UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PROTOTIPO DE UN ASCENSOR Y SU CONTROL COMPARANDO TECNOLOGÍAS DIFERENTES DE ACCIONAMIENTO

MEMORIA PRESENTADA POR:

Kevin Iván Barrera Llanga

Máster Universitario en Ingeniería Mecatrónica

DIRECTOR:

Rubén Puche Panadero

COODIRECTOR:

Ángel Sapena Bañó

Febrero de 2019

Resumen

El presente documento contiene el Trabajo Fin de Máster Universitario en Ingeniería Mecatrónica de la Universitat Politècnica de València de Kevin Iván Barrera Llanga dirigido por Rubén Puche Panadero y codirigido por Ángel Sapena Bañó. El proyecto se llevó a cabo desde el primer curso del master en el laboratorio de máquinas eléctricas de la Universitat Politècnica de València por lo que, el proyecto es destinado a dicho establecimiento, para ensayos, proyectos y análisis de motores, utilizando preferentemente software de libre distribución para futuras mejoras a código abierto.

El proyecto consiste en poder comparar diferentes tecnologías de accionamientos y diferentes tipos de control de las mismas. Para ello, se diseñan dos unidades lineales en paralelo, a modo de ascensor para espacios pequeños, sin embargo, esto podría aplicarse a cualquier otro tipo de unidad desplazada por un motor.

El objetivo de este proyecto es comparar los diferentes tipos de control que puedan reemplazar a un autómata programable (PLC), como pueden ser un arduino o una Raspberry, los cuales resultan más económicos y pueden aportar soluciones de valor como por ejemplo el control a distancia basado en bluetooh y el acceso a las nuevas tendencias, pero por contra presentan poca robustez.

Además, se pretende comparar el empleo de distintos tipos de motores, como son el motor asíncrono de inducción o el motor síncrono de imanes permanentes. Es por ello, que existen dos unidades lineales en paralelo donde ubicaremos cada uno de estos motores para que desde el control se pueda actuar de forma simultánea para su comparación cuando se requiera.

Índice

Tabla de figuras

1.Objetivos

El objetivo del presente Trabajo Fin de Máster es la realización de un prototipo didáctico compuesto de dos ascensores-montacargas, accionados por una tecnología diferente de accionamiento y montados en paralelo para poder comparar entre ellos.

Las dos tecnologías diferentes de accionamientos son: un motor de corriente alterna asíncrono de jaula de ardilla, conocido como el motor de inducción, el cual estará controlado por un convertidor de frecuencia o variador de velocidad con un control escalar y el otro, un motor de corriente alterna síncrono de imanes permanentes, el cual está controlado por un convertidor de frecuencia.

Para controlar el prototipo, el sistema de control se encargará de dar la misma orden a ambos ascensores a la vez y leerá los sensores disponibles en dichas estructuras y accionamientos. Además, se va a poder controlar desde tres tecnologías de control diferentes:

- Un autómata programable
- Raspberry
- Arduino

1.1 Objetivos específicos

A. Realizar el diseño mecánico de los prototipos de dos unidades lineales de transporte.
B. Realizar el diseño eléctrico de alimentación de los accionamientos y de la sensorización del prototipo. Además del diseño eléctrico de los diferentes controles (PLC, Arduino, Raspberry) así como el acondicionamiento de la señal desde el prototipo al control.
C. Realizar el control del sistema mecatrónico basado en dos ascensores en paralelo utilizando para ello tres tecnologías diferentes, (PLC, Arduino, Raspberry) las cuales efectuarán el control del ascensor-montacargas de forma automática controlando los dos a la vez o manual uno a uno.
D. Visualizar y operar el prototipo de forma local y remota a través de un sistema Scada, basado en una aplicación funcional en Android la cual envíe y reciba señales a un dispositivo móvil. Desde el cual se pueda controlar y monitorizar el prototipo con una comunicación inalámbrica, basada en bluetooth.

E. Diseñar un sistema embebido y de bajo coste el cual sea capaz de monitorizar las características técnicas de los motores en su régimen de funcionamiento, como puede ser velocidad, par, tensión, corriente y/o potencia consumida. Este sistema se conectará vía inalámbrica en tiempo real a una aplicación móvil.
F. Comparar el resultado del uso entre los sistemas y determinar la valoración existente en cada uno.

2. Introducción

En el presente proyecto se pretende diseñar un puesto didáctico de dos unidades lineales tipo ascensores-montacargas, cada una de ellas estará accionada por un motor, inducción y de imanes permanentes.

De tal forma que sirvan para comparar las características de los motores entre sí, como velocidad de respuesta, precisión, configuraciones, tiempo de respuesta, conexiones, ruidos de comportamiento dinámico. También se quiere comparar la forma de controlarlos, desde elementos robustos industriales tales como los autómatas programables a elementos de bajo coste tipo arduino, Raspberry o cualquier tipo de tarjetas de bajo costo con hardware y software abierto. Comparando los tres sistemas y los dos motores en funcionamiento de forma local y remota.

Para finalizar se incorpora un osciloscopio, el cual es hecho específicamente para el monitoreo de las unidades lineales, en donde se puede visualizar las gráficas de tensión, corriente y los valores nominales de velocidad, tensión, corriente y par.

3. Justificación

La realización del presente trabajo fin de master es indispensable para la obtención del Master en Ingeniería Mecatrónica, de acuerdo con la normativa vigente del Ministerio de Educación y Ciencia. El Trabajo ha sido realizado en la Escuela Técnica Superior de Ingeniería del Diseño en la Universitat Politècnica de València. En el laboratorio de control de máquinas y accionamientos eléctricos del Departamento de Ingeniería Eléctrica.

El trabajo cumple con las especificaciones de la ideología mecatrónica impartida a lo largo de todo este máster, es decir que el mismo utiliza conocimientos de mecánica, eléctrica y control industrial aplicado en un sistema el cual abre las puertas a la innovación en la automatización ,es decir el proyecto contribuye a que arquitecturas de bajo costo basadas en lenguaje C puedan utilizarse para controlar ascensores-montacargas domésticos e industriales, llevando a cabo una comparación de accionamientos y de controles para su implementación.

Hoy en día es muy importante tener un alto conocimiento en el tema de automatización industrial, ya que cada día son más exigentes las demandas de este tipo de sistemas. Los montacargas o los ascensores son máquinas muy utilizadas, un método adecuado de mejorar estos sistemas es compara alternativas de accionamientos y de controles, con esto se genera un sistema el cual se adapte a las necesidades de cada industria.

Además, también adquiriremos otros conocimientos, que serán necesarios para la realización de este Trabajo Fin de Master. Estos conocimientos son los siguientes:

- Conocimientos sobre conexionado de los motores a ensayar.
- Conocimientos sobre maquinas eléctricas, electrónica de potencia y controladores industriales
- Conocimientos de electrónica, trabajo con encoder, fuentes de alimentación y sensores.
- Conocimientos sobre motores, posibles problemas que puedan surgir.
- Conocimientos sobre como coordinar motores y un PLC.
- Conocimiento de lenguaje C, C++
- Conocimiento de comunicaciones inalámbricas.

4. Estudio Previo

4.1 Ascensores

El ascensor es un sistema de transporte vertical, diseñado para mover personas u objetos entre los diferentes niveles de un edificio o estructura. Está formado por partes mecánicas, eléctricas y electrónicas que funcionan en conjunto para ponerlo en marcha. El ascensor surge como complemento perfecto para la forma moderna de organización de las ciudades y de nuestra vida.

Un ascensor montacargas es un tipo de elevador con capacidad para transportar tanto carga como personas salvando las distintas plantas de altura de una vivienda, edificio, local comercial o empresa. Este tipo de elevador recibe también nombres como Ascensor Mixto o Montacargas para personas. Es importante no confundir este tipo de elevadores con los montacargas, los cuales solamente son aptos para transportar objetos o cargas y que por tanto tienen que cumplir con una normativa diferente que en el caso de los ascensores para personas y objetos.

4.1.1 Evolución

El primer elevador fue construido por el arquitecto Arquímedes y funcionaba con cuerdas y poleas, estos seguían el sistema de tracción sobre la base del actual mecanismo de la grúa. El sistema más evolucionario desde la antigüedad fue el basado en la transmisión a tornillo, surgiendo la necesidad debido al aumento del precio del suelo, lo que llevó a hacer parcelas con más plantas utilizando los mismos metros cuadrados, y el descenso del precio del acero fueron las causas que movieron a los hombres a pensar en algún aparato que permitiera suplir las necesidades de subir y bajar edificios, sin utilizar las escaleras.

Elisha Otis, en 1852, inventó el primero freno de seguridad para ascensores. Hasta el momento montar en uno de los elevadores de vapor, era toda una odisea, además de que con demasiada frecuencia se desplomaban. En 1857 se instaló el primer ascensor con este freno, en un edificio de cinco plantas. Este nuevo tipo de ascensor retiró el de vapor. Y hasta el 1904 los elevadores hidráulicos fueron el sistema dominante en los edificios, aunque ya desde 1880 se empezaron a instalar los primeros ascensores eléctricos de engranajes. Al principio, muy lentos y solo aptos para edificios con poca altura. Pero en 1904, se instalaron las primeras máquinas sin engranajes y estos desbancaron los hidráulicos.

Estos nuevos ascensores, rápidos y con límites de altura elevadísimos, causaron la revolución de los rascacielos. En la actualidad, el edificio más alto del mundo, la Torre Burj Khalifa en Dubái, con 828 m de altura, tiene ascensores de la compañía Otis Elevator Company que suben la distancia más larga del mundo: 504 metros; también tiene el acceso de ascensor situado a mayor altura del mundo: a 638 metros; y el ascensor con doble cabina más rápido del mundo: 10 metros por segundo.

4.1.2 Tipos de ascensores

Cuando hablamos de tipos de ascensores, aunque son diversos los criterios a los que podemos atender, una de las clasificaciones más universales es la que los divide en dos tipos de ascensores básicos según su mecanismo de accionamiento: los ascensores eléctricos y los hidráulicos.

Ascensores eléctricos

Los ascensores eléctricos son aquellos cuya tracción se realiza mediante un motor eléctrico. En su conjunto, este tipo de ascensores está formado por un motor, una polea, una máquina tractora y un cable de tracción. En lo que se refiere a la cabina, esta está suspendida en uno de los extremos del cable, mientras que el otro se encuentra un contrapeso. También pueden ser con engranajes (el método más clásico, pero con peor rendimiento) o sin engranaje o gearless, mucho más silenciosos. En cualquier caso, todos ellos pueden ser de tiro directo o con suspensión de poleas.

Ascensores hidráulicos

La principal diferencia de este tipo de ascensores es que funcionan con un bloque de válvulas y una bomba hidráulica que, en algún caso, puede estar gestionada de manera electrónica. En este tipo de ascensores el accionamiento se logra a través de un bloque de válvulas y una bomba que puede estar acoplada al motor eléctrico. Utilizan como elemento impulsor un aceite (por eso también se les conoce como ascensores oleodinámicos) que se inyecta a través de las válvulas desde un depósito hasta el pistón. La cabina de este tipo de ascensor sostiene en el émbolo del pistón y una vez se llena de aceite, sube y empuja la cabina hacia arriba. Así, el motor realmente solo está en funcionamiento durante la subida, mientras que la bajada se produce abriendo paso al aceite y dejándolo marchar, de modo que el efecto de la propia gravedad es el que hace

descender al ascensor. Este tipo de ascensor se desplaza más despacio que los eléctricos, pero no es un inconveniente en edificios de pocas alturas.

4.2 Motores Eléctricos

Un motor eléctrico es una máquina eléctrica que transforma energía eléctrica en energía mecánica mediante interacciones electromagnéticas. Algunos motores eléctricos son reversibles, pueden transformar energía mecánica en eléctrica funcionando como generadores. Pueden funcionar conectados a una red de suministro eléctrico o a baterías.

Los motores de corriente continua y los de corriente alterna se basan en el mismo principio de funcionamiento, el cual establece que, si un conductor por el que circula una corriente eléctrica se encuentra dentro de la acción de un campo magnético, éste tiende a desplazarse perpendicularmente a las líneas de acción del campo magnético.

4.2.1 Tipos de motores

Motores de corriente continua o directa

Los Motores de Corriente Directa (CD) o Corriente Continua (CC) es una máquina que convierte energía eléctrica en mecánica, provocando un movimiento rotatorio, gracias a la acción de un campo magnético. Se utilizan en casos en los que es importante el poder regular continuamente la velocidad del motor, además, se utilizan en aquellos casos en los que es imprescindible utilizar corriente directa, como es el caso de motores accionados por pilas o baterías. Este tipo de motores debe de tener en el rotor y el estator el mismo número de polos y el mismo número de carbones.

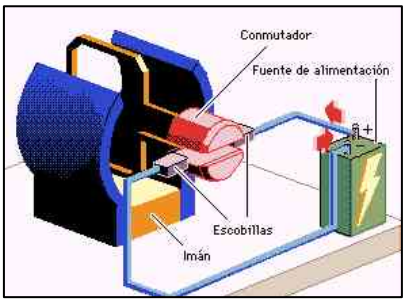


Figura 1 Esquema de un motor de corriente continua

Motor de inducción

Un motor asíncrono o motor de inducción es un motor eléctrico de corriente alterna monofásico o trifásico que cumple la función de girar a una velocidad diferente del campo magnético del estator, esto se produce cuando la corriente del rotor necesaria para producirse el par es inducida por el campo magnético de la bobina del estátor. El rotor

puede ser jaula de ardilla o bobinado mientras que su estator está conformado por bobinas desfasadas entre si a 120 grados.

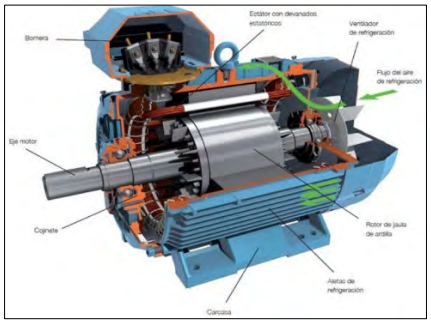


Figura 2 Esquema de un motor asincrónico a inducción

Motor de imanes permanentes

Los motores de imanes permanentes son motores eléctricos que pueden ser de corriente continua o alterna, utilizan la combinación de campos magnéticos los cuales son de naturaleza permanentes más conocidos como imanes y campos magnéticos inducidos por una corriente de excitación externa la cual fluye a través de los devanados del estator. Esta excitación puede hacerlo una señal eléctrica continua o alterna, pero cabe recalcar que los que utilizan una excitación alterna son los más empleados en las industrias ya que son más eficientes y poseen una gran variedad de par.

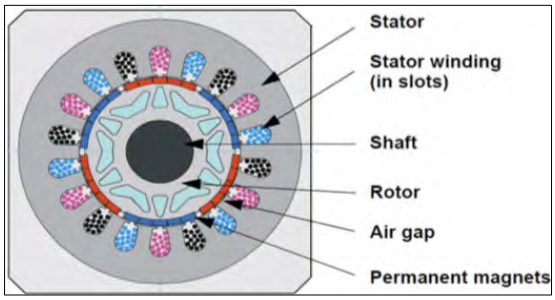


Figura 3 Esquema de un motor Sincrónico de Imanes Permanentes

Motor de reductancia variable

Es un tipo de motor eléctrico paso a paso, cuyo funcionamiento se basa en la reluctancia variable mediante un rotor dentado en hierro dulce que tiende a alinearse con los polos bobinados del estator. Se pueden diseñar para funcionar con pasos más pequeños que los habituales. Esto quiere decir, con pasos más cortos que los de un motor de imán permanente. Por otro lado, el rotor en este tipo de motores es de baja inercia. Ello implica la mejora de la respuesta dinámica, aunque su control sea en bucle abierto.

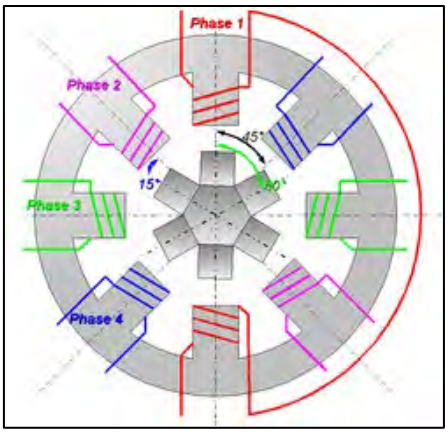


Figura 4 Esquema de un motor de reductancia variable

4.2.2 Control de velocidad

Control por doble devanado

Consiste en un motor con dos devanados independientes en el estátor, con diferente número de pares de polos, generalmente conectados en estrella. La conexión en estrella es interior. Cada bobinado lleva su correspondiente protección contra sobreintensidad por medio de relé térmico reglado a los valores de protección del bobinado. A partir del control se generan dos velocidades con devanados independientes. (Moeller, 2015)

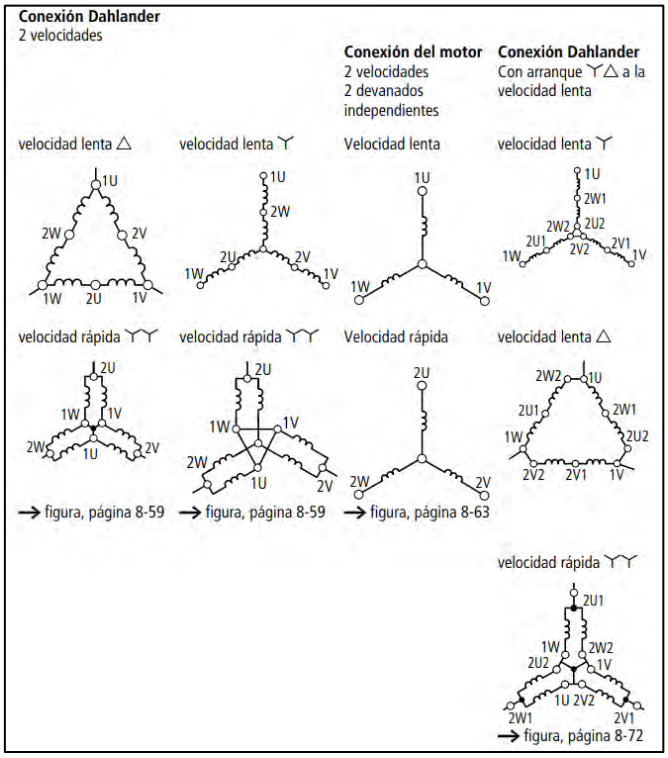


Figura 5 Tipos de conexiones con doble devanado

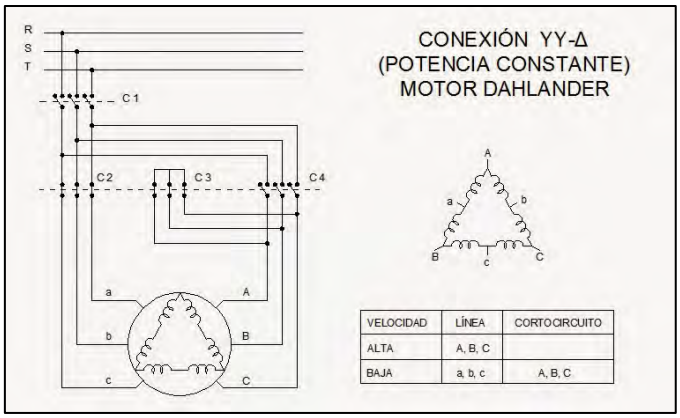


Figura 6 Esquema de conexión control por doble devanado

Control por variación de la resistencia del rotor

La inserción de una resistencia rotórica suplementaria produce un incremento en el deslizamiento del rotor gracias a esto existe una variación de la velocidad sobre una amplia gama por debajo de la velocidad síncrona del motor produciendo una simplicidad de funcionamiento, tanto desde el punto de vista manual como automático. Los costos iniciales y de mantenimiento son bajos para los reguladores manuales y automáticos, aunque su principal carencia es el bajo rendimiento, debido al aumento de las pérdidas de la resistencia del rotor. El motor de inducción de rotor bobinado se emplea mucho con control de la resistencia secundaria para cargas de naturaleza intermitente, requiriendo par de arranque elevado y aceleración-desaceleración relativamente rápidas tales como grúas de fundiciones, elevadores y siderúrgicas. Ya que la velocidad y el deslizamiento de un motor de inducción del rotor bobinado son proporcionales a la resistencia del rotor, el método de control de velocidad mediante variación de la resistencia del rotor se denomina también como control del deslizamiento.

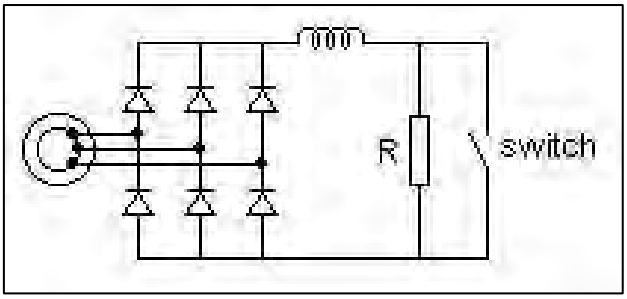


Figura 7 Control por variación de la resistencia del rotor.

Variador de frecuencia

Un variador de frecuencia es un sistema de control de la velocidad rotacional de un motor de corriente alterna, el sistema consiste en controlar la frecuencia de alimentación que es suministrada al motor, esta a su vez es una señal en forma de pulsos debido a la conmutación existente el circuito de potencia, utiliza comandos previamente

programados en los cuales existen una gran variedad de parámetros de manejo del motor, además posee entradas analógicas y digitales para controlar a través de pulsadores o de un autómata programable.

