. octubre, 2017, de National Instruments Corporation. Sitio web: http://www.ni.com/es-mx/support/model.ni-9237.html

National KNX México. (s.f). ¿Qué es KNX?. octubre, 2017, de KNX Association Sitio web: https://www.knx.org/mx/knx/associacion/que-es-knx/index.php

Nottoli, H. (2004). Parámetros de la Electricidad. En Física aplicada a la arquitectura (pp. 136-137). Argentina: Nobuko.

Nuncio, R. (2016). Los Pioneros. En La Magia del Software Historia, Fundamentos y Perspectiva (pp. 21-29). México: CreateSpace Independent Publishing Platform.

Nuncio, R. (2016). Las Matemáticas y la lógica en el software. En La Magia del Software Historia, Fundamentos y Perspectiva (pp. 31-37). México: CreateSpace Independent Publishing Platform.

Nuncio, R. (2016). Breve historia de la computadora. En La Magia del Software Historia, Fundamentos y Perspectiva (pp. 42-51). México: CreateSpace Independent Publishing Platform.

Nuncio, R. (2016). Los primeros pasos del software. En La Magia del Software Historia, Fundamentos y Perspectiva (pp. 53-59). México: CreateSpace Independent Publishing Platform.

Qt Wiki. (junio 3, 2016). Qt Creator/es. octubre, 2017, de Qt Wiki Sitio web: https://wiki.qt.io/Qt_Creator/es

Quiroga, P. (2010). ARQUITECTURA DE COMPUTADORAS. Buenos Aires: Alfaomega Grupo Editor Argentino.

Raspberry Pi Foundation. (s.f). Schematics. octubre, 2017, de Raspberry Pi Foundation Sitio web: https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/README.md

Richarte, J. (2018). Motherboard: partes y funcionamiento. En Servicio Técnico 03: Motherboard: partes y funcionamiento: Curso visual y práctico: PCS • NOTEBOOKS • REDES • MOBILE • Y MÁS (p. 2). Argentina: RedUsers.

roboticspot. (s.f). Historia de la Robotica. noviembre, 2017, de RoboticSpot Sitio web: http://www.roboticspot.com/especial/historia/his2004b.php

ROBOTIS. (2010). MX-28T / MX-28R / MX-28AT / MX-28AR. octubre, 2017, de ROBOTIS Sitio web: http://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-28at_ar.htm

ROBOTIS. (2010). MX-64T / MX-64R / MX-64AT / MX-64AR. octubre, 2017, de ROBOTIS Sitio web: http://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-28at_ar.htm

ROBOTIS. (2010). AX-18F/ AX-18A. octubre, 2017, de ROBOTIS Sitio web: http://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-28at_ar.htm

ROBOTIS. (2010). DYNAMIXEL. octubre, 2017, de ROBOTIS Sitio web: http://www.robotis.us/dynamixel/

ROBOTIS. (2010). USB2Dynamixel. octubre, 2017, de ROBOTIS Sitio web: http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.htm

Santos, M. (abril 12, 2017). Qué es la convergencia digital. octubre, 2017, de Malavida Sitio web: http://www.malavida.com/es/analisis/que-es-la-convergencia-digital-006414#gref

SoftBank Robotics Europe. (s.f). NAOqi Developer guide. noviembre, 2017, de SoftBank Robotics Europe Sitio web: http://doc.aldebaran.com/2-1/index_dev_guide.html

Tubella, I. & Vilaseca, J. (2005). La base tecnológica de la sociedad del conocimiento. En Sociedad del conocimiento (p. 3). Barcelona: Editorial UOC.

Udacity. (s.f). Become A Robotics Software Engineer. noviembre, 2017, de Udacity, Inc. Sitio web: https://www.udacity.com/course/robotics-software-engineer--nd209?utm_medium=email&utm_campaign=acq_123_2018-01-09_act_new-year-new-skills-mlnd-preview&utm_source=blueshift&utm_content=acq_123_2018-01-09_act_new-year-new-skills-mlnd-preview_a&bsft_eid=290ce23b-d3a3-41ef-a3b7-66ce4a6bea54&bsft_clkid=66913c52-5e0e-4f25-a105-35ca1252029c&bsft_uid=e4589567-6cf9-46bb-b27f-ee4340bde37d&bsft_mid=5c9b7da4-191c-4fbb-9bb8-bece21262587#

Universidad Politécnica de Valencia. (2013, Diciembre 18). RASPBERRY PI. Obtenido Enero 04, 2018, de http://histinf.blogs.upv.es/2013/12/18/raspberry-pi/

Vazquez, P., Gomez, J., Prat, A., & Molinero, X. (2006). Clasificación de los lenguajes de programación. En Programación en C++ para ingenieros (p. 47). España: Editorial Paraninfo.

Zabala, G. (2007). Actuadores. En Robótica (p. 39). Argentina: USERSHOP.