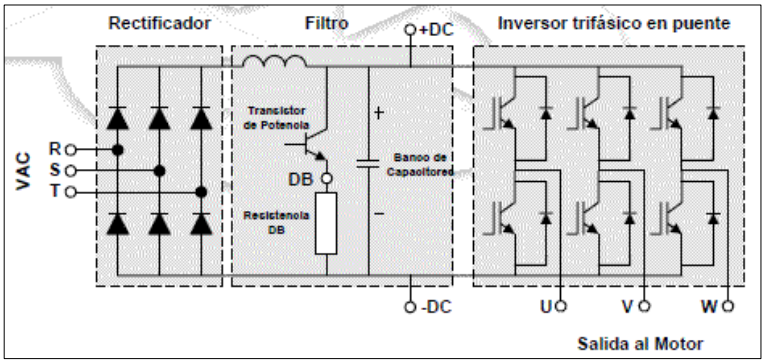


Figura 8 Circuito base de un variador de frecuencia

Las características principales del uso de variadores de frecuencia para ascensores y montacargas son:

- Acciona motores síncronos y asíncronos de forma silenciosa
- Proporciona un control preciso en bucle abierto para motores asíncronos sin encoder
- Safe Torque Off (SIL3) elimina contactores en el lado del motor (certificado por TÜV)
- Ofrece una amplia gama de configuraciones de aceleración, deceleración y amortiguación de sacudidas para un máximo confort.
- En caso de apagón, el modo de evacuación inteligente, combinado con una fuente de alimentación de respaldo (como una batería de CC o un SAI), lleva el ascensor hasta la planta más próxima para permitir que salgan sus ocupantes

Variador de frecuencia privados

El variador Otis OVF1, también conocido como el Otis OVF30 fue originalmente creado a principios de los años 90, este variador diseñado a medida, y al que se refieren también como el Otis DBSS cuando es utilizado como componente de un sistema de ascensor Otis, puede también ser conocido como el Otis OVF1, o como el Otis OVF30 dependiendo de la revisión del diseño. El sistema del variador OVF1 integra un controlador de vector de flujo de bucle cerrado altamente optimizado que incluye un circuito cerrado de seguridad para operar motores de ascensor sin engranajes. Cuando se

combina con el controlador Otis MCS321 el variador puede operar ascensores con cargas de hasta 2500 kg a velocidades hasta de 2.5 m/s. (ElevatorDriveS.A., 2017)

Figura 9 Variador de frecuencia Otis OVF1

La empresa italiana Daldoss fabrica elevadores para pasajeros y carga, incluidos elevadores para sillas de ruedas y plataformas, las tarjetas implementadas para la variación de la frecuencia en sus ascensores, ayuda a que se creen nuevos modelos entre los cuales destacan:

- Microlift: Dumbwaiter con una capacidad máxima de 300 kilogramos.
- Microlift Evolution: el instalador de pesas con una capacidad máxima de solo 50 kilogramos.
- Microfreight: Tractor de carga. Tiene dos versiones; Microfreight Classic (con una capacidad máxima de 1500 kilogramos) y Microfreight Plus (con una capacidad máxima de 1400 kilogramos).
- EasyGo: sala de máquinas menos ascensor de pasajeros. Tiene dos versiones; EasyGo-e (tracción sin engranajes) y EasyGo-h (hidráulico). (montacarichi, 2018)

Figura 10 Variador de frecuencia Daldoss

4.3 Control del Ascensor

A finales de la década de 1960 comenzó a surgir una revolución en la cual las industrias necesitaban adaptarse a las altas demandas de los usuarios, es aquí donde nuevas tecnologías electrónicas son vistas como una solución más eficiente a sistemas de control, basados en relés. Es en el año de 1968 donde se emite una solicitud de propuesta para remplazar los sistemas de control dando como resultado el primer PLC de la historia. El PLC (control lógico programable) es una computadora con múltiples señales de entrada y de salida resistente a vibraciones, ruido eléctrico, rangos de temperatura e impactos que automatiza procesos electromecánicos utilizados usualmente en las industrias. Su propósito inicial no era otro que el de brindar solución a una gran cantidad de inconvenientes que tenían los automatismos clásicos, basados en componentes electromecánicos o desarrollados con electrónica específica mediante tarjetas, lo que hacía que una máquina o un proceso controlado con esta tecnología difícilmente podría ser modificado o mejorado y para lograr el objetivo que se perseguía había la necesidad de reconstruir toda la ingeniería implementada y adaptar la nueva diseñada.

Hoy en día la tecnología crece a pasos gigantescos, el sector industrial no es una excepción, la automatización se convierte en una parte fundamental para la industria, en la actualidad se ofrece al mercado distintas alternativas, ajustándose a la necesidad y al bolsillo de cada empresa. (Tedesco, 2011)

4.3.1 Problemática Encontrada

El autómata programable, es una computadora empleada en las industrias, son considerados como dispositivos de automatización empresarial, desde su primera implementación fueron evolucionando aumentando más funciones que mejoran su uso y que se adapten a un campo mucho más grande de aplicación, pero todo conlleva un coste. La implementación de este tipo de dispositivos implica una alta inversión de implementación y mantenimiento periódico haciendo que las industrias iniciales y medias busquen soluciones más rentables que brinde similares características de automatización, es aquí en donde nace la necesidad de utilizar tarjetas que cumplan con las funciones básicas de un PLC sin perder su adaptabilidad.

Arduino y Raspberry, dos grandes proyectos que simplifican la tarea del usuario al brindar una placa de desarrollo de hardware para construir dispositivos interactivos de control con una baja inversión y con módulos capases de adaptarse a cada necesidad del cliente. Debido a que en una aplicación específica en donde se requieren entradas/salidas digitales

o analógicas como lo es este proyecto, es necesario recurrir a otras alternativas de control que se acoplen y simplifiquen, brindando las mismas características que un dispositivo industrial, inclusive mejorando las experiencias del usuario. Es por tal motivo que se da este proyecto el cual pone a prueba las características de estas 2 tarjetas y compara su comportamiento frente a dos unidades lineales de transporte que se mueven mediante el uso de motores industriales de imanes permanentes y asíncrono. Logrando así dar un paso más en la evolución de la automatización, desarrollando un proyecto de fácil implementación, viable y rentable para el uso industrial

4.3.2 Tarjetas de control

PLC

Un PLC o progammable logic controller es un dispositivo electrónico que se programa para realizar acciones de control automático, es muy utilizado en las industrias y a su vez es el cerebro que activa componentes en la maquinara para que ejecuten tareas las cuales pueden ser peligrosas, repetitivas o de precisión. Con un costo de inversión elevado se obtiene un producto el cual resuelve los requerimientos de control de procesos y secuencias en una maquinaria dentro del área industria, además puede implementarse como una solución domestica uy comercial a tareas poco automatizadas. (Balcells, 2010)

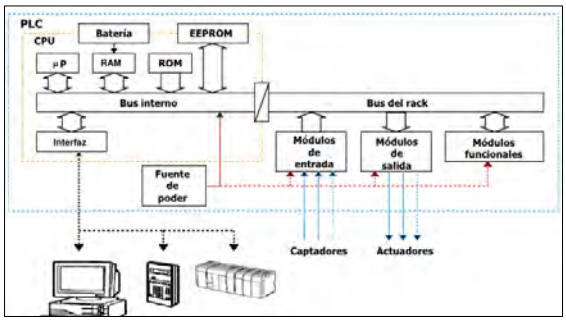


Figura 11 Esquema de diseño interno de un PLC

Arduino

Es un dispositivo electrónico de código abierto basado en hardware y software libre, es utilizada como una tarjeta de control y adquisición de datos. Esta tarjeta permite crear diferentes tipos de controles para usos varios desde domótica hasta implementación industrial. Al ser de hardware abierto permite conectar diferentes tipos de periféricos para que amplíen las funciones de usos como conectarse a internet, módulos de encoder, bluetooth, temperatura y humedad, etc. La principal característica de estas tarjetas es su costo de implementación y uso ya que se adapta a las necesidades y al bolsillo de cada usuario. (ArduinoSA, 2015)

Figura 12 Arduino DUE

Raspberry

Es un ordenador de placa reducida, con puertos GPIO, basa en programación Linux, este ordenador soporta varios componentes necesarios en un ordenador común, es pequeño pero capaz de realizar acciones como procesador de texto, multimedia y sirve también como una tarjeta de control, con el uso de sus puertos GPIO, estos puertos sirven como entradas y salidas digitales, además de poseer una comunicación serial puede conectarse mouse o teclados a este dispositivo, dando así como resultado una placa con su propio sistema operativo, funcional y que puede ser aplicado en un campo muy amplio de desarrollo. (Upton, 2014)

Figura 13 Raspberry Pi 3B

Tarjetas de control privadas

Otis: Es una empresa dedicada al transporte vertical. Otis es uno de los fabricantes más grandes del mundo de ascensores, escaleras mecánicas y andenes móviles. Otis fabrica, realiza la instalación y el mantenimiento de todos sus dispositivos en sistema de ascensores para el edificio más alto del mundo o un simple ascensor para una casa de dos pisos utilizando sus tarjetas de control propias. (Company, 2018)

Figura 14 Tarjeta de control velocidad Otis Lcb2

Daldoss Elevetronic: Es una empresa encargada del diseño construcción y mantenimiento de elevadores a nivel mundial, conocida por su amplio campo de aplicación y adaptabilidad de sistemas de elevación, son la solución ideal para almacenes, industrias, supermercados y negocios. (montacarichi, 2018)

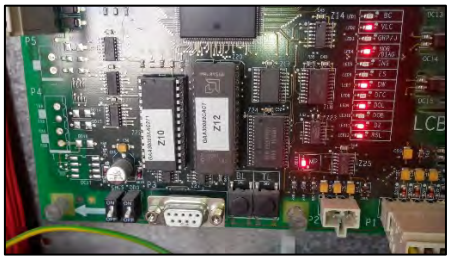


Figura 15 Tarjeta de control velocidad Daldoss

Orona: Es un grupo empresarial formado por 30 empresas en España, Francia, Portugal, Reino Unido, Bélgica, Holanda e Irlanda. La actividad de Orona se centra en el diseño, fabricación, instalación, mantenimiento y modernización de soluciones de movilidad, tales como ascensores, escaleras mecánicas, rampas y pasillos. (Orona, 2017)

Figura 16 Tarjeta de control velocidad Orona

4.4 Sensores de control

4.4.1 Sensores de velocidad

Sensor de medición lineal

El sensor de medición lineal es un sistema de posicionamiento de código de barras de alta precisión y sin contacto. En la caja del ascensor, el sensor no solo determina la

posición absoluta de la cabina del ascensor, sino que además controla la velocidad. Gracias a su tecnología de cámaras, el sensor no sufre desgaste ni necesita mantenimiento con una resolución ajustable de hasta 0,1 mm. (Sensor, 2018)

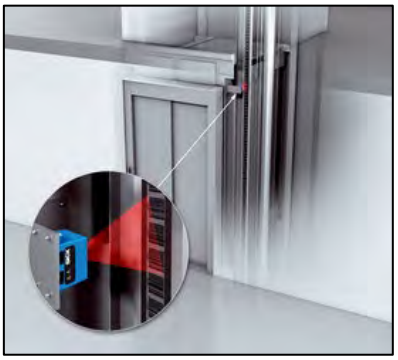


Figura 17 Sensor de medición lineal

Encoder

Es un dispositivo electrónico de detección que genera una respuesta en función a una revolución, es decir convierte el movimiento de un eje en una señal eléctrica, esta va a un sistema de control el cual se encargara mediante algoritmos de transformar esta señal a un valor de posición, número de vueltas, velocidad, dirección. El circuito funciona a través de un haz de luz que pasa a través de un disco codificador, este genera una señal la cual es analizada en la placa del circuito y a su vez enviada desde los pines de salida en el encoder a un dispositivo el cual será el encargado de controlar este valor. (Cho, 2014)

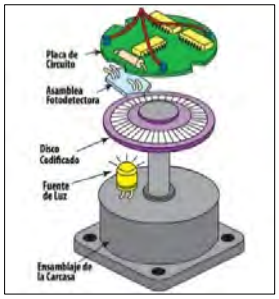


Figura 18 Esquema básico de funcionamiento del encoder

Resolver

Es un tipo de transformador eléctrico rotativo utilizado para medir los grados de rotación. Se considera un dispositivo analógico, funciona al enviar el voltaje inducido de dos espiras, que por razones constructivas dan como resultado dos señales senoidales desfasadas, examinando estas señales podremos conocer la posición, y examinando el desfase la dirección. (Dorf, 2005)

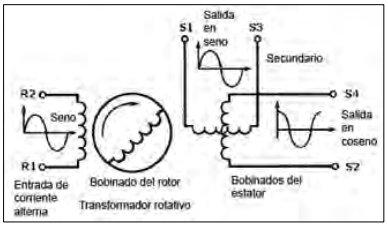


Figura 19 Principio de funcionamiento del resolver

4.4.2 Sensores de posición

Sensor Inductivo

Un sensor de proximidad inductivos puede detectar objetos metálicos que se acercan al sensor, sin tener contacto físico con los mismos. Los sensores de proximidad inductivos se clasifican más o menos en los siguientes tres tipos, de acuerdo con su principio de funcionamiento: el tipo de oscilación de alta frecuencia que utiliza la inducción electromagnética; el tipo magnético que emplea un imán; y el tipo de capacitancia que aprovecha los cambios en la capacidad eléctrica.

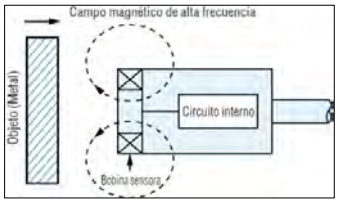


Figura 20 Principio de funcionamiento de un sensor inductivo

4.4.3Sensores de seguridad

Limitador de velocidad

Cuando la cabina supera una determinada velocidad, se bloquea la polea del limitador y con ella el cable, dando un tirón a la palanca del paracaídas, y accionando así el mecanismo que presionará las zapatas sobre las guías y detendrá finalmente la cabina. Existen 2 tipos de poleas del limitador de velocidad: limitador de velocidad oscilante y limitador de velocidad centrífuga. En el primero de ellos es un gatillo oscilante el que se enclava al acelerarse, y en el segundo es la acción de la fuerza centrífuga la causante de la operación de frenada. La única ventaja que tiene uno sobre otro es que el centrífugo es más silencioso aún a velocidades elevadas, motivo por el que se emplea en mayor medida. (Pérez-Formoso, 2010)

Figura 21 Limitador de velocidad

Paracaídas

Los paracaídas de aceleración actúan cuando la cabina adquiere una velocidad superior a la norma. Cuando el cable del limitador se detiene a consecuencia del propio funcionamiento del limitador de velocidad, tira, accionando una timonería que hace desplazar en dirección vertical unas varillas de actuación. (Pérez-Formoso, 2010)

Figura 22 Paracaídas de ascensores

Amortiguadores

Los ascensores deben estar provistos de amortiguadores para detener la cabina o el contrapeso en caso necesario. Se sitúan en el foso al final del recorrido de la cabina o del contrapeso, aunque también pueden montarse en la parte inferior del bastidor de éstos. En este caso, según la Norma EN 81-1, deben golpear en el foso sobre un pedestal. Los amortiguadores pueden ser elásticos (de caucho), de resorte (o muelle) o hidráulicos en lo que a su estructura se refiere. La Norma EN 81-1 distingue 3 clases de amortiguadores atendiendo a otras prestaciones:

- Amortiguadores de acumulación de energía (elástico), que no pueden emplearse más que para ascensores de velocidad nominal no superior a 0.63 m/s.

- Amortiguadores de acumulación de energía con amortiguación del movimiento de retorno (de resorte), para ascensores de velocidad no superior a 1 m/s.
- Amortiguadores de disipación de energía (hidráulico), que pueden ser empleados en ascensores de cualquier velocidad. (Pérez-Formoso, 2010)

Figura 23 Amortiguadores de ascensores

Sensor fin de carrera

Son dispositivos electrónicos, neumáticos o mecánicos situados al final del recorrido o de un elemento móvil, como por ejemplo una cinta transportadora, con el objetivo de enviar señales que puedan modificar el estado de un circuito. Internamente pueden contener interruptores normalmente abiertos (NA), cerrados (NC) o conmutadores dependiendo de la operación que cumplan al ser accionados, de ahí la gran variedad de finales de carrera que existen en mercado. (Areny, 2014)

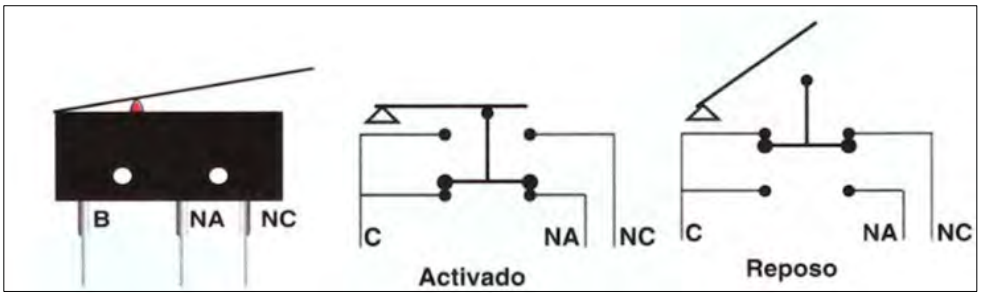


Figura 24 Funcionamiento de un sensor fin de carrera

5. Diseño Mecánico Adoptado

Una parte fundamental es dar forma, dimensiones, materiales, tecnología de fabricación y funcionamiento de todo el ecosistema del proyecto. Teniendo en cuenta las especificaciones técnicas se procede a realizar el análisis de sistema mecánico.

5.1 Análisis de la carga

Partiendo de que los ensayos van a ser realizados en el laboratorio de máquinas eléctricas, se decide diseñar el prototipo ajustándose a la altura en una pared del laboratorio, los ensayos serán realizados para un control de posición con un máximo de 3 puntos de elevación, por tal motivo y tomando las consideraciones de altura en la pared es necesario una unidad lineal con una carrera máxima de 2600 mm. El peso que elevará la unidad lineal tendrá como máximo un valor de 15 Kg, debido a que en el prototipo será compacto y realizarán pruebas sin carga, media carga y carga completa, en la siguiente figura se puede apreciar el diagrama de cuerpo libre, la presentación gráfica utilizada para analizar las fuerzas que actúan sobre un cuerpo libre en este caso el peso del ascensor montacargas en donde se especifican las fuerzas involucradas en el prototipo. Según el principio de acción y reacción, es la misma fuerza que ejerce sobre la estructura, pero en sentido opuesto. Aplicando la segunda ley de Newton sobre el ascensor llegamos a la ecuación.

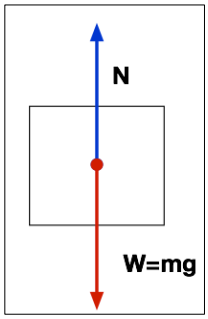


Figura 25 Diagrama de cuerpo libre del prototipo elevador

$$N - W = m \cdot a$$
$$W = m \cdot g$$
$$W = 15 \cdot 9.81$$
$$W = 147.15\ [N]$$

Partiendo de este diagrama de cuerpo libre se determina que la fuerza necesaria en la unidad lineal debe ser menor o igual a 148 N, por lo tanto, se procede a la selección de acuerdo al análisis realizado.

5.2 Selección de la unidad lineal

La unidad lineal representa un elemento de elevación compacto y robusto. Consiste en una guía lineal con un cilindro. Se puede instalar directamente sobre el objeto a elevar. Esto significa que multitud de mesas y dispositivos pueden ser equipados fácilmente con uno de sistemas de elevación, de las cuales resaltan las siguientes.

Tabla 1 Tipos de unidades lineales

Nombre	Fuerza Máxima	Recorrido	Movimiento
ErgoSwiss	120N	2500mm	Correa
Paternoster	140N	2200mm	Correa
Item LRE5	150N	2600mm	Correa
Winkel	160N	2000mm	Cadena
Mechbelt	135N	1800mm	Correa

Por tal motivo se ha seleccionado una unidad lineal compacta con gran rendimiento, el modelo es el LRE 5 D6 60x20 ZU 40 R10, posee unas robustas unidades de rodadura y correa dentada, trabaja en poco espacio disponible y a una gran velocidad. De acuerdo con las especificaciones, la fuerza máxima que soporta es de 150 N y su carrera máxima es de 2600mm por lo cual cumple con las especificaciones de selección de la unidad lineal.

Tabla 2 Especificaciones de rendimiento de la unidad lineal LRE5

Unidad de suministro		**=**	**1 pza.**
Rendimiento	s_{min}	=	46 mm
Carrera máx.	H_{max}	=	2670 mm
Repetibilidad	$F_{x\,max}$	=	150 N
	$F_{y\,max}$	=	400 N
	$F_{z\,max}$	=	320 N
	$M_{x\,max}$	=	4 Nm
	$M_{y\,max}$	=	6 Nm
	$M_{z\,max}$	=	8 Nm
Momento de inercia, eje x	I_x	=	16.09 cm^4
Momento de inercia, eje y	I_y	=	2.06 cm^4
Momento resistente, eje x	W_x	=	5.36 cm^3
Momento resistente, eje y	W_y	=	2.06 cm^3
Peso	m	=	1.296242kg
Peso, espec. Longitud	m	=	1.97 kg/m
Momento de inercia, torsional	I_t	=	1.61 cm^4

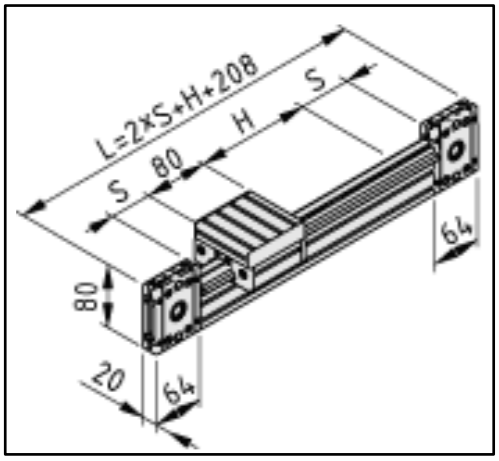


Figura 26 Dimensiones de la unidad lineal LRE 5 (Manual de especificaciones LRE5)

Las dimensiones son óptimas para ser implementadas en el laboratorio, cabe destacar que esta unidad es recomendada para aplicaciones de estudio por su versatilidad y adaptación a diferentes sistemas de control. En el diagrama de cuerpo libre de la *Figura 25* se determinó la fuerza máxima a aplicarse, en el manual de diseño del LRE 5 D6, tenemos el siguiente diagrama de fuerzas y momentos.

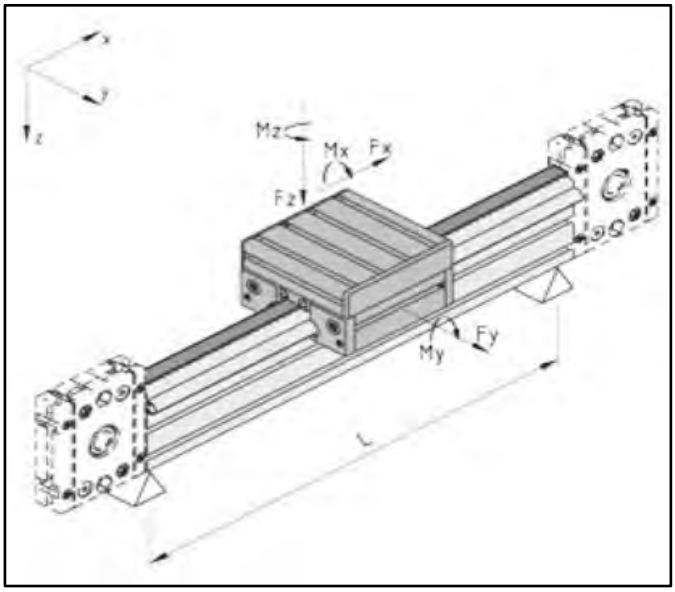


Figura 27 Diagrama de momentos y fuerzas en la unidad lineal (Manual de especificaciones LRE5)

De acuerdo con la *Figura 27*, la fuerza Fx es la fuerza máxima que la unidad lineal puede elevar, partiendo de este principio, es necesario determinar el par de giro a aplicarse en la unidad lineal, con lo cual la *Figura 28* nos indica que para una fuerza de 150 N como máximo el par que resiste la unidad lineal es menor a 3 Nm.

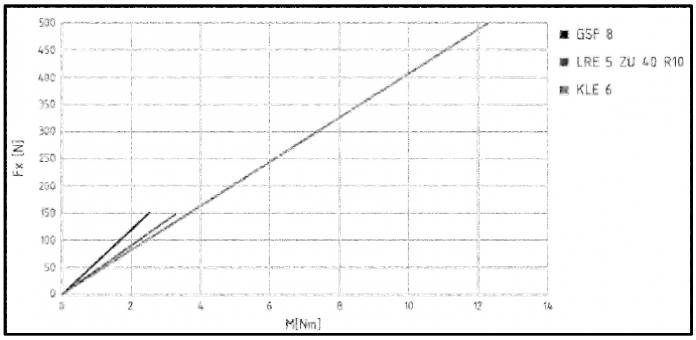


Figura 28 Carga útil FX según el par de giro aplicado (Manual de especificaciones LRE5)

5.3 Modelado de la unidad lineal

A partir de los datos del fabricante se procede a realizar el modelo en el software SolidWorks. Las medidas son tomadas a partir de los planos oficiales de la unidad lineal LRE5, con el fin de asemejarse a un análisis real de funcionamiento. El tamaño de la unidad lineal incluyendo los ejes de giro es de 2.8 m mientras que el recorrido total que tiene la unidad lineal es de 2.68m, la documentación correspondiente a los planos de construcción como vistas y perspectiva se encuentra en el **ANEXO I**

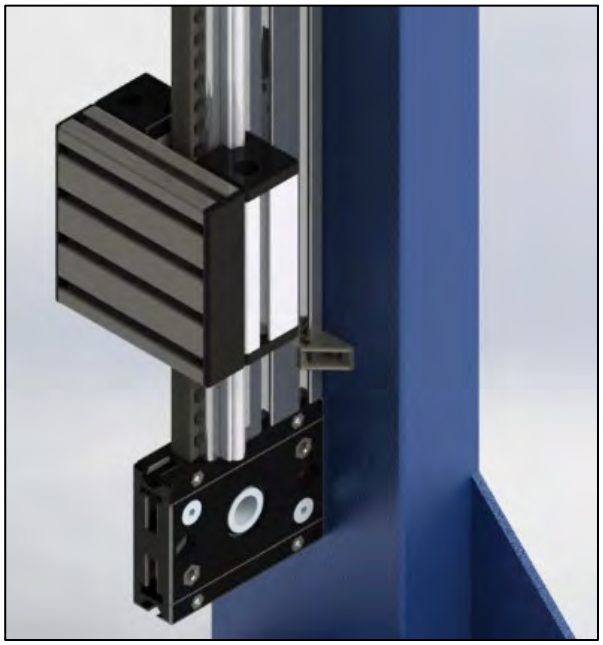


Figura 29 Unidad lineal LRE5 modelado en SolidWorks

Cuando se modela o diseña una pieza mecánica es fundamental realizar un análisis estático de esfuerzos, comprobando si el modelo a implementarse es adecuado a partir de la información de regiones de interese sometidas a tensión.

Para este análisis utilizamos la barra de recorrido ya que en sus extremos estará la correa dentada con una carga máxima de 150N, el empotramiento de esta barra es en su cara posterior, como se ve en la siguiente imagen.

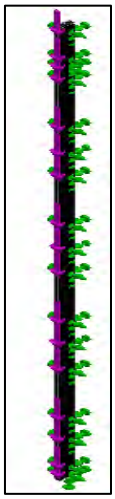


Figura 30 Sujeción y fuerza aplicada en la barra del recorrido

La información referente al análisis resuelto en el software SolidWorks se encuentra en el **ANEXO II.** En el informe se genera el criterio de máxima tensión de von Mises, esta teoría de la energía de cortadura o de la energía de distorsión máxima nos permite saber el límite máximo elástico de ruptura. En la barra de desplazamiento se obtiene que el límite elástico de deformación para la barra de aluminio 1060 es de 5.67 N/m^2 en donde el límite máximo de ruptura es de $4.937x10^5$ N/m^2. Cumple el estándar de diseño, tomando en cuenta que se aplica una carga máxima al sistema.

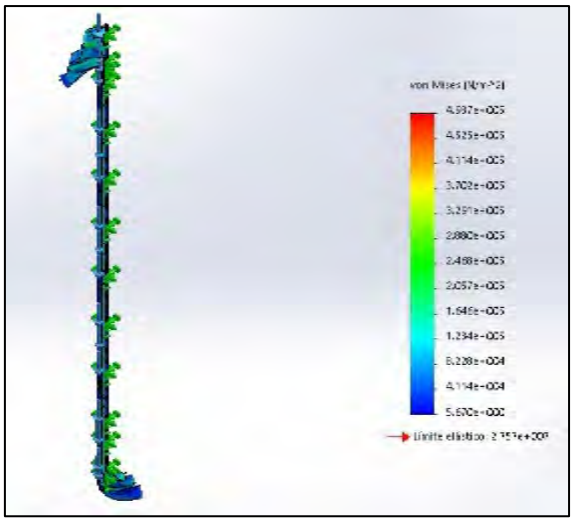


Figura 31 Tensión de Von Mises en la barra de desplazamiento

5.4 Diseño y construcción del acople

Una vez seleccionados el servomotor y el motor de inducción, sabemos claramente cada uno de los ejes de giro, sin embargo, la unidad lineal tiene un diámetro de 8mm por lo que necesita un acople adecuado que permita un agarre y giro adecuado sin perder fuerza. A partir de las medidas obtenidas de los planos del fabricante del motor a inducción y servomotor se procede a su diseño.

5.4.1 Acople caja reductora-unidad lineal

El diseño del acople es realiza en SolidWorks, en donde se opta por un diseño simple de acople, la caja reductora SITI es desmontable por lo que se parte desde estas medidas.

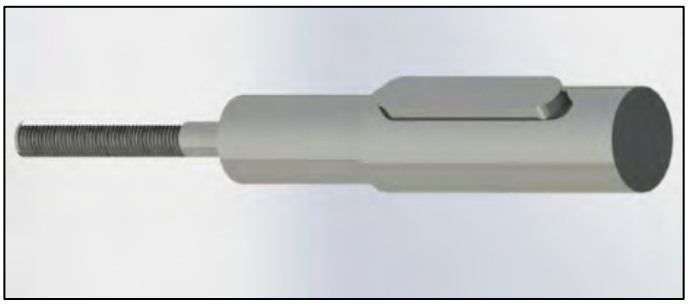


Figura 32 Acople Caja reductora-Unidad Lineal

El diseño parte desde un eje cilíndrico con una chaveta de sujeción para el eje de la caja reductora, ya que mecánicamente está diseñada para una fijación por chaveta, en la unidad lineal dispone de un orificio de 8mm si espacio para chaveta por lo que se decide diseñar un eje cilíndrico pasante con rosca M8 estándar para sujetar a través de una tuerca y contratuerca el eje de la unidad lineal. Diseño del acople en **ANEXO I**

Igual que en la barra deslizante se procede a realizar el análisis de esfuerzos para determinar el punto de ruptura del sistema, arrojando los siguientes datos.

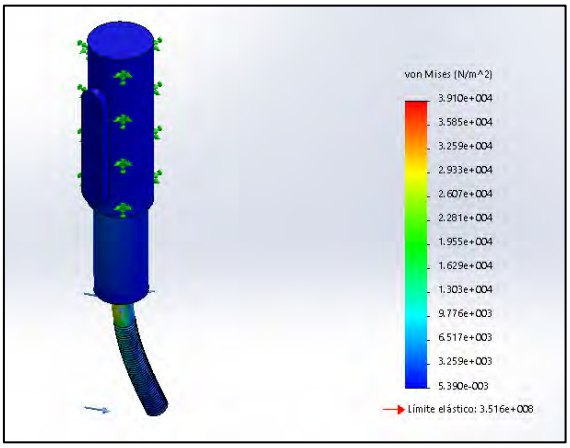


Figura 33 Tensión de Von Mises en el acople de la caja reductora-unidad lineal

Como se observa en la figura anterior, el punto de ruptura o falla sería en la reducción del eje de 20mm a 8mm, sin embargo, la simulación fue realizada con un par torsional de 3Nm, según el fabricante de la unidad lineal, el par máximo permitido en la correa es de 1.5Nm, por lo que cumple con los estándares de diseño, ya que, si el par aumentara, la cinta con diente de sierra se rompería antes de que llegue a su punto crítico de corte.

El estudio determina que el diseño es seguro para ser implementado, el prototipo es mandado a fabricarse, siendo este su resultado final.

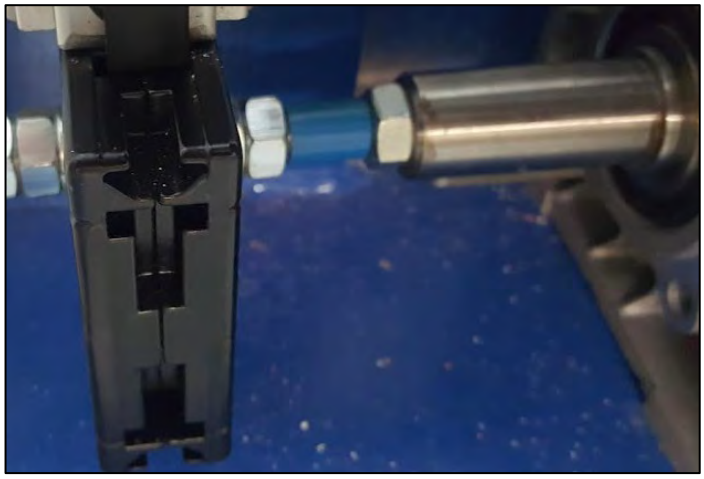


Figura 34 Implementación real del acople caja reductora-unidad lineal

5.4.2 Acople servomotor-unidad lineal

El diseño del acople al igual que el caso anterior se opta por un diseño simple de acople, el eje del servomotor es de 9mm y este no puede ser mecanizado debido a sus características de construcción. Diseño del acople en **ANEXO I**

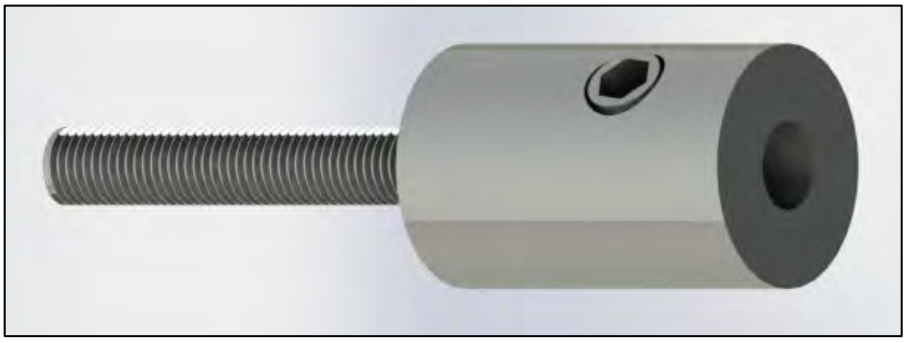


Figura 35 Acople Caja reductora-Unidad Lineal

El diseño parte desde un eje cilíndrico con un prisionero de sujeción para el eje del servomotor, debido que el eje al ser tan pequeño no dispone de chaveta, es necesario un ajuste por presión, se decide diseñar un eje cilíndrico pasante con rosca M8 estándar para sujetar a través de una tuerca y contratuerca el eje de la unidad lineal.

Igual que en la barra deslizante se procede a realizar el análisis de esfuerzos para determinar el punto de ruptura del sistema, arrojando los siguientes datos.

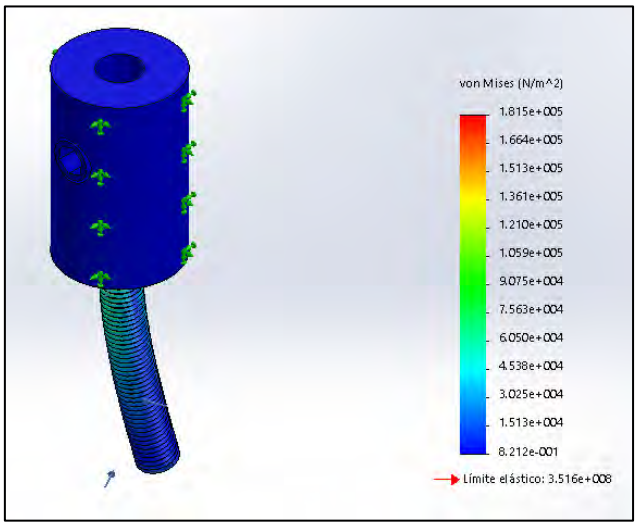


Figura 36 Tensión de Von Mises en el acople de la caja reductora-unidad lineal

Como se observa en la figura anterior, el punto de ruptura o falla sería en la reducción del eje de 9mm a 8mm, sin embargo, la simulación fue realizada con un par torsional de 3Nm, según el fabricante de la unidad lineal, el par máximo permitido en la correa es de 1.5Nm, por lo que cumple con los estándares de diseño, ya que, si el par aumentara, la cinta con diente de sierra se rompería antes de que llegue a su punto crítico de corte.

El estudio determina que el diseño es seguro para ser implementado, el prototipo es mandado a fabricarse, siendo este su resultado final.

Figura 37 Implementación real del acople servomotor-unidad lineal

5.5 Diseño de la estructura de sujeción y montaje de la unidad lineal

Para el montaje de la unidad lineal se toma como consideración la altura total de la unidad lineal, así como sus accionamientos, estos se ubicarán en la parte inferior del

sistema, a su vez es necesario dejar un espacio entre la pared y la unidad para colocar los sensores de posicionamiento y los cables de alimentación por lo que se decide designar un espacio de 4cm entre la pared y la unidad lineal, se ocupa un perfil cuadrado con una altura de 2.84 metros de altura, esto con el fin de dejar 4 cm de sobra a da lado de la unidad línea, además se añadirá un soporte para la fijación de los tornillos a la pared de 5cm de altura y una base para el descanso del motor a inducción del servomotor, diseño de la estructura para la sujeción a pared en **ANEXO I,** finalmente el diseño sería:

Figura 38 Elemento de sujeción para los motores y la unidad lineal

De la misma forma se procede a realizar el análisis de esfuerzos para determinar el punto de ruptura del sistema, arrojando los siguientes datos.

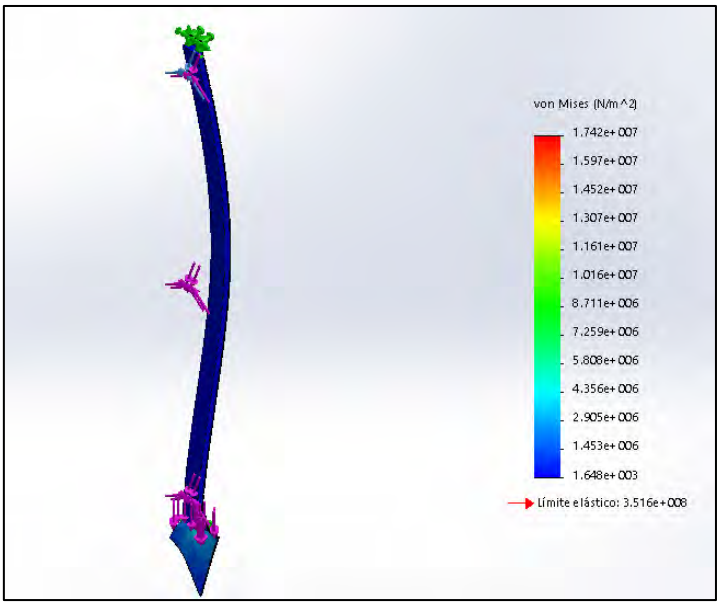


Figura 39 Tensión de Von Mises en la estructura de sujeción de la unidad lineal

Como se observa en la figura anterior, se implementa una fuerza total de 260N que simboliza el peso de la unidad lineal con carga máxima en los orificios de sujeción de la unidad lineal mientras que en la parte inferior se colocan 250N adicionales para un peso de los dos motores descansando en la estructura, los puntos de sujeción a la pared son tanto en la parte de arriba como en la de abajo con 2 tornillos de M14. El estudio determina que el diseño es seguro para ser implementado, el prototipo es mandado a fabricarse, siendo este su resultado final.

Figura 40 Implementación real de la estructura de sujeción en el laboratorio

6. Diseño Eléctrico y Electrónico

Tanto el servomotor como el motor a inducción necesitan accionamientos electromecánicos que regulen variables básicas en el movimiento del motor como velocidad angular, par, etc. Para lo cual necesitan una etapa electrónica de potencia que module la potencia de la red eléctrica de forma adecuada para controlar las variables

6.1 Selección de los motores y los accionamientos

En el prototipo se compararán dos motores, uno asincrónico de inducción y otro síncrono de imanes permanentes, es necesario tomar en cuenta el diámetro del eje en la unidad lineal, se necesitan motores con un eje de rotación menor a 12 mm y un par máximo menor a 3 Nm. Como se aprecia en la *Figura 28*, la gráfica de carga útil máxima es de 150 N para un par aplicado de 3 Nm, por lo cual se procede a investigar motores con estas características, cumpliendo los requisitos de diseño.

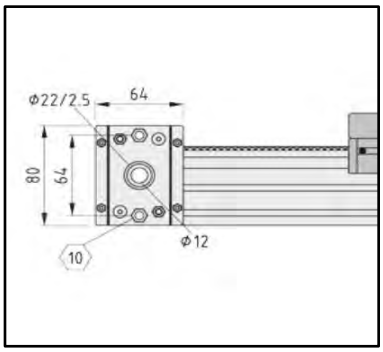


Figura 41 Especificaciones de diseño para el acople

Sin embargo, en la misma página web de la unidad lineal se oferta el servomotor apropiado para el control, pero estos tienen un costo elevado de adquisición y además solo serviría para esta aplicación específica. El proyecto pretende utilizar componentes que no están incluidos en el Kit de movimiento, es decir ocupar motores que puedan adaptarse al sistema y sean comerciales. Partiendo de esto, necesitamos un motor que tenga un diámetro de 8 mm y una longitud de 23 mm. La distancia restante será para implementar una chaveta para fijar el eje a la unidad lineal.

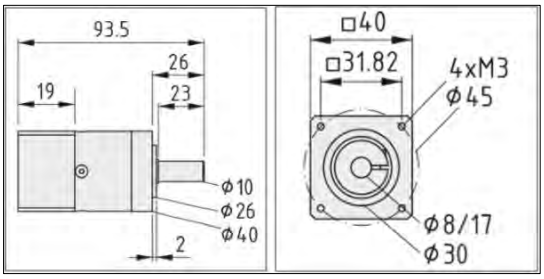


Figura 42 Dimensiones del motor sugerido por el Kit de sincronismo LRE 5 D6

Se realizo una amplia investigación de motores apropiados para el funcionamiento entre los cuales destacan los siguientes:

Tabla 3 Selección del motor de imanes permanentes

Motor	Par Nominal	Diámetro del Eje	Longitud del Eje
Schneider BMI0702P01A	2.2 Nm	11 mm	23mm
ABB BSM63N-333AA	2.09 Nm	10.9 mm	22.5mm
KEB A2SMHF2-84J0	0.8 Nm	9 mm	20.5mm
Festo EMMS-AS	2.29 Nm	9 mm	25mm

Tabla 4 Selección del motor a inducción

Motor	Par Nominal	Diámetro del Eje	Longitud del Eje
WEG Wquattro	3.53 Nm	19 mm	40mm
Siemens 1LE1001	1.15 Nm	18 mm	20mm
ABB 3GAA062	1.5 Nm	20 mm	20mm
REM MS 63B4	1.26 Nm	20 mm	25mm

6.1.1 Selección del motor asincrónico de inducción

Para la selección del motor a inducción se opta por el Motor REM MS 63B4, debido a que este brinda un rendimiento adecuado en altas y bajas velocidades con un costo inferior a los anteriores mencionados. Necesitará una caja reductora, con el fin de disminuir el diámetro el eje hasta las especificaciones deseadas.

Una de las principales ventajas de la caja reductora SITI MU 40 5/1 PAM 14/105 (plano caja reductora en Anexos), es que permite ser desmontada para acoplar cualquier tipo de eje. Por lo cual implementaremos un eje mecanizado desde la caja reductora hasta

la unidad lineal con un diámetro de 8mm y una longitud de 23 mm con una reducción del par de 5 a 1.

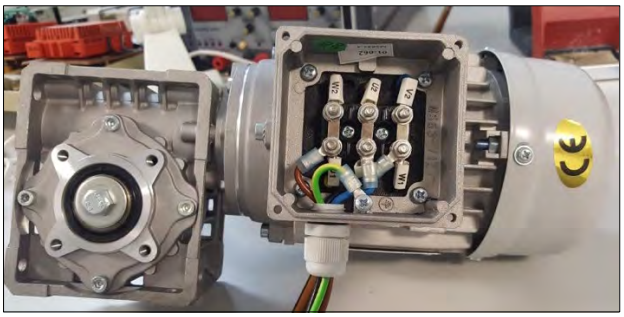


Figura 43 Motor Asincrónico a Inducción REM MS 63B4 y caja reductora SITI MU 40 5/1 PAM 14/105

La placa característica del motor a inducción tiene los siguientes datos:

- Tensión: Triángulo 230V, Estrella 400V
- Frecuencia: 50Hz
- Potencia: 0.18 kW con 0.25 Hp
- Velocidad Nominal: 1365 rpm
- Corriente: Triángulo 1.1A, Estrella 0.63A
- Factor de potencia: 0.64

Para calcular el número de pares de polos se toma en cuenta la velocidad más cercana a la nominal es decir 1500 rpm, partiendo de la fórmula de velocidad de sincronismo en donde:

$$n_{sinc} = \frac{60f}{p} \qquad 1500 = \frac{60 * 50}{p} \qquad p = 2\ pares\ de\ polos$$

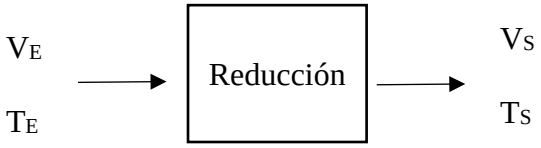
Para calcular el par que genera el motor a inducción es necesario implementar la siguiente ecuación:

$$T_{motor} = \frac{P_{motor}}{n_{sinc}}$$

$$T_{motor} = \frac{180}{1365 * \frac{2\pi}{60}}$$

$$T_{motor} = 1.26\ Nm$$

Mientras que al implementar la caja reductora el par de salida del motor a inducción se genera a partir de la siguiente relación:



$$V_S = \frac{V_E}{R_{trans}}$$

$$V_S = \frac{1350}{5} = 270\ rpm$$

Este valor de velocidad concuerda con el proporcionado en la tabla de valores de la caja reductora:

i	n_1	n_2	M_2	kW_1	HP_1	RD	sf
5	1400	280	17	0,55	0,75	0,90	2,69
7,5		187	25	0,55	0,75	0,87	1,84
10		140	32	0,55	0,75	0,86	1,39
15		93	46	0,55	0,75	0,82	0,98
20		70	39	0,37	0,50	0,77	1,11
25		56	32	0,25	0,34	0,75	1,21
30		47	35	0,25	0,34	0,74	1,23
40		35	46	0,25	0,34	0,67	1,00

Figura 44 Datos técnicos de la caja reductora SITI MU40

Para calcular el par se toma en cuenta el rendimiento indicado en la figura anterior, por lo que el par resultante será de:

$$T_s = T_E * \eta * i$$

$$T_s = 1.26 * 0.9 * 5$$

$$T_s = 5.67\ Nm$$

6.1.2 Selección del variador de frecuencia

Para variar la velocidad del motor asíncrono habría que utilizar un instrumento electrónico el cual se encargue de alimentar y controlar la frecuencia por el sistema de control, para lo cual es necesario el uso de un variador de frecuencia. Utilizaremos el control de frecuencia escalar para variar la velocidad del motor a inducción, el control escalar se basa en modificar la frecuencia de alimentación con la que se alimenta el estator del motor, ya que esta frecuencia modifica la velocidad de sincronismo del campo giratorio y por tanto la velocidad mecánica de giro cercana a la de sincronismo en función del deslizamiento del motor. Existen diferentes marcas de variadores de frecuencia que ofrecen una amplia gama de aplicaciones y funciones, en nuestro caso al ser un sistema que controlará la velocidad de giro, el arranque y sentido del motor a inducción, se opta por un modelo que posea las características suficientes para el diseño, sin embargo, además poseerá todas las seguridades industriales de un variador.

Tabla 5 Selección de variador de frecuencia

Nombre	Tensión	Potencia	Control
Siemens SINAMICS V20	Mono/trifásica	15kW	Escalar
ABB ACS150	Monofásica	4kW	Escalar
Delta VFDL	Monofásica	0.75kW	Escalar
Danfoss VLT2800	Mono/trifásica	22kW	Escalar
Weg CFW300	Monofásica	0.8kW	Escalar
Mitsubishi FRF740	Mono/trifásica	20kW	Escalar

El convertidor de frecuencia CFW300 *Figura 42* es un accionamiento de alta performance desarrollado para el control de variación de velocidad de motores de inducción trifásicos, ideal para aplicaciones en máquinas o equipos que necesitan control preciso y facilidad de operación. Posee tamaño compacto, instalación eléctrica similar a contactares, control vectorial WEG (VVW) o escalar (V/F) seleccionable, interfaz de operación (HMI) incorporada, SoftPLC, software de programación WPS gratuito, y accesorios tipo plug-in que pueden ser incorporados al variador, agregando más funcionalidad y proporcionando una solución flexible al control de motor a inducción.

Figura 45 Variador de frecuencia WEG cfw300

Señales de control en el variador de frecuencia

El variador de frecuencia, posee un hardware robusto, el sistema costa de 4 entradas digitales, una puesta a 0 V, una entrada analógica de voltaje y de amperaje. Dos interruptores, uno normalmente cerrado y uno abierto y una señal de tensión de 10V.

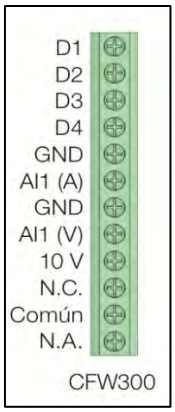


Figura 46 Señales de entrada y salida en el variador de frecuencia WEG

Para la implementación del proyecto se decide utilizar 2 entradas digitales D1 y D2, estas están conectadas a los relés y como común la señal GND como se aprecia en el **ANEXO 14.5,** estas entradas controlarán el sentido de giro del motor y el paro/marcha del mismo, además son implementadas 2 entradas auxiliares la D3 y D4 para que pueda incrementar las capacidades del proyecto a futuro, es decir puedan implementarse rampas variadas de aceleración, velocidades variables, etc. Como se puede observar en la siguiente figura.

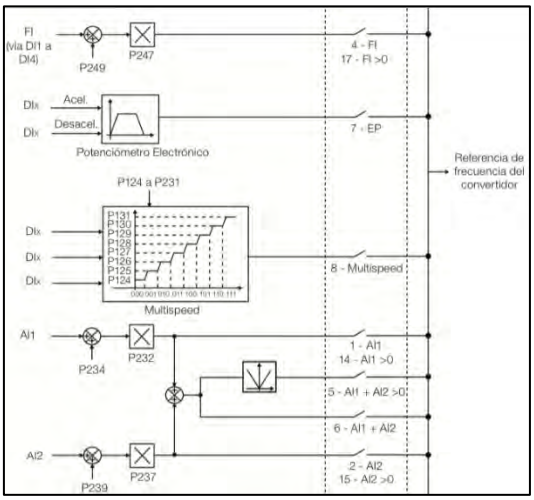


Figura 47 Tipos de señales en el variador WEF

Para la referencia de velocidad se ocupa una entrada analógica, esta señal trabaja en un rango de entrada de 0 a 10V, las tarjetas de control Arduino y Raspberry tienen una salida máxima de 3.3V por lo que es necesario a través de la parametrización del variador, configurar un multiplicador de señal que amplifique la referencia analógica entrante.

Se implementa la expansión CFW300-IOAENC, esta expansión permite utilizar un encoder con entradas A, A-, B, B-, con el fin de mejorar el control de posición regulando la velocidad de forma exacta. Dispone adicionalmente de 2 salidas y entrada analógicas, estas puedes ser útiles para la motorización, a través de la tarjeta para futuras mejoras se

podría visualizar una gráfica representativa de la velocidad recibida y el voltaje aplicado por el variador de frecuencia, aunque en el caso de la Raspberry sería necesario utilizar un conversor analógico digital ya que dispone de entradas y salidas GPIO para su control y programación.

Figura 48 Módulo encoder CFW300-IOAENC

6.1.3 Conexión del motor asíncrono de inducción

Para comenzar la conexión, se parte de una alimentación de 220V, debido a que el variador de frecuencia trabaja a dicha tensión, de sus bornes de salida de voltaje UVW, se conecta un contactor en caso de que llegue al final de carrera, este corte la alimentación del motor de inducción, una vez se realizada esta conexión se procede a conectar el motor a las salidas 2,4,6 del contactor de seguridad. Del manual de programación CFW300 v1.3X (ver en anexos), la figura 3.3 de dicho manual nos detalla el diagrama de bloques del variador de frecuencia, para el proyecto ocuparemos las entradas digitales D1 para el sentido de giro y D2 para marcha/paro del motor a inducción, además conectaremos dos entradas auxiliares D3 y D4 para futuras implementaciones de un control por frecuencias fijas, encendido y apagado, selección de rampa o cualquier implementación futura que se desee realizar. Para el control de velocidad se ocupará la entrada analógica A1, desde el sistema de control. Parametrización en **ANEXOS IV**

Figura 49 Partes del variador de frecuencia WEG cfw300

Al utilizar una red de alimentación de 220V, es necesario conectar el motor asincrónico de tal forma que funcione adecuadamente de acuerdo a la red suministrada. Para lo cual se procede a realizar la conexión en triángulo como se ve en la *figura 41*, de acuerdo a la placa característica del motor a inducción, con esta conexión la alimentación necesaria hacia el motor será de 220V.

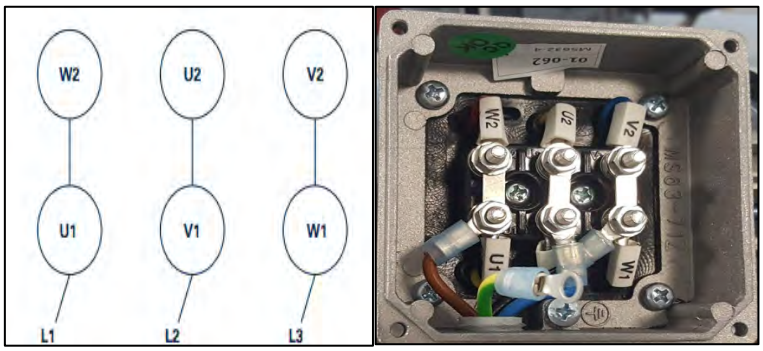


Figura 50 Conexión triángulo del motor a inducción trifásico

Finalmente, en la *figura 43* se puede apreciar el esquema de conexión del variador y del motor a inducción con sus entradas de control y seguridad, todo este sistema está alimentada a 220V con puesta a tierra. El esquema completo se encuentra en los anexos.

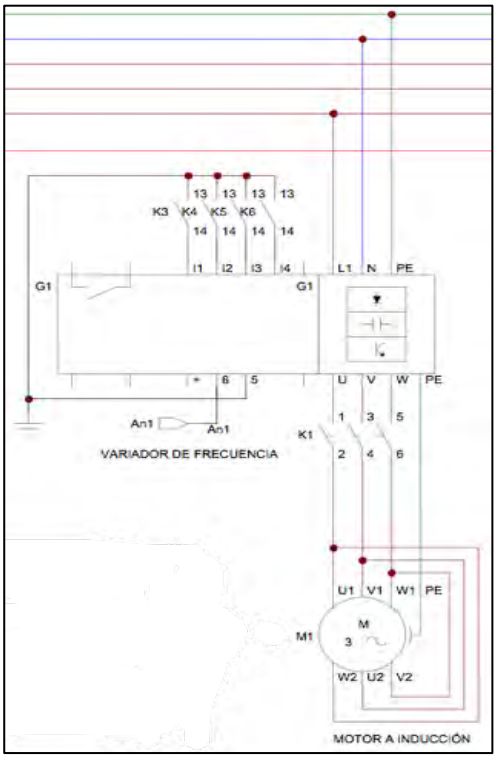


Figura 51 Esquema de conexión del motor de inducción (Esquema completo en Anexos III)

6.1.4 Selección del motor sincrónico de imanes permanentes

En el caso de los servomotores es necesario un accionamiento de avance que sea compatible con el mismo. Tomando en cuenta eso será necesario utilizar el kit completo que incluiría servomotor, servo drive y encoder o resolver. Ya que el servomotor trabaja siempre en bucle cerrado.

KEB es una empresa la cual brinda equipos de automatización y accionamientos con un coste de adquisición razonable. Además de que posee una central de ayuda y mantenimiento en España. Se opta por el modelo A2SMHF2-87J0, que incluye un resolver en su diseño con un control a través del servo drive Combivert S6 K.

Figura 52 Motor sincrónico de imanes permanentes A2SMHF2-87J0

Utilizando el manual de servomotores KEB Dinamic Line III (en anexos), podemos determinar los parámetros del servomotor, del apartado 2.3 determinamos:

- Sincrónico de tres fases
- Auto enfriamiento con brida
- Norma de construcción: IP54
- Velocidad nominal: 8000 rpm
- Tensión nominal: 400V
- Par: 0.8 Nm
- Potencia: 0.59 kW
- Intensidad: 1.5 Arms
- Incluido resolver de 2 polos

Además de estos datos en el apartado 5.4 del mismo documento nos encontramos con las especificaciones técnicas del servomotor, cabe recalcar que en el software COMBIVIS KEB, se encuentran todos los parámetros establecidos del servomotor.

Motor type		A1	A2	A3
Stall torque	M_0 / Nm	0,5	0.8	1.21
Current at stall torque	I_0 / A	0.85	1.5	2.2
Rated motor frequency	f / Hz		400	
Rated current	I_N / A	0.85	1.3	1.85
Max. torque	M_{max} / Nm	2.69	4.18	6.36
Max. current	I_{max} / A	4.9	7.7	11.4
Max. speed Mech	n_{max} / rpm		12000	
Winding resistance 1)	R_{u-v} / Ω	39.4	13.2	8.5
Winding inductance 1)	L_{u-v} / mH	82.4	36.8	25.2
Voltage constant 1)	k_e / V_{pk}/1000 rpm	52	49	52
Rotor inertia 2)	J_L / kgcm²	0.134	0.253	0.373
Ground 2)	m / kg	1.0	1.3	1.7
Pole-pair number	p		3	

Table 10: Technical data servo motors AxSMHFx-xxxx

Figura 53 Especificaciones técnicas del servomotor A2SMHF2-87J0

Una de las principales diferencias entre los motores de inducción y los servomotores es que a velocidad 0 el servomotor conserva su par, por tal motivo la gráfica par velocidad será lineal en todo momento de forma teórica, pero en el manual del servomotor el par cambiará en función a la velocidad de giro como se puede apreciar en la *figura 27*.

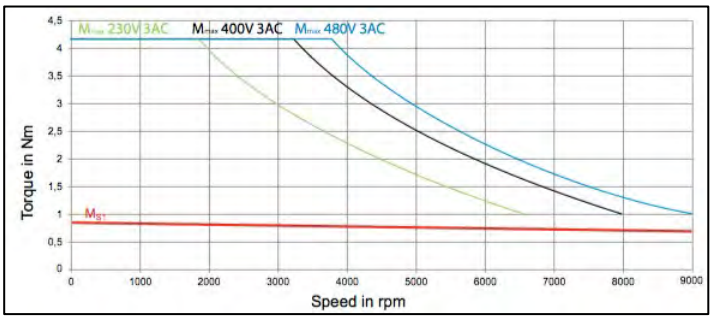


Figura 54 Servomotor: Par útil en función a la velocidad a 400V trifásico

En el caso de la intensidad, la relación intensidad/intensidad de salida viene dada en la siguiente gráfica:

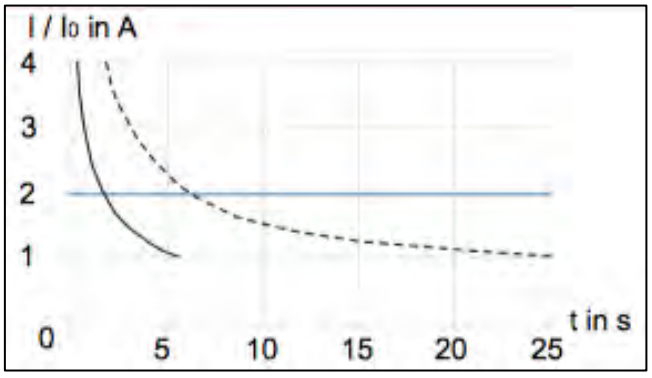


Figura 55 Servomotor: Relación de intensidad en función al tiempo

Verificamos que el motor sirve para la unidad lineal debido a los datos de fuerza axial y radial permitida como se ve en la figura 26.

Motor type	n_{max} / rpm	$F_{R\,max}$ / N	F_R in dependence of n					F_A / N	$F_{A\,max}$ / N
			20 % * n_{max}	40 % * n_{max}	60 % * n_{max}	80 % * n_{max}	100 % * n_{max}		
01SMHF	8000	320	168.7	133.9	117	106.3	98.7	0.2 * F_R	160
02SMHF	8000	300	184.5	146.4	127.9	116.2	107.9		
03SMHF	8000	280	194.2	154.1	134.7	122.3	113.6		
A1SMHF	8000	500	240.0	190.5	166.4	151.2	140.4	0.2 * F_R	340
A2SMHF	8000	470	268.5	213.1	186.2	169.2	157.0		
A3SMHF	8000	420	286.0	227.0	198.3	180.2	167.3		

Figura 56 Servomotor: Fuerza radial y axial permitida.

6.1.5 Selección del convertidor de frecuencia para motor de imanes permanentes

Un servo drive es un amplificador electrónico especial utilizado para alimentar servomecanismos eléctricos. Un servo drive controla la señal de retroalimentación del servomecanismo y se ajusta continuamente a la desviación del comportamiento esperado.

Tabla 6 Selección del servo accionamiento

Nombre	Tensión	Potencia	Software
Siemens SINAMICS V90	Trifásica	7kW	Tia Portal
Omron Accurax G5	Trifásica	15kW	CX-Drive
ACOPOS 8V10	Trifásica	0.45kW	ACOPOS Server
KEB Combivert S6K	Trifásica	1.8kW	Combivis 6
ABB MicroFlex e100	Trifásica	2kW	NextMove 2.0

El servomotor KEB, necesita de una tarjeta controladora o servo drive, esta tarjeta viene recomendada en el kit de control, para lo cual utilizaremos la tarjeta Combivert S6 K control, esta tarjeta consta de un monitor de temperatura, entradas/salidas digitales y analógicas, entradas de seguridad a 24V, un bus can, entrada para 2 encoder o resolver.

Figura 57 Combivert S6 k control

El servo drive se puede alimentar a 220V o a 400V, pero el servomotor trabaja a 400 V en su tensión nominal para lo cual optaremos por la conexión de 400V, como se

puede apreciar en la siguiente figura, de acuerdo al manual de instalación INSTALLATION S6 K-Control. (manual completo en Anexos)

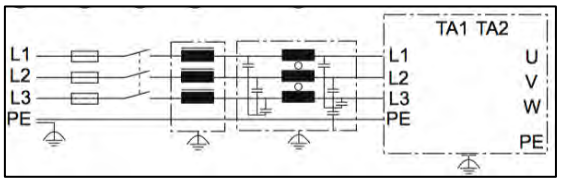


Figura 58 Esquema de conexión del servo drive según manual de instrucciones S6k-control

Una vez configurado el servo drive se procede a constatar su correcto funcionamiento tomando en cuenta los leds de seguridad que incluye este sistema, en el apartado 2.3.8.1 del manual se detalla las posibles alarmas del servo drive, como se ve en la siguiente figura, se puede apreciar cuando el servo drive está listo para funcionar. Es necesario aclarar que para su correcto funcionamiento necesita una alimentación de control externa de 24V, y sus pines STO debidamente conectados al positivo y negativo de la fuente de voltaje. Esto con el fin de que las luces leds, se enciendan en los colores adecuados para su funcionamiento.

VCC ● NET ● ST DEV ● ST OPT ○	Ready for operation	Device is ready for operation and the LEDs start with their normal function (approx. 3 s).

Figura 59 Leds indicadoras del estado servo drive

6.1.6 Señales de control en el servo drive

La gama Combivert S6, está diseñada para trabajar con múltiples señales de entrada y salida, es necesario tomar en cuenta la tabla de características del servo drive.

Inverter size				07	09	10
Housing type				2		
Output rated power		Sout	kVA	1.8	2.8	4.0
Max. rated motor power		Pmot	kW	0.75	1.5	2.2
Output rated current		IN	A	2.6	4.1	5.8
Output voltage		Uout	VAC	0...Uin or 0...Uindc/√2		
Output phases				3		
Output frequency		fout	Hz	0...599		
Overload current (60s)		I60s	%	200	200	200
Overload current (3s)		I3s	%	250	250	250
Overcurrent		IOC	%	300	300	300
Rated switching frequency		fSN	kHz	8	8	8
Input rated current		Iin	AAC	3.6	6.0	8.0
Rated input voltage		UN	VAC	400		
Input voltage range		Uin	VAC	184...550		
Input voltage range DC		Uin_dc	VDC	260...750		
Input phases				3		
Mains frequency		fN	Hz	50 / 60		

Figura 60 Tabla características del Servo drive Combivert S6K

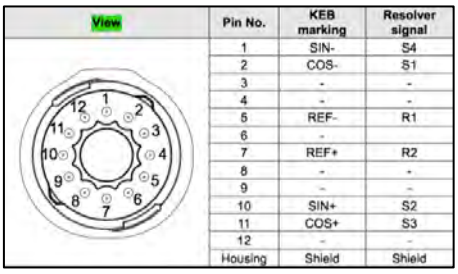
Al ser un servoamplificador moderno, es compacto y flexible para niveles de rendimiento de 0.75kW a 7.5kW o corrientes eléctricas de 2.6A a 12.5A. Además, el controlador de accionamiento proporciona un rendimiento óptimo en el control de par, velocidad y posición. El combivert S6 maneja el control para motores de CA asíncronos

o servomotores síncronos. La plataforma de accionamiento modular y la consistencia estricta garantizan una base uniforme en toda la gama. El combivert S6 también ofrece una función de comunicaciones y seguridad diseñada para adaptarse a este proyecto. Las partes del servo drive se pueden apreciar en la siguiente imagen. Parametrización en **ANEXOS IV**

Figura 61 Partes del Servo drive

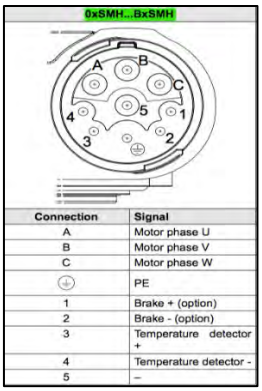
6.1.7 Conexión del motor sincrónico de imanes permanentes

Al momento de conectar el servo drive al servomotor, dirigirse al apartado 4.3 del manual Dynamic Line III, en este se especifica los pines de salida y de entrada tanto de la etapa de potencia como del resolver del servomotor. Tomar en cuenta que los cables de conexión llevan una numeración referente a la conexión del servomotor, verificar que los cables recubiertos por aislante térmico son los detectores de temperatura y los cables con un diámetro más grueso se refieren a los cables de alimentación UVW a 400V.



Pin No.	KEB marking	Resolver signal
1	SIN-	S4
2	COS-	S1
3	-	-
4	-	-
5	REF-	R1
6	-	
7	REF+	R2
8	-	-
9	-	-
10	SIN+	S2
11	COS+	S3
12	-	-
Housing	Shield	Shield

Figura 62 Pines de conexión del resolver en el servomotor KEB



Connection	Signal
A	Motor phase U
B	Motor phase V
C	Motor phase W
⏚	PE
1	Brake + (option)
2	Brake - (option)
3	Temperature detector +
4	Temperature detector -
5	–

Figura 63 Pines de conexión del servomotor KEB

La conexión del servomotor se puede apreciar en la siguiente figura en donde se conectan los pines de alimentación a 400V, la entrada digital I1 es la encargada del giro FORWARD y la entrada digital I3 se destina al giro REVERSE, en la configuración de la palabra de seguridad, se ejecuta el comando para que el motor se encuentre siempre en run, además se ocupa una entrada analógica la cual será la encarga de controlar la velocidad del servomotor. Los pines STO+ 4 y 2 están conectados a una fuente externa de 24V y los pines STO- 3 y 1 están conectados a tierra. Este sistema también posee una conexión a un contactor el cual es desactivado por el final de carrera en caso de que el sistema llegue al final de su recorrido sin un control, la tensión se desactiva automáticamente. De la misma forma que en el caso del variador de frecuencia se conectan 2 entrada la I5 e I7 de forma auxiliar para futuras implementaciones en donde puedan ajustarse parámetros adicionales como rampas de aceleración o tiempo de funcionamiento.

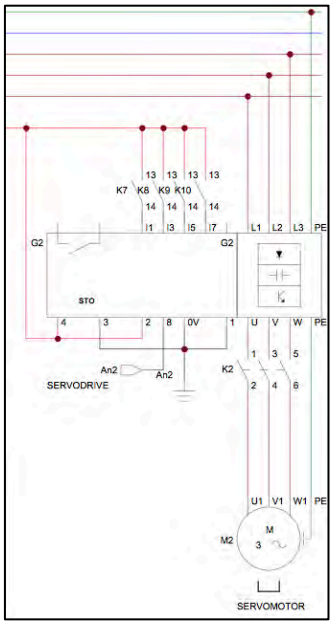


Figura 64 Esquema de conexión del servomotor (Esquema completo en Anexos III)

6.2 Selección y conexión de sensores para control

Un sensor es un dispositivo que transforma una magnitud física en una señal eléctrica, para el proyecto es necesario la implementación de sensores que controlen el posicionamiento del elevador, para lo cual se procede a determinar que sensor es el adecuado.

6.2.1 Posicionamiento

Los sensores de distancia y transductores, son utilizados para medir una distancia especifica o un desplazamiento lineal, existen dos tipos de sensores de distancia los cuales son muy utilizados, los sensores digitales funcionan al enviar una señal y calcular el tiempo que se demora en recibir la misma, esto hace que el sensor sea de carácter digital, otros como los sensores de distancia analógicos poseen un fototransistor que recibe la señal enviada por un emisor, haciendo que el voltaje entre la base y el colector varíe en función a la distancia, se considera ideal para medir distancias con precisión, aunque su desventaja es el rango de medición existente ya que a mayor distancia existe un fallo de medida, aparte de los fallos de histéresis.

El uso de sensores capacitivos es otra opción presente, estos envían una señal eléctrica en presencia de un objeto, tanto de distancia axial como radial. La principal aplicación en la actualidad es en dispositivos móviles, detención de nivel, sensor de humedad y detección de posición. Este dispositivo detecta cualquier tipo de objeto presente en el movimiento tanto radial como axial, una de sus principales deficiencias es el alcance debido al diámetro del sensor, la masa a detectar también es otra de sus deficiencias ya que para una detección específica depende de una constante eléctrica.

Se opta por utilizar un sensor inductivo, debido a que la unidad lineal es de aluminio y esta clase de sensores especiales sirven para detectar materiales ferrosos, su implementación es industrial tanto para aplicaciones de posicionamiento como para detectar la presencia o ausencia de algún material metálico. Para la distancia de medición del sensor inductivo se parte de la construcción de la unidad lineal, el rango de detección adecuado debe ser entre 5 a 8 mm desde la cara de la unidad lineal, su tensión de alimentación no debe superar los 24V, ya que todo el proyecto esta referenciado en ese rango de voltaje.

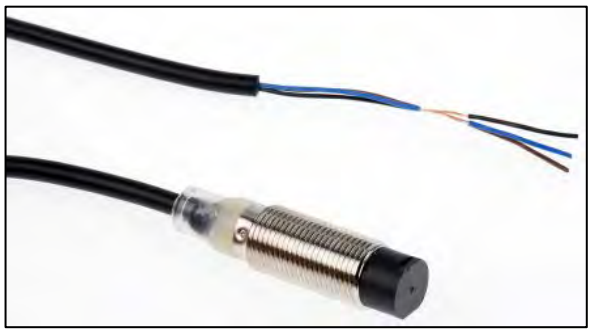


Figura 65 Sensor Inductivo OMROM E2B

El sensor de proximidad cilíndrico E2B es implementado en el proyecto, debió a su alta fiabilidad para aplicaciones industriales con una buena relación calidad precio. El sensor al ser no blindado posee un rango de medición más amplio, con un mejor resultado en la detección de presencia, sus características técnicas son las siguientes.

- Distancia de detección: 8 mm
- Tamaño: M12 no protegido
- Material: Latón
- Longitud del cuerpo: Corto
- Cantidad de hilos: 3
- Tensión de alimentación: 12-24V
- Corriente: 200mA
- Salida: PNP normalmente abierto

Es necesario determinar el rango de funcionamiento de acuerdo a la gráfica de medición en el sensor, como se puede ver en la siguiente figura, la distancia de detección para el modelo E2B-M12 N8 es desde 0 hasta 8mm en el eje X mientras que en el eje Y, el que va colocado en la carilla es pequeño con un valor máximo de 5mm, para lo cual en su implementación se colocara a una distancia prudente para evitar medidas erronas.

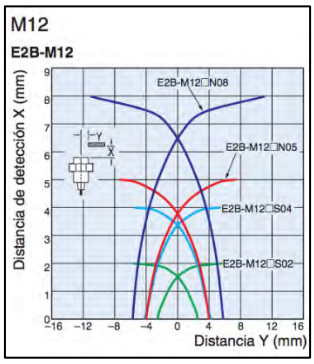


Figura 66 Rango de medición del sensor inductivo E2B

De acuerdo a especificaciones de diseño, ocuparemos 10 sensores de este tipo, 5 por cada unidad lineal, colocados a una distancia equidistante para que el ascensor tenga un rango considerable de pisos. A partir de lo mencionado se procede a realizar el esquema de conexión.

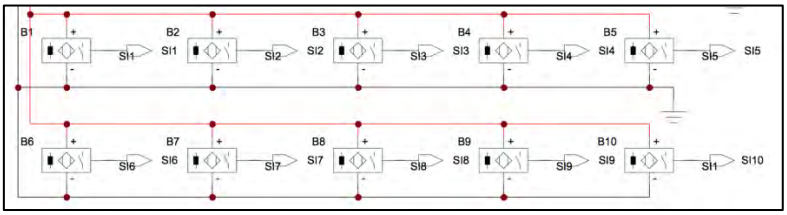


Figura 67 Esquema de conexión de los sensores inductivos E2B

La señal de salida en los sensores al ser PNP dependen de la señal de alimentación, por lo que su salida es de 24V, pero las tarjetas Arduino y Raspberry trabajan con tecnologías lógicas TTL (lógica transistor transistor) es decir con valores binarios en el rango de 0 a 0.8V como 0 lógico y 2 a 5V como 1 lógico.

Es necesario utilizar divisores de tensión en cada sensor, a partir de las fórmulas implementadas como se pueden ver en la siguiente figura obtenemos que:

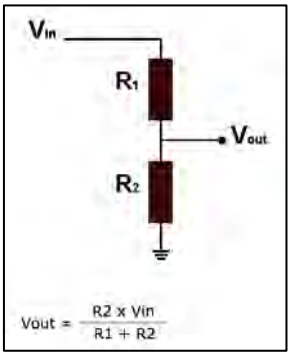


Figura 68 Esquema del divisor de tensión

$$V_{out} = \frac{1500}{1500 + 10000} * 24$$

$$V_{out} = 3.13V$$

Por lo tanto, se ocuparán resistencias de 10K y 1.5K en los divisores de todas las entradas digitales, tanto para la Raspberry como para el Arduino.

6.2.2 Sensores fin de carrera

Generalmente lo sistemas de accionamientos industriales es necesario utilizar dispositivos de seguridad eléctricos, al ser una unidad lineal y controlar el giro de motores de potencia, utilizamos sensores de fin de carrera, estos dispositivos son sensores de proximidad al contacto que constan de un actuador mecánico el cual

habilita o deshabilita el paso de tensión a través de sus pines de conexión, en la unidad lineal serán colocados en la parte superior e inferior de cada unidad lineal para que una vez supere los límites de recorrido establecido en la unidad lineal estos corten la tensión de alimentación de los motores.

Figura 69 Sensor Cherry fin de carrera DC SERIES

Implementamos los sensores Cherry DC2 Series, debido a sus características de diseño son óptimas para la unidad lineal, entre sus principales características se encuentran:

- Tensión de funcionamiento máximo: 250 VAC
- Temperatura máxima de operación: 120 C
- Ciclos de operación: 10000
- Corriente máxima: 10A
- Terminales: Alimentación, común, NO y NC

Para su implementación se partir de una conexión serie hasta una bobina, esta permanecerá alimenta, a su vez los contactores del sistema se cerrarán para alimentar los bornes UVW en los motores a inducción y de imanes permanentes. Una vez se abran los pines NC de los sensores, se dispondrá al corte de energía en los motores y a su vez se enviará una señal a las tarjetas de control. Como se puede ver en la siguiente figura, los sensores fin de carrera de las 2 unidades lineales son alimentadas por una fuente de 24V, para enviar la señal a las tarjetas se utiliza divisores de tensión previamente calculados para que sea compatible con la lógica TTL de las mismas.

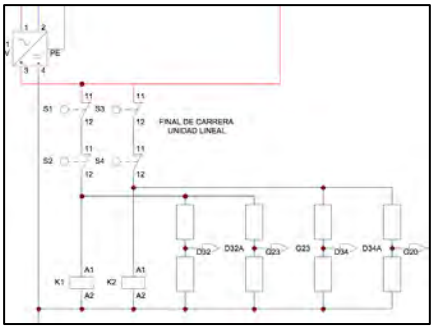


Figura 70 Esquema de conexión sensores fin de carrera

6.3 Selección y conexión de sensores para osciloscopio

En el trabajo fin de máster también se desarrolla un prototipo encargado de la monitorización del estado del motor a modo osciloscopio, lo que se pretende diseñar es un dispositivo el cual ayude a verificar la tensión, corriente, par y velocidad ya que se utilizan dispositivos como tacómetros y osciloscopios digitales para este proceso como se aprecia en la siguiente figura.

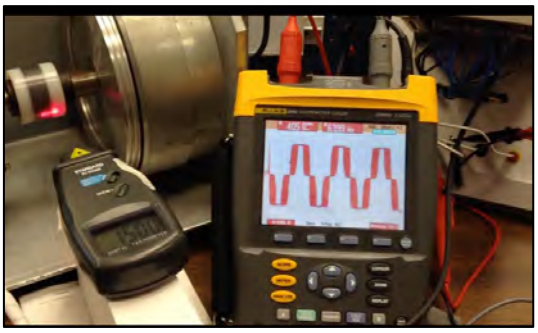


Figura 71 Método manual de monitorización

Para lo cual es necesario utilizar sensores que permitan un acceso a la tensión y corriente sin poner en peligro al usuario ni al dispositivo de control, tomando en cuenta estos parámetros se procede a la selección de los sensores correspondientes.

6.3.1 Selección del sensor de Corriente y Par

El uso de una pinza amperimétrica es un tipo especial de amperímetro que permite medir la corriente que pasa por el circuito sin la necesidad de abrir el mismo, en el proyecto utilizaremos el mismo principio de funcionamiento con la implementación de un sensor transductor que permita medir corrientes continuas y alternas. En la siguiente tabla se pueden apreciar los distintos tipos de sensores utilizados para medir corriente.

Tabla 7 Selección del sensor de corriente

Sensor	Voltaje de operación	Tipo de salida de señal	Corriente máxima
PCS301P	3.3	Analógica	4A
ACS712	5V	Analógica	5A
SCT013-030	5V	Digital	10A
ACS714	3.3V	Analógica	7A
Varicel RSE 0-40	10V	Analógica	20A
LTS 6NP	5V	Analógica	19A

LEM se considera una empresa líder a nivel mundial en sensores para medición de parámetros eléctricos, estos se consideran innovadores y de alta calidad, al alimentar el sistema con 5V, se implementa el sensor LEM LTS 6NP debido a que este sensor trabaja a bucle cerrado con el uso de un transductor de efecto Hall, soportando un voltaje unipolar apropiado para un diseño en PCB. Las principales características técnicas del sensor son:

- Corriente nominal: 6A
- Rango de medición: 0 a +-19.2A
- Sensibilidad: 104.16 mV/A
- Tensión de alimentación: 5V DC

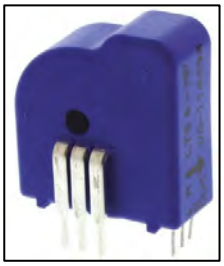


Figura 72 Sensor LTS 6NP

Cabe recalcar que el principio de funcionamiento correspondiente del sensor es, a partir de sus 3 pines de salida, en donde uno es la tensión de alimentación, tierra y del pin de salida se obtener una variación de señal positiva y negativa a partir de 2.5V, como se puede apreciar en la siguiente gráfica. Por lo que es necesario en el control restar este offset para determinar el amperaje del montacargas en funcionamiento.

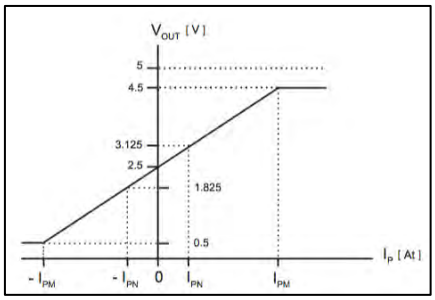


Figura 73 Voltaje de salida en función al amperaje medido con el sensor LTS 6NP

6.3.2 Conexión del sensor de Corriente

Se implementarán dos sensores de corriente, uno será el encargado de medir la corriente que se genere en una de las fases de los motores, el otro también medirá la

corriente, pero a través de la relación corriente-par de 15.5/14.4 transformará y visualizará este valor en la aplicación correspondiente. El esquema de conexión de los sensores de corriente se puede apreciar en la siguiente imagen, esquema completo en **ANEXO III.**

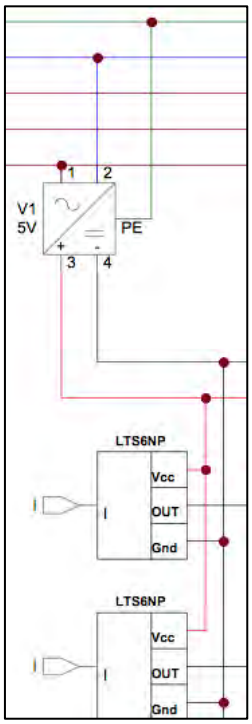


Figura 74 Esquema de conexión sensores de corriente LTS6NP

6.3.3 Selección del sensor de Voltaje

La implementación del sensor de voltaje se hace a través de un conversor analógico digital, debido a que la tarjeta seleccionada posee una pobre resolución de bits, a mayor cantidad de bits más precisión en los valores de voltaje transformará y visualizará este valor en la aplicación correspondiente. En la siguiente tabla se pueden apreciar los distintos tipos de sensores utilizados para medir corriente.

Tabla 8 Selección de sensor de voltaje

Nombre	Resolución	Voltaje de operación
REM ADI2	12 bits	5V
AD9215BCPZ-105	10 bits	5V
AMC1305M25DW	12 bits	5V
LTC2486IDE#PBF	16 bits	12V
MCP3208	12 bits	5V
MCP3304	13 bits	5V

Microchip Technology empresa mundialmente conocida en microcontroladores, conversores y reguladores son los encargados de desarrollar conversores ADC, al alimentar el sistema con 5V, se implementa el sensor MCP3308 debido a que este sensor posee una alta resolución y funcional en el voltaje de alimentación adecuado, uno de sus bits es del signo de la señal mientras que los otros son del valor de adquisición de la señal analógica, es importante tomar en cuenta la velocidad de muestreo, debido que al implementarse una aplicación, necesitamos un muestreo exacto por lo cual según el datasheet del fabricante posee un periodo por muestra de 40microsegundos.

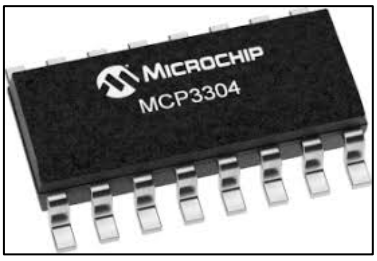


Figura 75 Conversor Analógico digital MCP3304

6.3.4 Conexión del sensor de voltaje

El Bus SPI (del inglés Serial Peripheral Interface) es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj (comunicación sincrónica). (Llamas, 2018)

Este microchip utiliza una comunicación SPI por lo que sus pines correspondientes de comunicación irán a la tarjeta, por otro lado, tiene 8 canales analógicos, en el proyecto se utilizarán 3. Se implementarán un divisor de tensión y un sumador, el divisor de tensión se utiliza para que la entrada de voltaje no supere los 5 V, mientras que el voltaje sumado es para compensar el offset de la señal alterna generada. El esquema de conexión del sensor de tensión se puede apreciar en la siguiente imagen, esquema completo en **ANEXO III.**

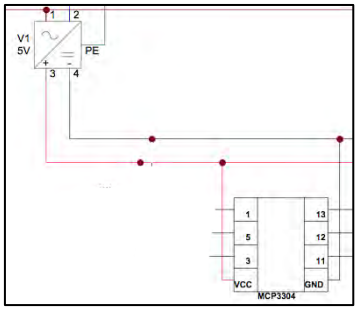


Figura 76 Esquema de conexión sensores de corriente LTS6NP

7. Diseño de Control

A través del sistema de control podemos gestionar todos los actuadores y utilizar las señales de los sensores para hacer un sistema robusto en el ascensor montacargas, a partir de los sensores inductivos, podremos determinar en qué planta se encuentra el ascensor y a través del control determinar el sentido de giro de los motores y la velocidad de desplazamiento de los mismos.

Existen dos tipos de sistemas de control, en lazo abierto y lazo cerrado, utilizaremos un servomotor, este sistema es en bucle cerrado, es decir que posee un sistema de retroalimentación a través de un resolver el cual indica la posición del servomotor de forma precisa y controlado. Estos datos son regulados mediante el servo drive, de forma simultánea un variador de frecuencia sigue una referencia de velocidad a partir de la tensión señalada, sin embargo, para convertirlo en un sistema en bucle cerrado necesitamos un encoder conectado a la caja reductora.

Las tarjetas de hardware y software abierto, van a ser utilizadas para controlar las entradas digitales y analógicas tanto del servo drive como del variador de frecuencias, el voltaje de funcionamiento del proceso como se vio en el apartado anterior es de 24V mientras que del control es de 5V de alimentación, por lo que se diseñará un control adecuado a través de relés que separen ambas tensiones.

7.1 Selección del control

Para la implementación de las tarjetas de control es necesario comparar las distintas tarjetas del mercado para lo cual a partir de una investigación rigurosa se obtiene la siguiente tabla:

Tabla 9 Tabla comparativa de tarjetas de control

PLACAS	LIMITE	PIN DIGITALES	PIN ANALOGICO	PIN PWM	VEL_RELOJ	INFORMACION EXTRA
Arduino Due	12V_800 mA	38	12	16	84Mhz	2 salidas DAC, trabaja a 3,3 V
Arduino Mega	12V_800 mA	39	16	15	16Mhz	Trabaja a 5 V
Arduino Mega ADK	12V_800 mA	39	16	15	16Mhz	Se puede trabajar directamente desde Android
Arduino MKR1000 WIFI	5V_300mA	8	6	4	48Mhz	Arduino con wifi incluido
Raspberry Pi3B+	5V_2.5A	26 pines GPIO, pines de entradas y salidas de usos múltiples			1,4Ghz	Bluetooth y wifi integrados

ODROID-XU4	5V_1.5A	13 pines GPIO	2Ghz	Wifi integrado
Jaguarboard	5V_2A	-	1.83Ghz	Soporte para Windows
Orange Pi	5V_1.5A	20 pines GPIO, pines de entradas y salidas de usos múltiples	1.6Ghz	Almacenamiento interno de 8Gb
Siemens s7-200	24V	8 entradas y salidas	CPU22X	Módulos de expansión, fiabilidad industrial
Festo FC34	24V	8 entradas y salidas	CPUFC34	Módulos de expansión, fiabilidad industrial
ABB AC500	24V	8 entradas y salidas	128kB de memoria de programa	Módulos de expansión, fiabilidad industrial

7.2 Arduino

La tarjeta Arduino Due es la primera placa Arduino basada en un microcontrolador de núcleo ARM de 32 bits. Con 54 pines de entrada / salida digital, 12 entradas analógicas, es la placa perfecta para proyectos Arduino de gran escala y gran alcance. A diferencia de la mayoría de las placas Arduino, la placa Arduino Due funciona a 3.3V. Este circuito, el controlador o driver de motores, tomará energía de otra fuente (variador o servo drive) y siguiendo las instrucciones de Arduino hará funcionar el motor.

7.2.1 Conexión de control Arduino

Las tarjetas de hardware y software abierto, trabajan con tecnología TTL. La tarjeta Arduino DUE al ser un board basado en un microcontrolador, necesita de 3 a 5 V para su funcionamiento, puede conectarse al ordenador o a una fuente externa a través de un regulador de tensión, para este caso, utilizaremos, fuentes de salida 5V DC a partir de una conexión monofásica de 220V. Como se puede ver en la siguiente figura.

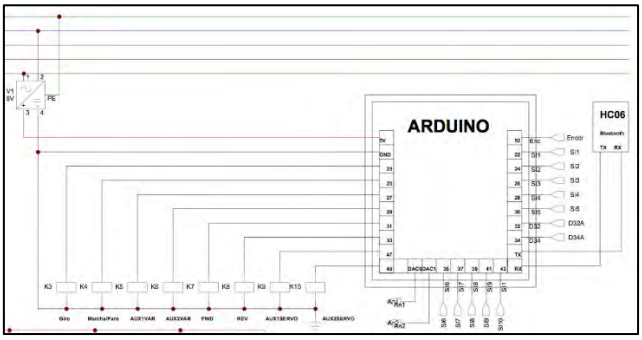


Figura 77 Esquema de conexión Arduino

La tarjeta Arduino está conectada a una regleta de relés, las bobinas funcionan con tecnología TTL, por lo que las salidas de los pines digitales de las tarjetas envían una señal a un transistor y un opto acoplador que brinda energía a las bobinas, energizándolas adecuadamente separando el circuito de proceso y control.

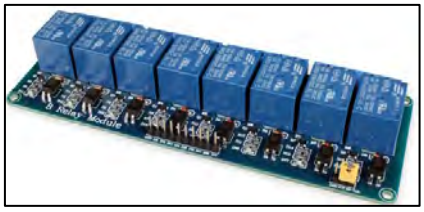


Figura 78 Tarjeta de 8 relés de 5V optoaislados a 10A

Sin embargo, las salidas de los sensores y de los finales de carrera funcionan a 24V por lo que es necesario implementar un divisor de tensión.

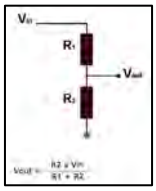


Figura 79 Esquema del divisor de tensión

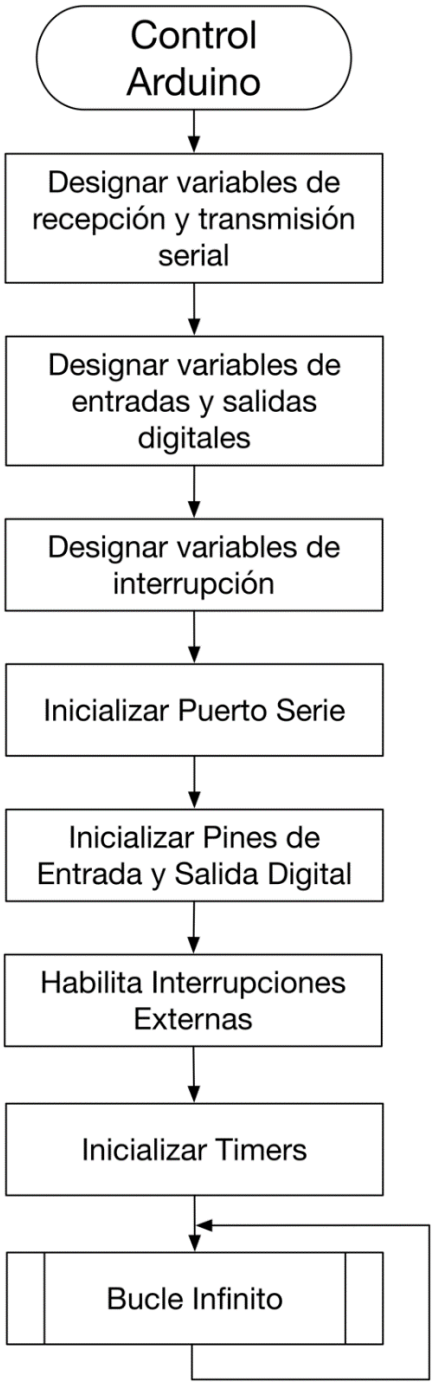
$$V_{out} = \frac{1500}{1500 + 10000} * 24 = 3.13V$$

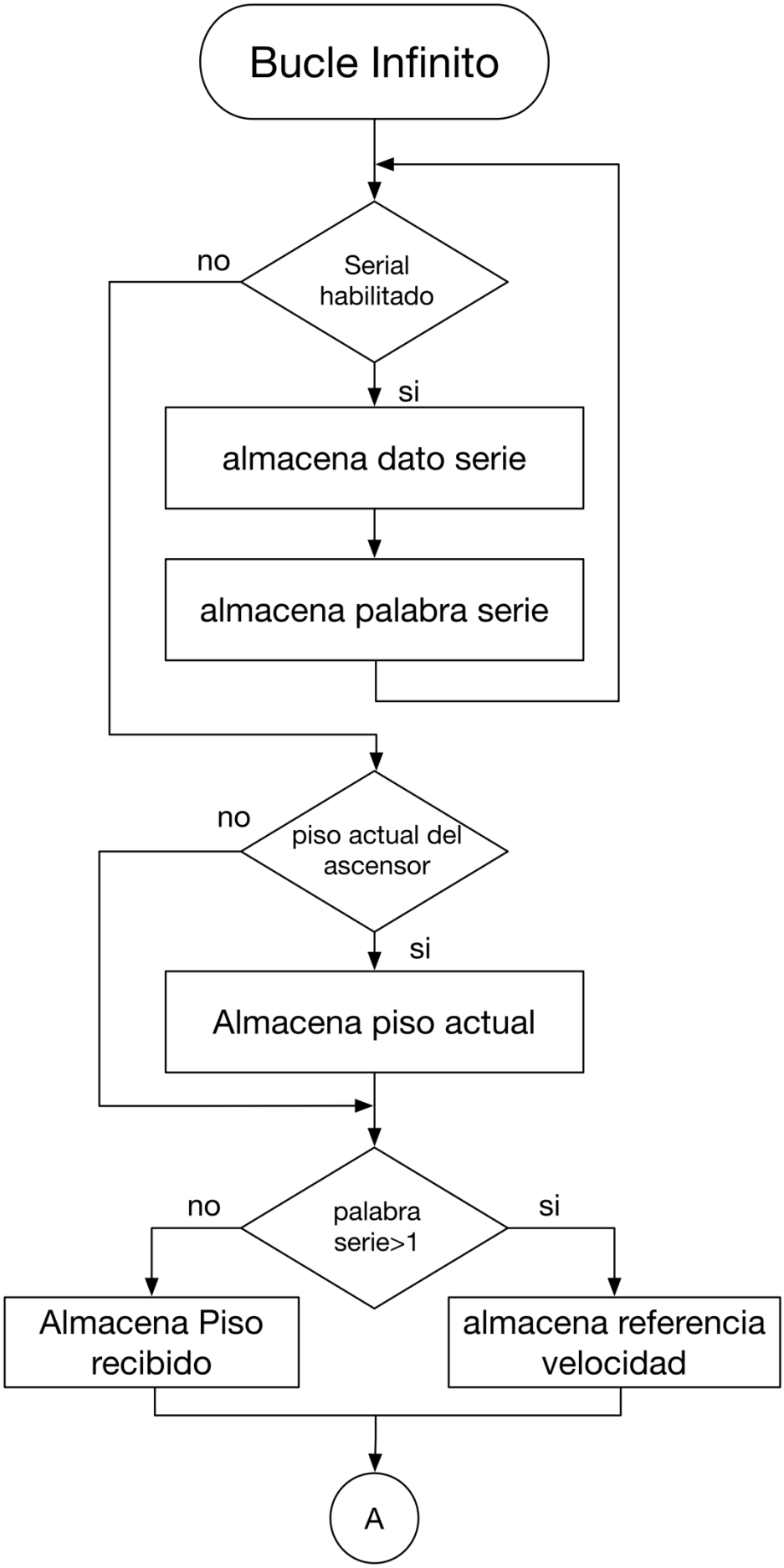
El envío de la información se realizará desde un celular, utilizando una señal de radiofrecuencia segura de 2.4Ghz, es la tecnología bluetooth la que brinda esta característica ya que el protocolo de comunicación permite enviar de forma inalámbrica datos con un rango de 10 metros aproximadamente de distancia. Se opta por este tipo de tecnología ya que el control de los elevadores debe ser uno global para ambas tarjetas, otra alternativa era el WI-FI sin embargo es necesario de un servidor para el control de forma remota, además el control se lo hará desde el laboratorio por lo que no será necesario más de un rango de distancia para el control siendo el bluetooth una buena opción. La tarjeta Arduino no dispone de este tipo de comunicaciones, por lo que se utiliza el módulo HC06 para comunicación, este módulo permite la recepción de datos a través de sus pines serial TX y RX por lo que estos van conectados como se ve en el esquema anterior.

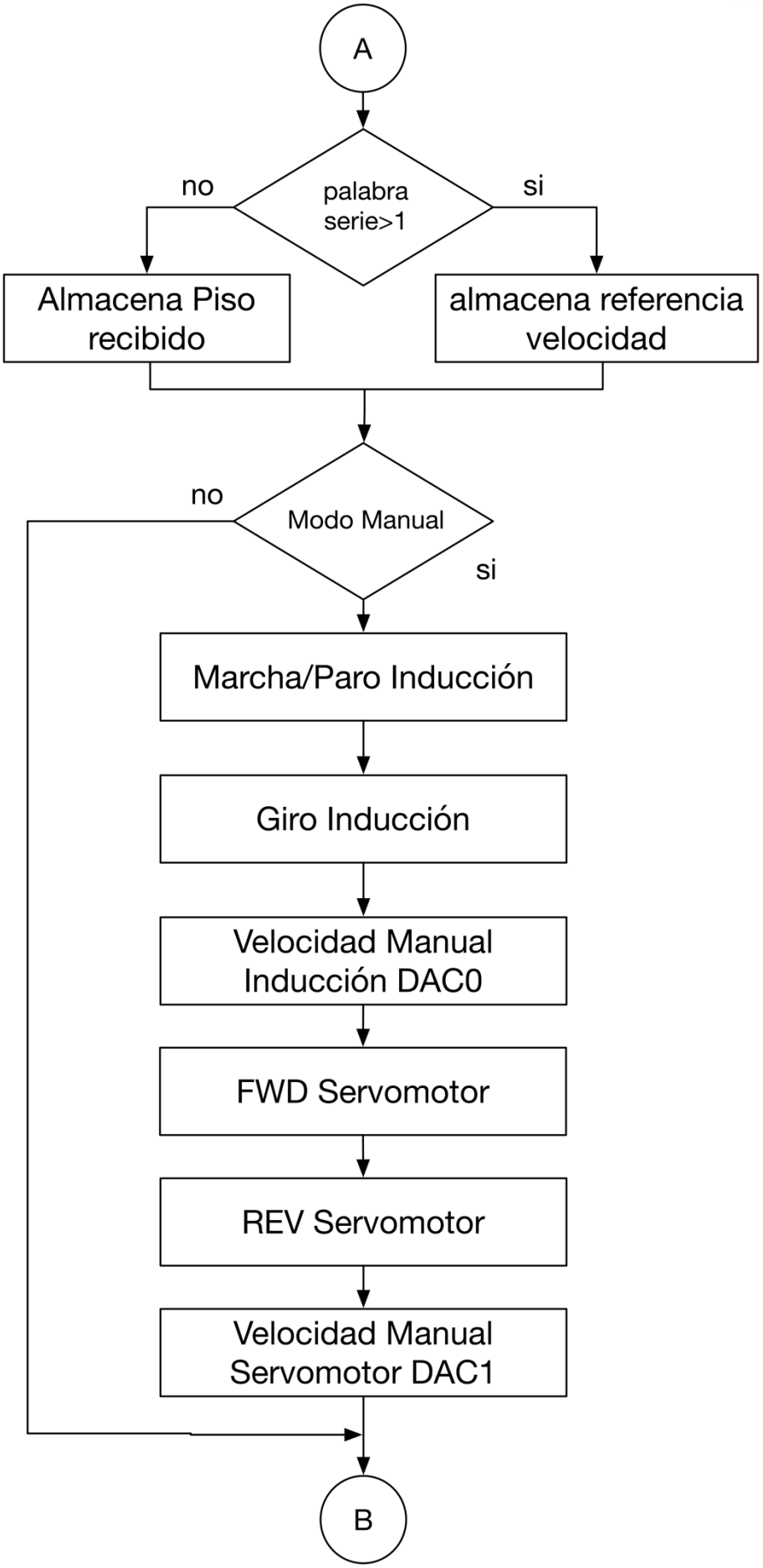
Figura 80 Modulo HC06, pines de funcionamiento

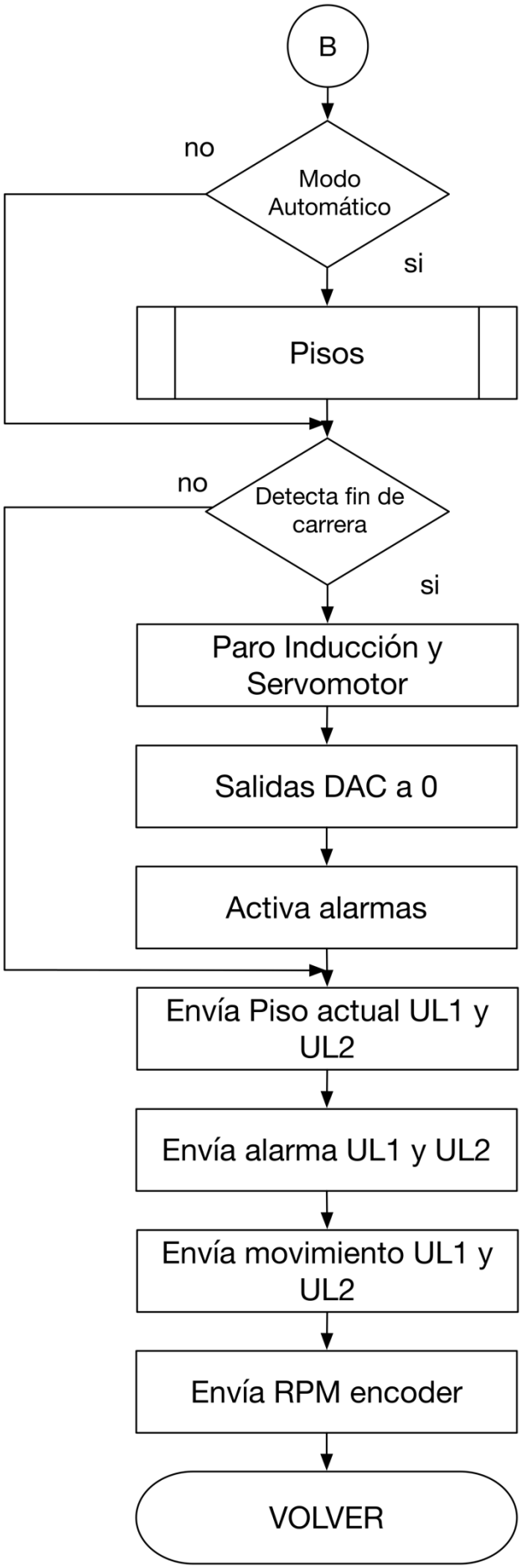
7.2.2 Control de velocidad Arduino

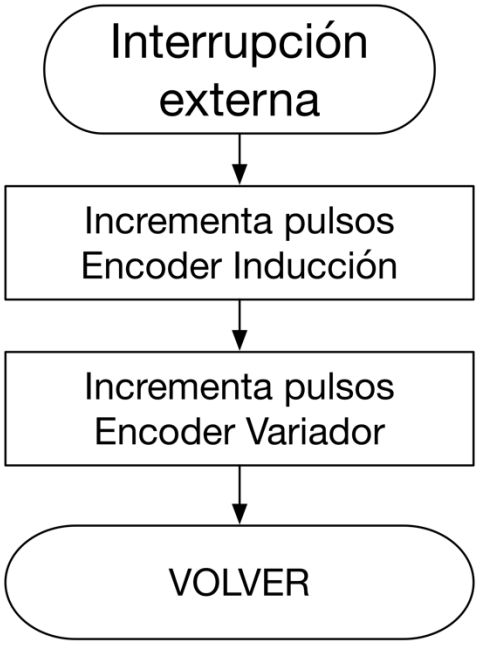
El control de velocidad en el arduino parte desde su arquitectura, es decir, al ser una tarjeta que ejecuta el programa al conectar tensión, inicializa su sistema con todos los valores en 0, a través de la aplicación móvil recibirá los datos de control y la tarjeta se encargará de mover los motores como se puede ver en el siguiente diagrama de flujo. La instalación del compilador de código arduino se encuentra en el **ANEXO V** y el código completo de programación se encuentra en el **ANEXO VII**.




Bucle Infinito
Serial habilitado
no
si
almacena dato serie
almacena palabra serie
piso actual del ascensor
no
si
Almacena piso actual
palabra serie>1
no
si
Almacena Piso recibido
almacena referencia velocidad
A


A
palabra serie>1
no
si
Almacena Piso recibido
almacena referencia velocidad
Modo Manual
no
si
Marcha/Paro Inducción
Giro Inducción
Velocidad Manual Inducción DAC0
FWD Servomotor
REV Servomotor
Velocidad Manual Servomotor DAC1
B


B
no
Modo Automático
si
Pisos
no
Detecta fin de carrera
si
Paro Inducción y Servomotor
Salidas DAC a 0
Activa alarmas
Envía Piso actual UL1 y UL2
Envía alarma UL1 y UL2
Envía movimiento UL1 y UL2
Envía RPM encoder
VOLVER



7.3 Raspberry

La tarjeta Raspberry Pi3B+ es un ordenador de placa simple y de bajo coste, este puede realizar tarjetas básicas de ofimática como documentos, navegar por internet e inclusive como dispositivo multimedia, a diferencia del Arduino esta tarjeta ya no es dedicada a una acción específica, esta lleva un sistema operativo de distribución libre, es diseñada como un ordenador ya que tiene más potencia de cálculo por lo que su inicio es un poco más lento.

7.3.1 Conexión de control Raspberry

Esta tarjeta posee pines de entrada y salida GPIO (General Purpose Input OutpuT), este es un sistema de entrada y salida de propósito general, es decir, consta de una serie de pines o conexiones que se pueden usar como entradas o salidas para múltiples usos con tecnología TTL, pero a diferencia del Arduino no posee pines de entrada/salida analógica, quiere decir que de ser necesario se necesita un conversor externo de señales de voltaje a datos digitales. Para el proyecto es necesario entradas, salidas digitales y salidas analógicas. La tarjeta posee 2 pines de salida PWM, la modulación por ancho de pulsos de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga. El variador de frecuencia y el servo drive son compatibles con este tipo de señal de control. De igual forma que en Arduino necesitamos

un regulador de 5VDC por lo que será necesario utilizar un regulador incluido en el kit complementario de compra, además a esto necesita un monitor, teclado y mouse para su uso como ya fue mencionado esta no es una tarjeta dedicada como un microcontrolador, este posee su propio sistema operativo. A continuación, se especifica el esquema de conexión de la Raspberry pi.

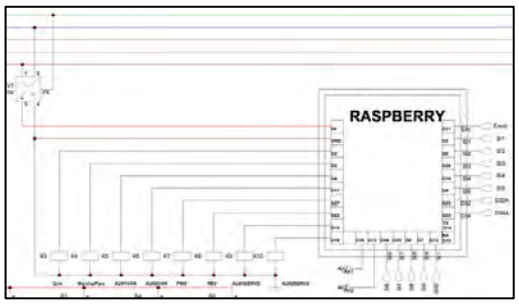


Figura 81 Esquema de conexión Raspberry

La tarjeta Raspberry está conectada a una regleta de relés, en primer lugar, alimentamos la electrónica del módulo VCC y GND a la señal de 5V de la Raspberry mediante los terminales existentes. Los pines de salida de los relés son terminales NO o NC, ocuparemos los terminales común y normalmente abierto NO

Las salidas de los sensores y de los finales de carrera funcionan a 24V por lo que es necesario implementar un divisor de tensión y configurar los pines GPIO. El dispositivo consta de bluetooth y wifi por lo que el uso de módulos no será necesario, sin embargo, necesita iniciar el sistema operativo y arrancar el programa de control a través de los periféricos como una computadora estándar.

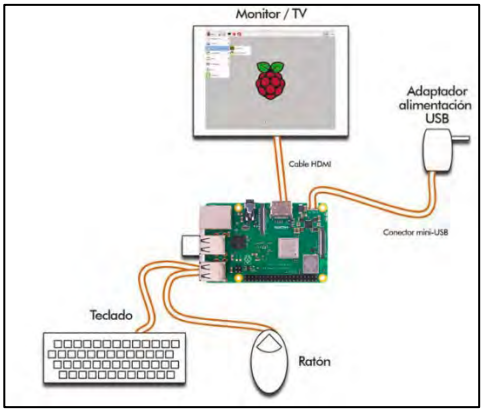
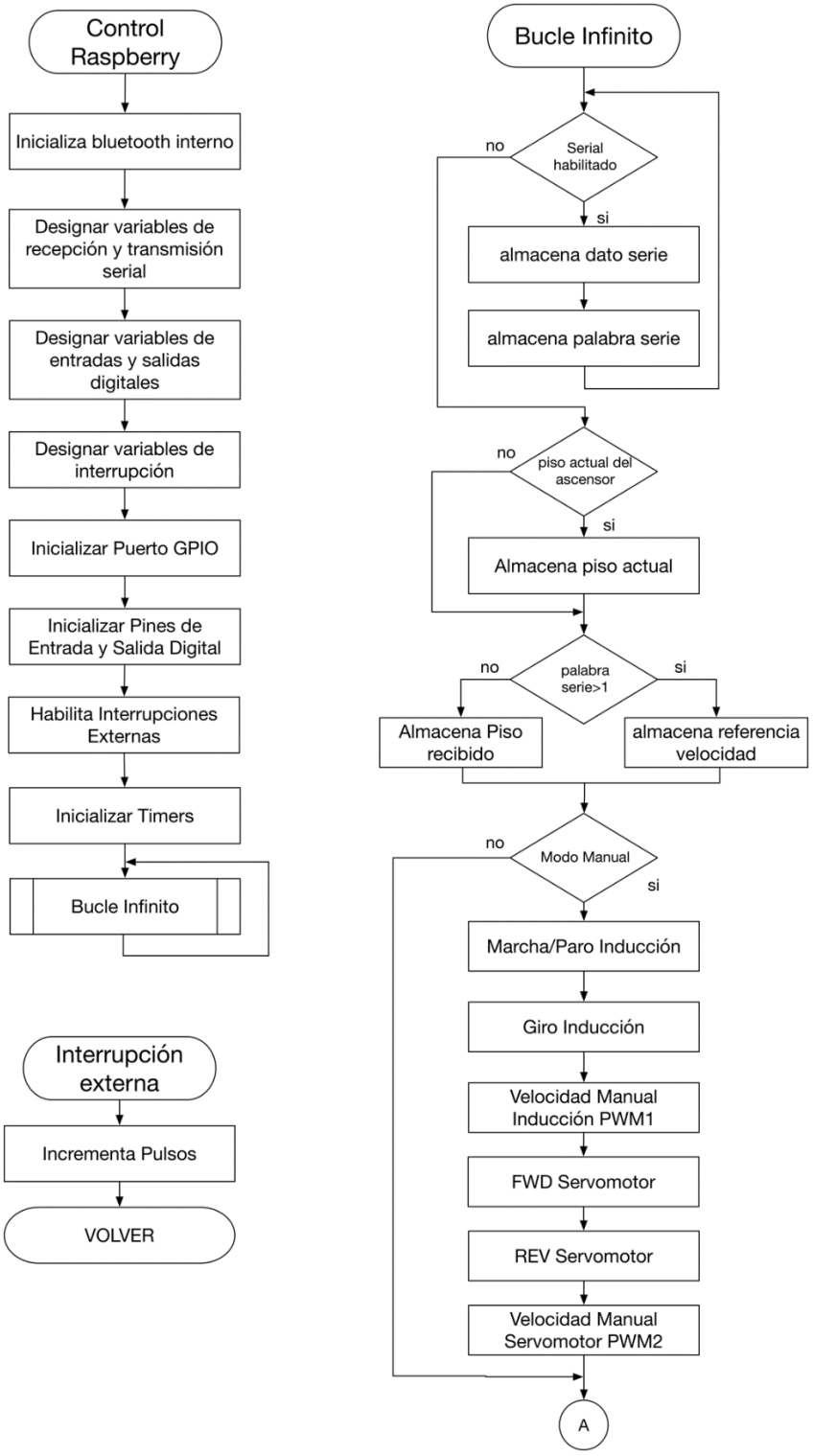


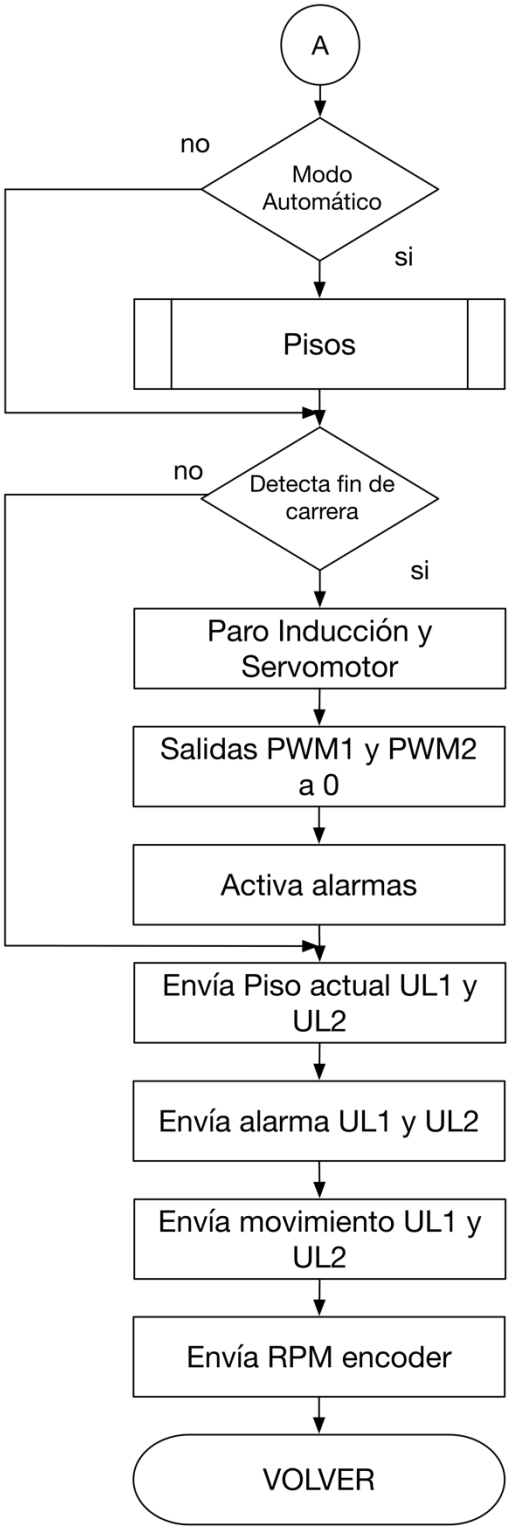
Figura 82 Raspberry Conexión periféricos

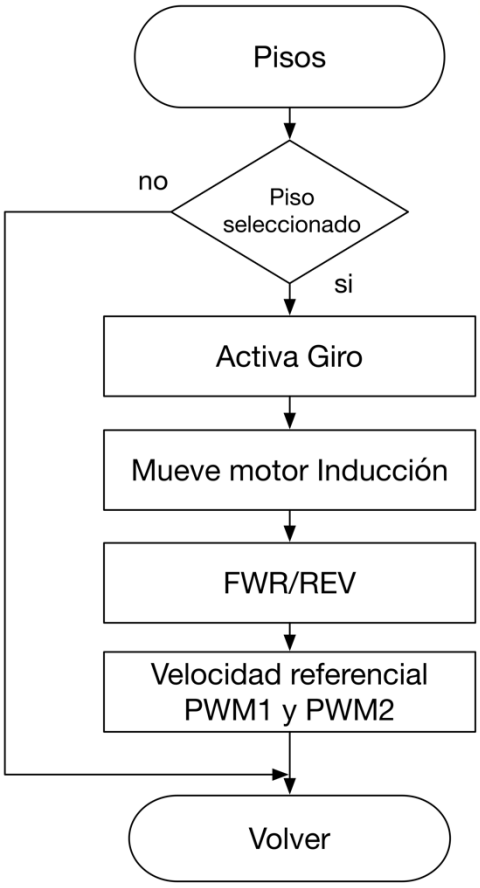
7.3.2 Control de velocidad Raspberry

El control de velocidad en la Raspberry parte desde su arquitectura, es decir, al ser una tarjeta que conlleva su propio sistema operativo, es necesario encender el dispositivo

y ejecutar el código a partir de comandos de programación, de igual forma los valores iniciales son 0, a través de la aplicación móvil recibirá los datos de control y la tarjeta se encargará de mover los motores como se puede ver en el siguiente diagrama de flujo. La instalación de raspbian, sistema operativo de la Raspberry se encuentra en el **ANEXO VI** y el código completo de programación se encuentra en el **ANEXO VIII**.




A
no
Modo Automático
si
Pisos
no
Detecta fin de carrera
si
Paro Inducción y Servomotor
Salidas PWM1 y PWM2 a 0
Activa alarmas
Envía Piso actual UL1 y UL2
Envía alarma UL1 y UL2
Envía movimiento UL1 y UL2
Envía RPM encoder
VOLVER


Pisos
no
Piso seleccionado
si
Activa Giro
Mueve motor Inducción
FWR/REV
Velocidad referencial PWM1 y PWM2
Volver

7.4 Autómata Programable

Los Autómatas programables de ABB proporcionan soluciones con un gran rendimiento y flexibilidad, siendo compatibles con una gran variedad de segmentos y aplicaciones incluyendo agua, infraestructuras, centros de datos, energías renovables, automatización de maquinaria, manipulación de materiales, marina y muchas más. En el caso del control del elevador, utilizaremos el autómata programable de ABB para controlar los variadores de frecuencia del motor de imanes permanentes y del motor de inducción. La escalabilidad del PLC AC500 ofrece una gran variedad de dispositivos para diseñar e implementar configuraciones adecuadas para tareas de control sencillas o soluciones de automatización complejas. Será necesario un módulo de expansión DC532 para obtener la señal de todas las entradas y controlar todas las salidas del autómata.

7.4.1 Conexión de control del Autómata programable

El autómata programable AC500 permite expansiones de dispositivos de campo, por ello ABB ha desarrollado módulos de E/S configurables por software de forma independiente y multifuncionales, que permiten adaptarse a diferentes requerimientos de automatización, en nuestro caso será controlar las unidades lineales a partir de las entradas digitales de los sensores. La cabecera de expansión DC532 tiene 16 DI, por lo que será útil para el proyecto, en la siguiente figura se puede apreciar la conexión del PLC con su expansión.

Figura 83 Autómata programable ABB AC500 con expansión DC532

Es necesario configurar el autómata programable por lo que en CoDeSys (software de programación de ABB), se configura la comunicación entre los dispositivos, designando las variables de entrada y salida del autómata, además se toma en cuenta las

salidas analógicas y las entradas de los encoder, finalmente se implementa el código en lenguaje ST y la interfaz de usuario.

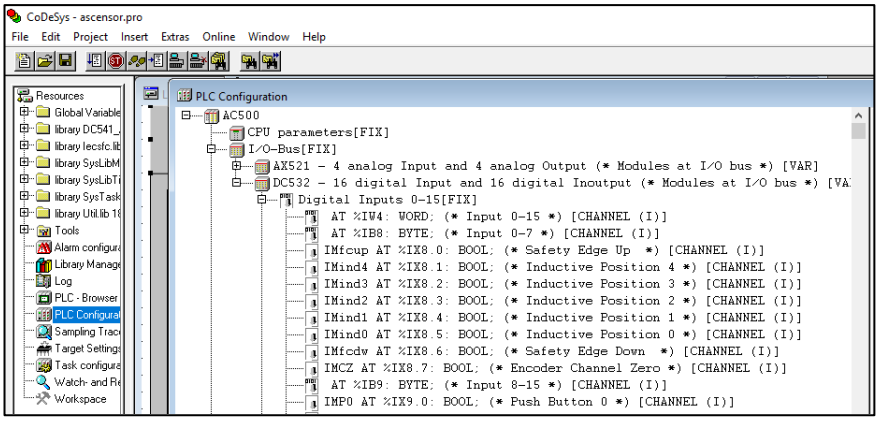
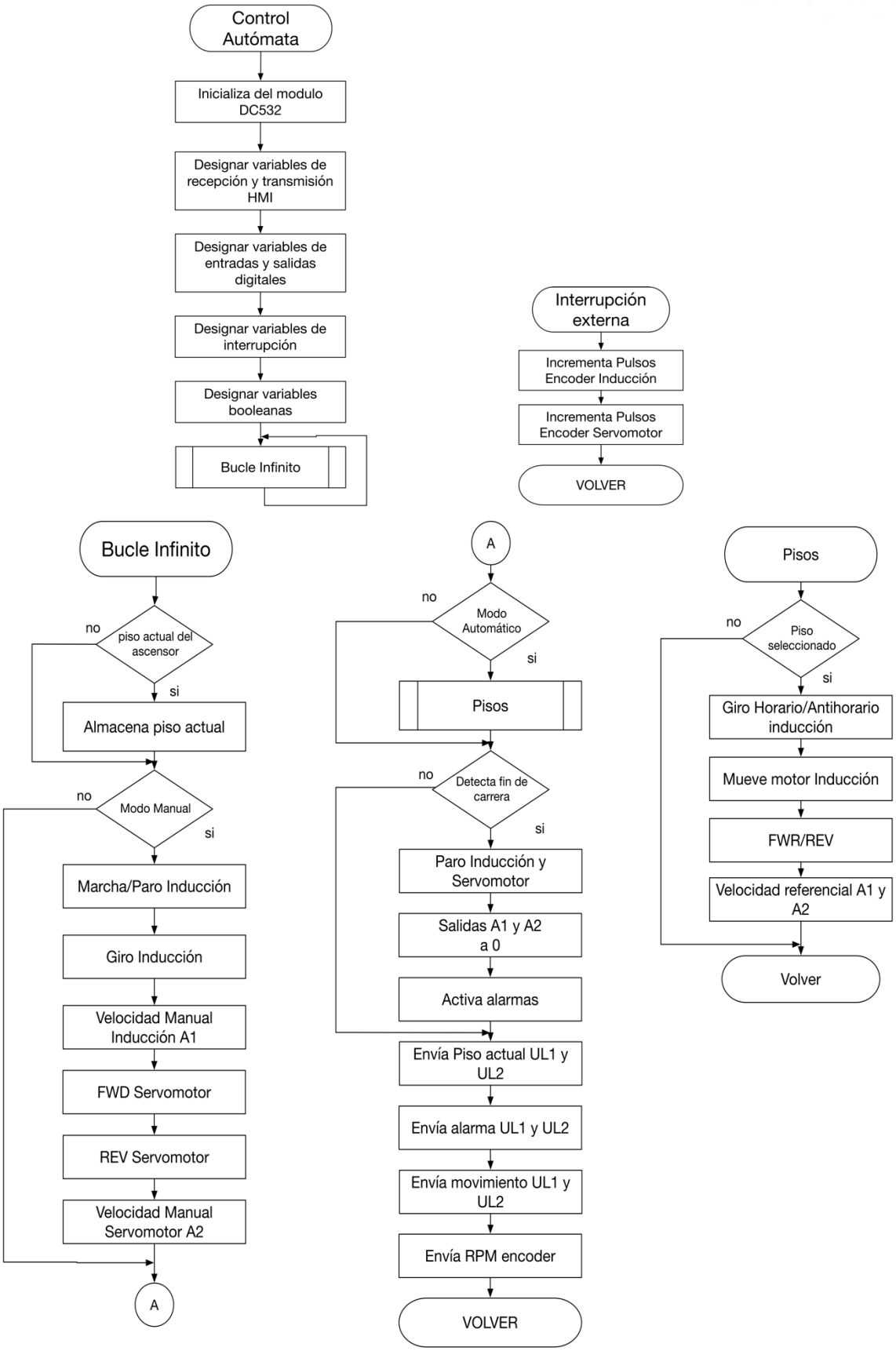


Figura 84 Configuración del autómata programable en CoDeSys

7.4.2 Control de velocidad en el Autómata Programable

En el control industrial se utiliza la norma estándar internacional IEC 61131-3 que define los lenguajes de programación de los Controladores Lógicos Programables (PLC). Uno de ellos es el Texto Estructurado (Structured Text - ST) que es un lenguaje textual de alto nivel similar a PASCAL. Un programa en ST se compone de declaraciones. Hay dos tipos de declaraciones: de asignación, para asignar un valor a una variable y de control, como sentencias de selección y sentencias de repetición. Las declaraciones se componen de los siguientes elementos: Variables, Constantes, Operadores y Funciones. (Fernández, 2019)

En el caso del autómata programable, el código este compuesto por llamadas que se realizan desde el HMI a las variables del lenguaje en texto estructurado, por tal motivo el diagrama, será una combinación entre el código fuente y la interfaz HMI, el código completo de programación se encuentra en el **ANEXO XI**.


Control Autómata
Inicializa del modulo DC532
Designar variables de recepción y transmisión HMI
Designar variables de entradas y salidas digitales
Designar variables de interrupción
Designar variables booleanas
Bucle Infinito
Interrupción externa
Incrementa Pulsos Encoder Inducción
Incrementa Pulsos Encoder Servomotor
VOLVER
Bucle Infinito
no
piso actual del ascensor
si
Almacena piso actual
no
Modo Manual
si
Marcha/Paro Inducción
Giro Inducción
Velocidad Manual Inducción A1
FWD Servomotor
REV Servomotor
Velocidad Manual Servomotor A2
A
A
no
Modo Automático
si
Pisos
no
Detecta fin de carrera
si
Paro Inducción y Servomotor
Salidas A1 y A2 a 0
Activa alarmas
Envía Piso actual UL1 y UL2
Envía alarma UL1 y UL2
Envía movimiento UL1 y UL2
Envía RPM encoder
VOLVER
Pisos
no
Piso seleccionado
si
Giro Horario/Antihorario inducción
Mueve motor Inducción
FWR/REV
Velocidad referencial A1 y A2
Volver

8. Diseño de la visualización y Monitorización

8.1 Visualización y monitorización local

El programa es diseñado en CoDeSys, la aplicación está programada para ser amigable con el usuario, pero a su vez posee seguridades de configuración, debido a que el proyecto tiene modos de funcionamiento de mantenimiento en donde se requiere un grado de conocimiento más amplio sobre las unidades lineales, mientras que el modo Ascensor es una interfaz de control para el elevador. En la pantalla principal se accede a la aplicación a través de un usuario y una contraseña. Por defecto el usuario es **admin** y la contraseña **admin**.

Figura 85 Ventana de seguridad en CoDeSys

Una vez ingresado, se despliega el menú de modos de funcionamiento, configuración y alarmas, en esta venta se puede seleccionar una configuración manual o los valores de velocidad para el modo ascensor y puesta en marcha.

Figura 86 Menú de control local

La ventana de configuración del ascensor, permiten al usuario elegir la cantidad de pisos, la velocidad de funcionamiento del motor a inducción y del servomotor, mientas que la ventana de mantenimiento permite controlar de forma manual las unidades lineales por separado, verificando su puesta en marcha, sentido de giro, tiempo de respuesta y señales de los sensores.

Figura 87 Ventanas de mantenimiento y configuración

Una vez en la ventana de configuración se presione el botón ON, el sistema comenzará a funcionar con los valores cargados de la pantalla de configuración, al dirigirse a la venta Ascensor, se puede seleccionar el piso hacia el cual se dirigirá el ascensor, además en la pantalla negra se indica la posición actual del piso y en el lado derecho se puede ver los pilotos de funcionamiento tanto de los sensores como de los accionamientos. En la siguiente imagen se puede visualizar el programa en funcionamiento.

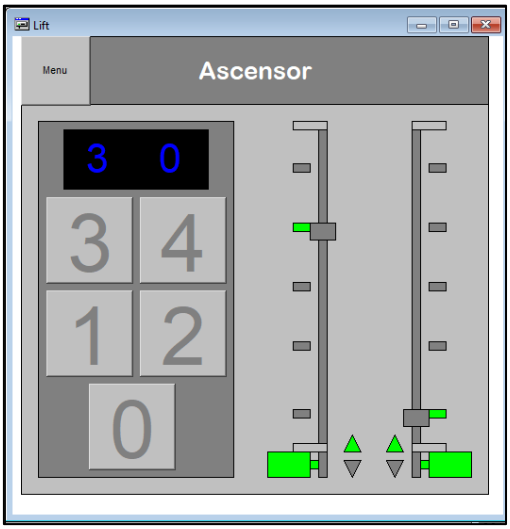


Figura 88 Ventana del modo Ascensor en funcionamiento

8.2 Visualización y monitorización remota

Para el diseño de la aplicación se tomó en consideración el índice de uso del móvil en España, a finales de 2015, 9 de cada 10 individuos tiene un móvil. A partir de 2013, el teléfono móvil se posiciona como el dispositivo líder para acceder a Internet. Junto a la tableta son los dos dispositivos que han experimentado un ascenso más notable en los últimos años. Un 81% de los españoles utilizan smartphones y se descargan 3.8 millones de apps al día. 27,7 millones de españoles usan las apps a diario. La franja de edad más adicta a las apps va de los 25 a los 34 años.

Por tal motivo la tendencia del uso de aplicaciones móviles y el avance tecnológico van a la par, en este trabajo fin de máster se opta por la implementación de una aplicación móvil que sea capaz de controlar los elevadores, utilizando un dispositivo Android.

Android es el sistema operativo móvil más utilizado en el mundo, posee un amplio mercado en plataformas de compilación, por lo cual se analizan distintos IDE (Entorno de Desarrollo Integrado) para realizar la aplicación entre los cuales resalta:

Tabla 10 IDE de compilación Android

IDE	Características	Nivel de Programación
Android Studio	IDE oficial creado por Google	Experto
Eclipse	Similar a Android Studio	Experto
Xamarin	Plataforma Android e iOS	Medio/Experto
AIDE	Programar desde Android	Medio
M4A	Simple y rápido	Básico
App Inventor	Programación Intuitiva	Básico

La aplicación será destinada a las Tablet marca Lenovo, que se encuentran en el laboratorio, además se instalarán en otros dispositivos móviles para comprobar su adaptabilidad, por tal motivo se toma el software de Android Studio, ya que al ser el software oficial de Google ofrece actualizaciones constantes, solución de errores y una mayor gama de compatibilidad entre todas las versiones del software.

8.2.1 Configuración inicial Android Studio

Android Studio es un entorno de desarrollo integrado para el sistema operativo Android lanzado por Google, diseñado para ofrecer nuevas herramientas para el desarrollo de aplicaciones y alternativa al entorno Eclipse, hasta ahora el IDE más utilizado del mercado.

Al crear un nuevo proyecto en Android Studio, la estructura del proyecto aparece con casi todos los archivos dentro del directorio SRC, un cambio a un sistema de generación basado Gradle que proporciona una mayor flexibilidad para el proceso de construcción. Además, gracias a su sistema de emulación integrado, Android Studio permite ver los cambios que realizamos en nuestra aplicación en tiempo real, pudiendo además comprobar cómo se visualiza en diferentes dispositivos Android con distintas configuraciones y resoluciones de forma simultánea.

8.2.2 Interfaz gráfica remota en Android Studio

La aplicación está programada desde 0 en dos partes, una es la encargada de la interfaz de usuario, inicialmente se debe emparejar el dispositivo bluetooth Arduino y Raspberry con el dispositivo móvil, una vez se ingresa a la app, en la pantalla principal se accede a la aplicación a través de un usuario y una contraseña. Por defecto el usuario es **admin** y la contraseña **admin**. Una vez es aprobado el usuario, se despliega una lista de dispositivos bluetooth, en este apartado se puede seleccionar entre los dos tipos de control HC06 Arduino y Rasberry.

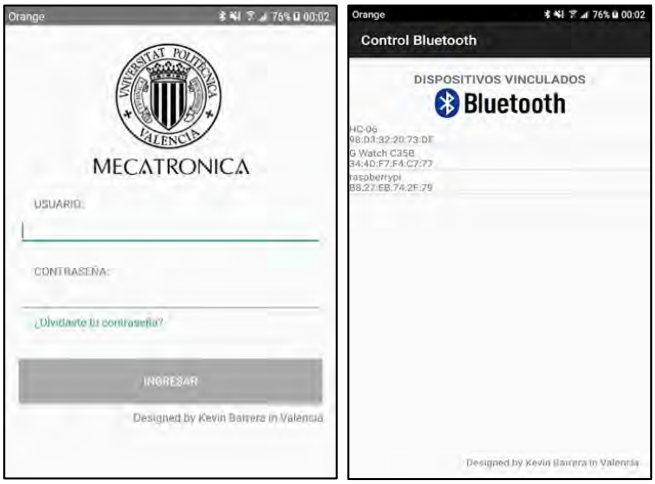


Figura 89 Ventana de seguridad en la aplicación y Dispositivos bluetooth de control

Por defecto los valores iniciales y el estado de los motores es apagado, en el lado superior derecho existe un menú emergente en el cual se puede seleccionar los modos de operación, mantenimiento, historial, alarmas configuración del ascensor.

Figura 90 Menú de selección en la aplicación

En el apartado de mantenimiento se puede acceder al control de marcha/ paro, giro y velocidad del motor de inducción y en el motor de imanes permanentes se puede activar FWD, REV y además controlar su velocidad. En la ventana de Alarmas, se visualiza cuando cualquiera de los elevadores llega al final de carrera+ o al final de carrera-, en la ventana de historial se visualizará todos los botones que se han pulsado y los valores que se han enviado y recibido, finalmente en la ventana de configuración, se puede designar los valores de velocidad tanto del motor a inducción como el servomotor para el control del elevador por pisos, además se puede seleccionar el número de piso y desconectarse o salirse del sistema.

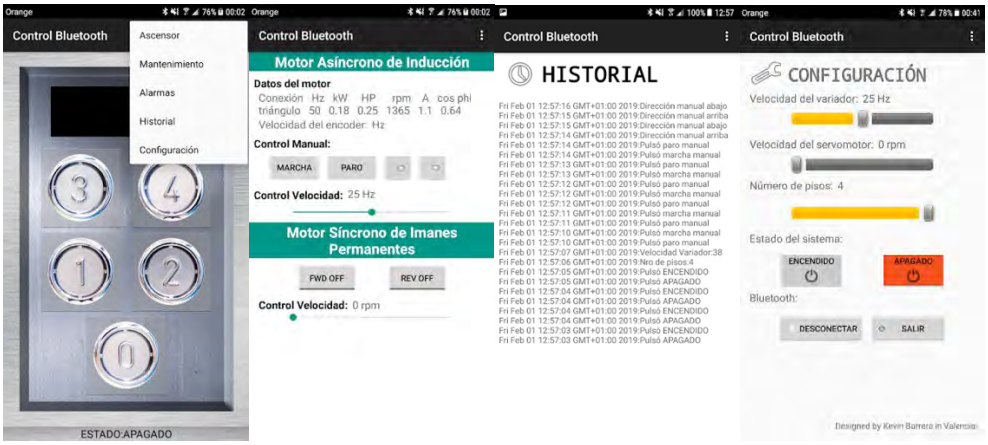


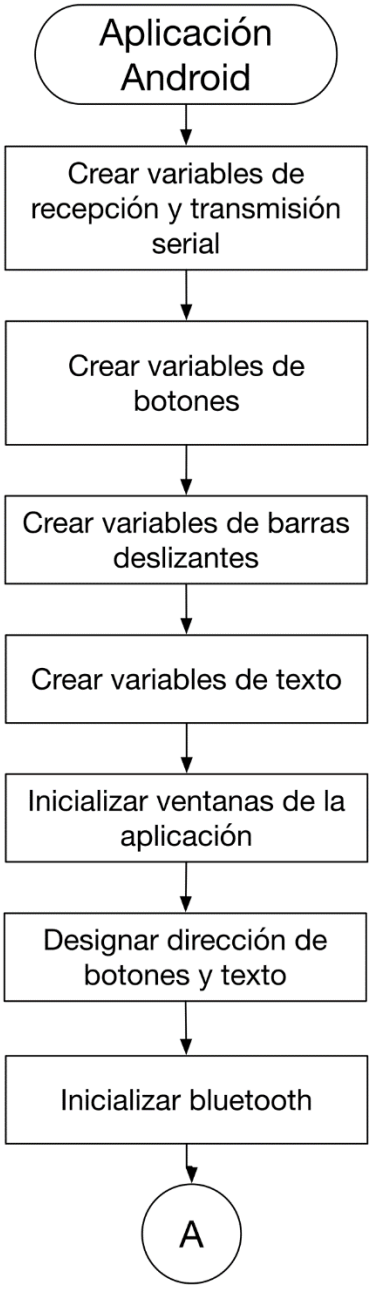
Figura 91 Ventanas de la aplicación móvil

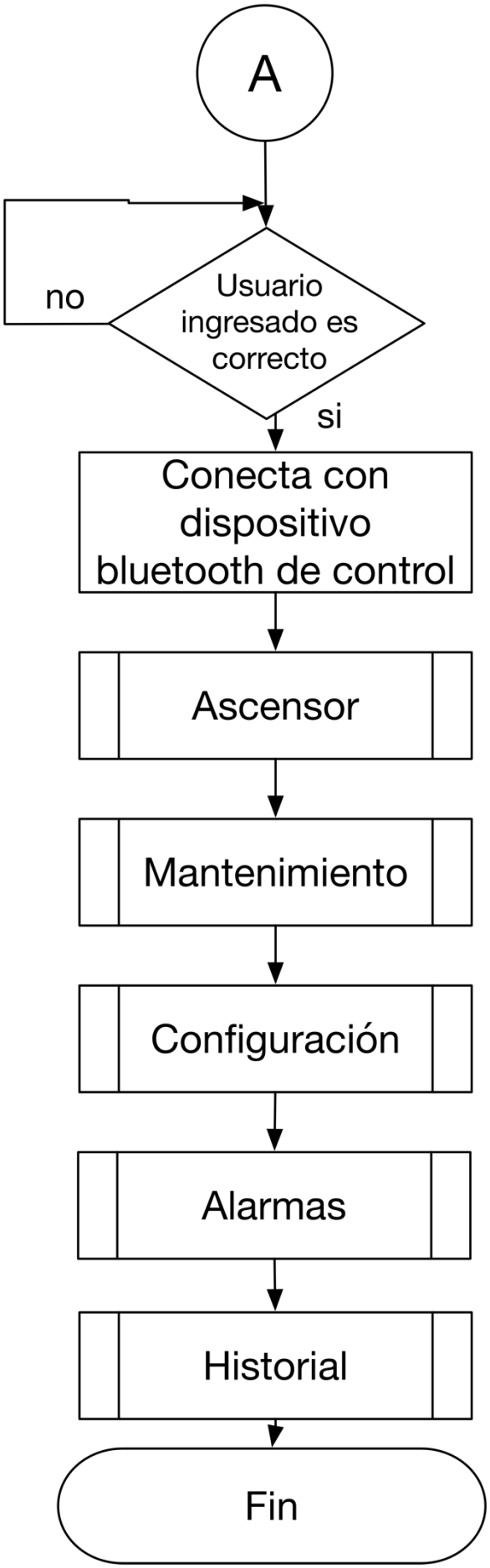
Una vez en la ventana de configuración se presione el botón de ENCENDIDO, el sistema comenzará a funcionar con los valores cargados de la pantalla de configuración, al dirigirse a la venta Ascensor, se puede seleccionar el piso hacia el cual se dirigirá el ascensor, además en la pantalla negra se indica la dirección del desplazamiento y la posición actual del piso. En la siguiente imagen se puede visualizar la aplicación en funcionamiento.

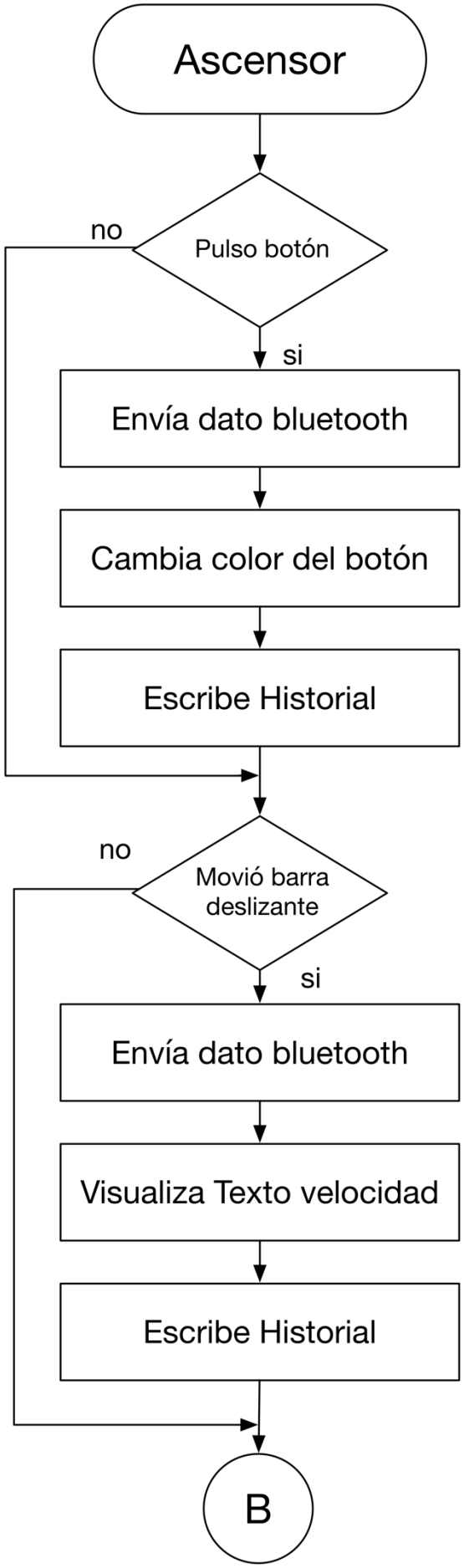
Figura 92 Aplicación de control funcional

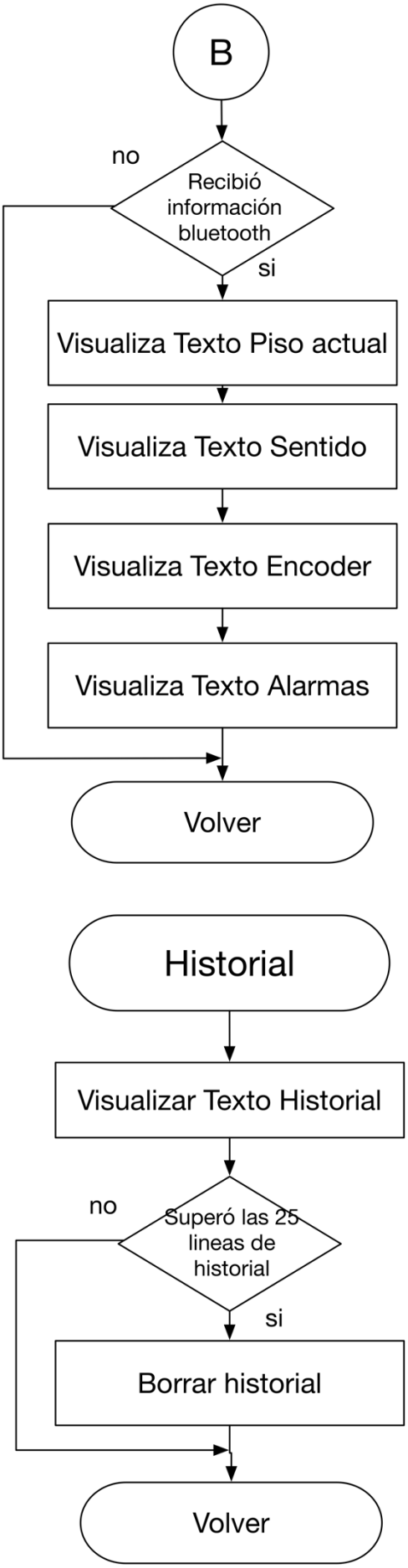
Programación de la aplicación móvil

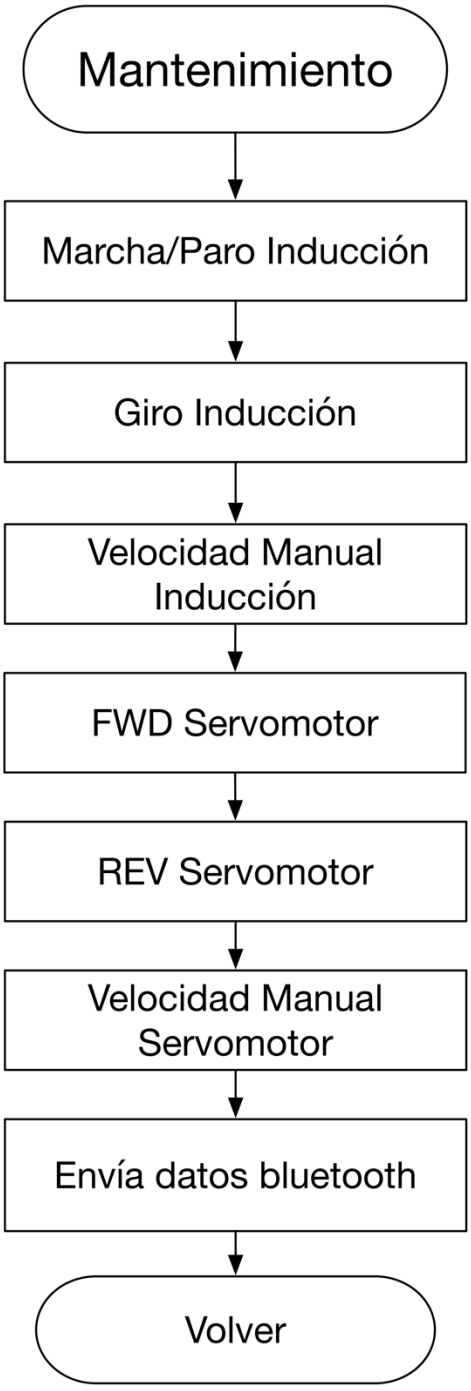
La aplicación está programada en base a lenguaje Java, para la descripción del algoritmo informático se implementa siguiente diagrama de flujo en el cual se puede apreciar el funcionamiento de la aplicación. Cabe recalcar que el código completo se encuentra en el **ANEXO IX.**


Aplicación Android
Crear variables de recepción y transmisión serial
Crear variables de botones
Crear variables de barras deslizantes
Crear variables de texto
Inicializar ventanas de la aplicación
Designar dirección de botones y texto
Inicializar bluetooth
A


A
Usuario ingresado es correcto
no
si
Conecta con dispositivo bluetooth de control
Ascensor
Mantenimiento
Configuración
Alarmas
Historial
Fin


Ascensor
Pulso botón
no
si
Envía dato bluetooth
Cambia color del botón
Escribe Historial
Movió barra deslizante
no
si
Envía dato bluetooth
Visualiza Texto velocidad
Escribe Historial
B


B
no
Recibió información bluetooth
si
Visualiza Texto Piso actual
Visualiza Texto Sentido
Visualiza Texto Encoder
Visualiza Texto Alarmas
Volver
Historial
Visualizar Texto Historial
no
Superó las 25 lineas de historial
si
Borrar historial
Volver


Mantenimiento
Marcha/Paro Inducción
Giro Inducción
Velocidad Manual Inducción
FWD Servomotor
REV Servomotor
Velocidad Manual Servomotor
Envía datos bluetooth
Volver

9. Osciloscopio

Un osciloscopio es un instrumento de medición para la electrónica. Representa una gráfica de amplitud en el eje vertical y tiempo en el eje horizontal. Es muy usado por estudiantes, diseñadores, ingenieros en el campo de la electrónica. Frecuentemente se complementa con un multímetro, una fuente de alimentación. Últimamente, con la explosión de dispositivos con tecnologías de radio frecuencia como Wifi o Bluetooth, el banco de trabajo se complementa con un analizador de espectro.

El osciloscopio presenta los valores de las señales eléctricas en forma de coordenadas en una pantalla, en la que normalmente el eje X (horizontal) representa tiempos y el eje Y (vertical) representa tensiones. La imagen así obtenida se denomina oscilograma.

9.1 Selección de la tarjeta

9.1.1 Osciloscopio Digilent Discovery2

Digilent Analog Discovery 2 es un osciloscopio USB e instrumento multifunción con un precio que ronda los 250 euros, permite a los usuarios medir, visualizar, generar, registrar y controlar circuitos de señal mixta de todo tipo. Desarrollado en colaboración con Analog Devices y soportado por el programa universitario de Xilinx, Analog Discovery 2 trabaja a 30Mhz, es lo suficientemente pequeño para caberle en el bolsillo, pero a la vez lo suficientemente potente para sustituir a una variedad de equipos de laboratorio, proporcionando a los estudiantes de ingeniería, aficionados y entusiastas de la electrónica la libertad de trabajar con circuitos analógicos y digitales en prácticamente cualquier entorno dentro o fuera del laboratorio. Las entradas y salidas analógicas y digitales se pueden conectar a un circuito con sondas de hilo simple. Alternativamente, se puede usar el adaptador BNC de Analog Discovery y sondas BNC para conectar y utilizar las entradas y salidas. Controlado por el software gratuito WaveForms (compatible con Mac, Linux). (Farnell, 2018)

Figura 93 Osciloscopio Digilent Discovery 2

9.1.2 Picoscope 2204A

El osciloscopio de la serie PicoScope 2200 ofrece una alternativa pequeña, ligera y moderna a los voluminosos dispositivos de banco con una resolución de 8 bits y 10Mhz de procesamiento. Es perfecto para ingenieros que han de trasladarse constantemente e ideal para un amplio abanico de aplicaciones que incluyen diseño, pruebas, educación, servicios, supervisión, detección y reparación de fallos, usos cotidianos mas no industriales. (PicoTech, 2018)

Figura 94 Osciloscopio Picoscope

9.1.3 Lolin Wemos D32 I2C PRO

Wemos es popular por su placa Wifi de bajo costo basada en SoEs de Espressif. Su tablero más popular es el Wemos D1 mini basado en ESP8266 gracias a su factor de forma compacta, bajo precio y tableros adicionales disponibles. LOLIN D32 Pro v2.00, cuenta con el módulo ESP32-WROVER, y además de exponer las E / S a través de cabeceras compatibles, también ofrece una ranura para tarjeta micro SD, un conector de pantalla TFT, una cabecera I2C y un procesador de hasta 240Mhz de doble núcleo.

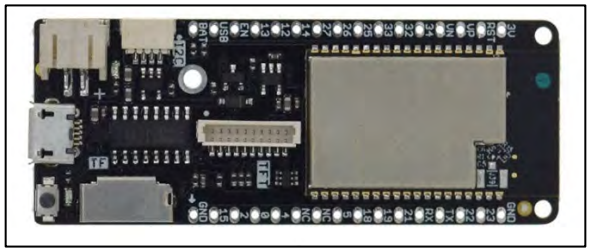


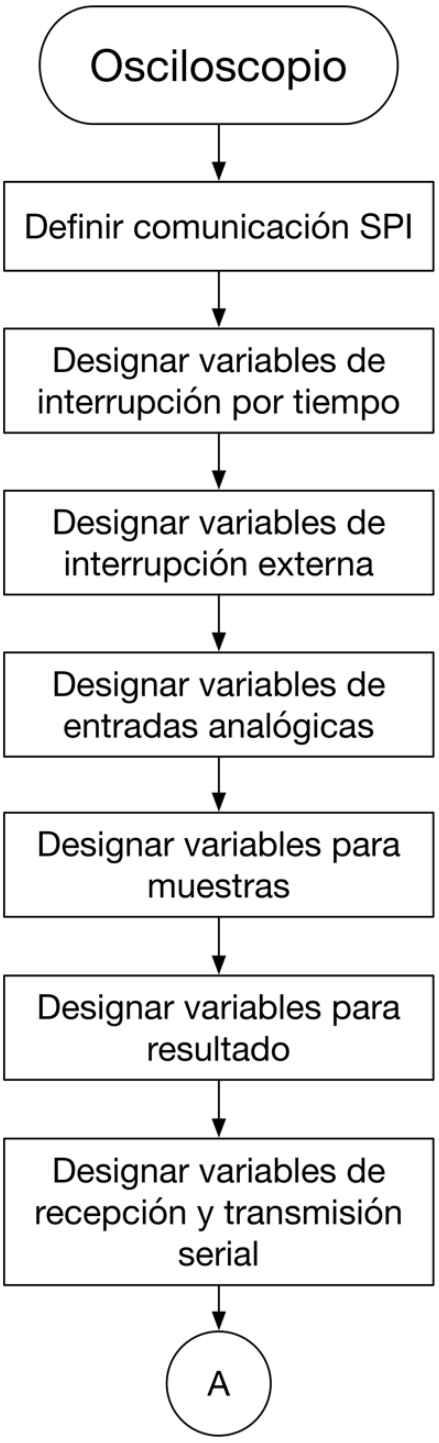
Figura 95 Lolin Wemos D32 I2C PRO

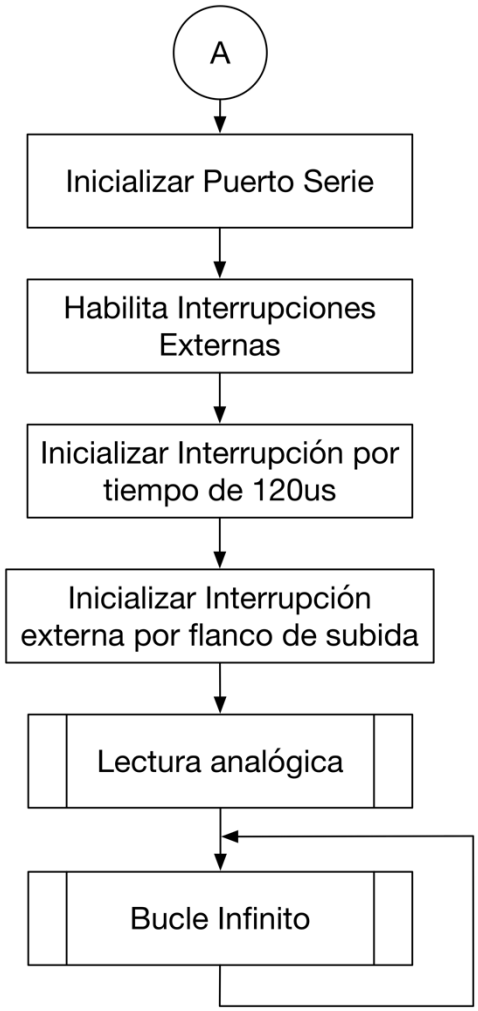
Se designa para la creación del osciloscopio la tarjeta Lolin D32 por las prestaciones de procesamiento que posee y además su compatibilidad con módulos bluetooth y programación en Arduino.

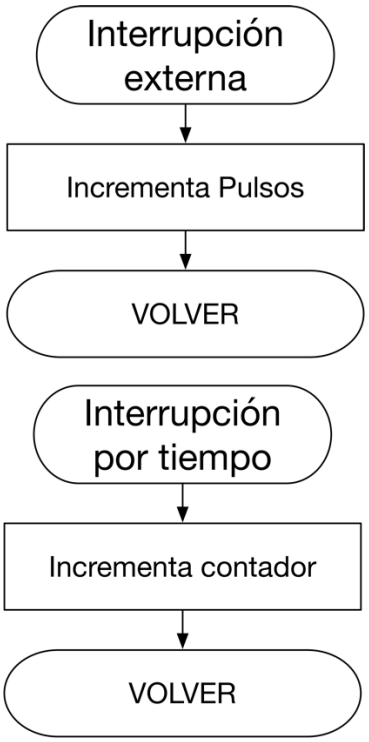
9.2 Osciloscopio con tarjeta Lolin I2C D32 PRO

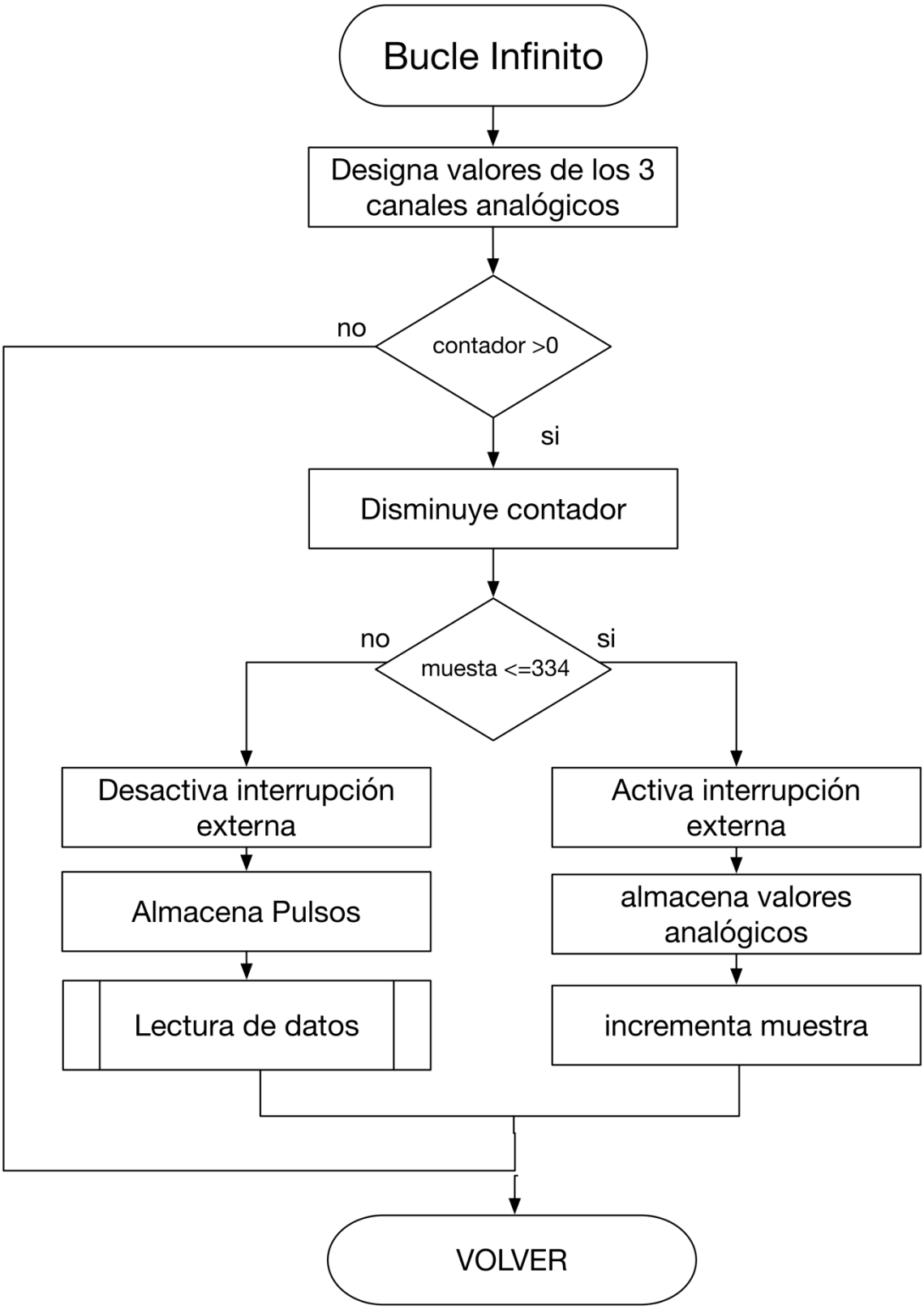
Es necesario tomar consideraciones al momento de programar, como utilizamos un conversor que toma muestras cada 40us por canal, implementaremos

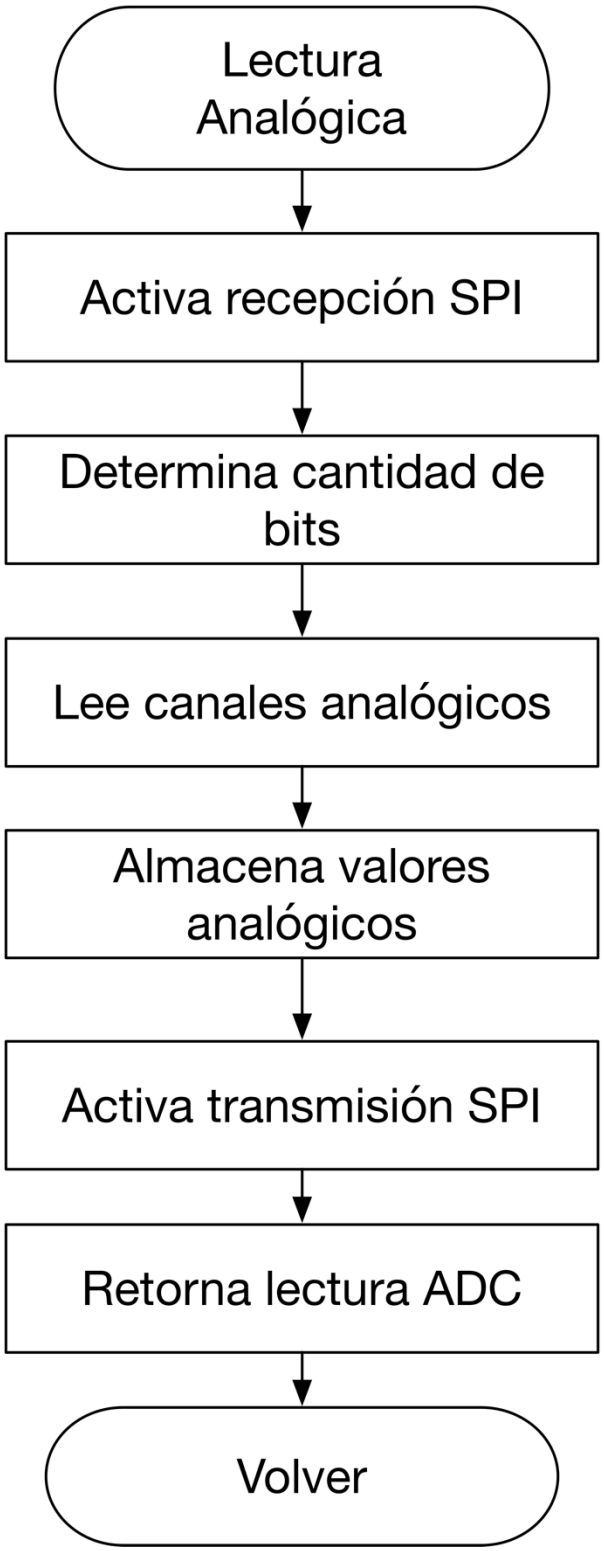
interrupciones temporizadas, para que tome cada 120us, debido a que ocuparemos 3 canales a la vez, además se ocuparan interrupciones externas para el control del encoder, en donde la cantidad de pulsos será dada por un encoder con una resolución de 10 bits. A continuación, se indica el diagrama de flujo correspondiente al código del osciloscopio. Cabe recalcar que el código completo se encuentra en el **ANEXO X.**

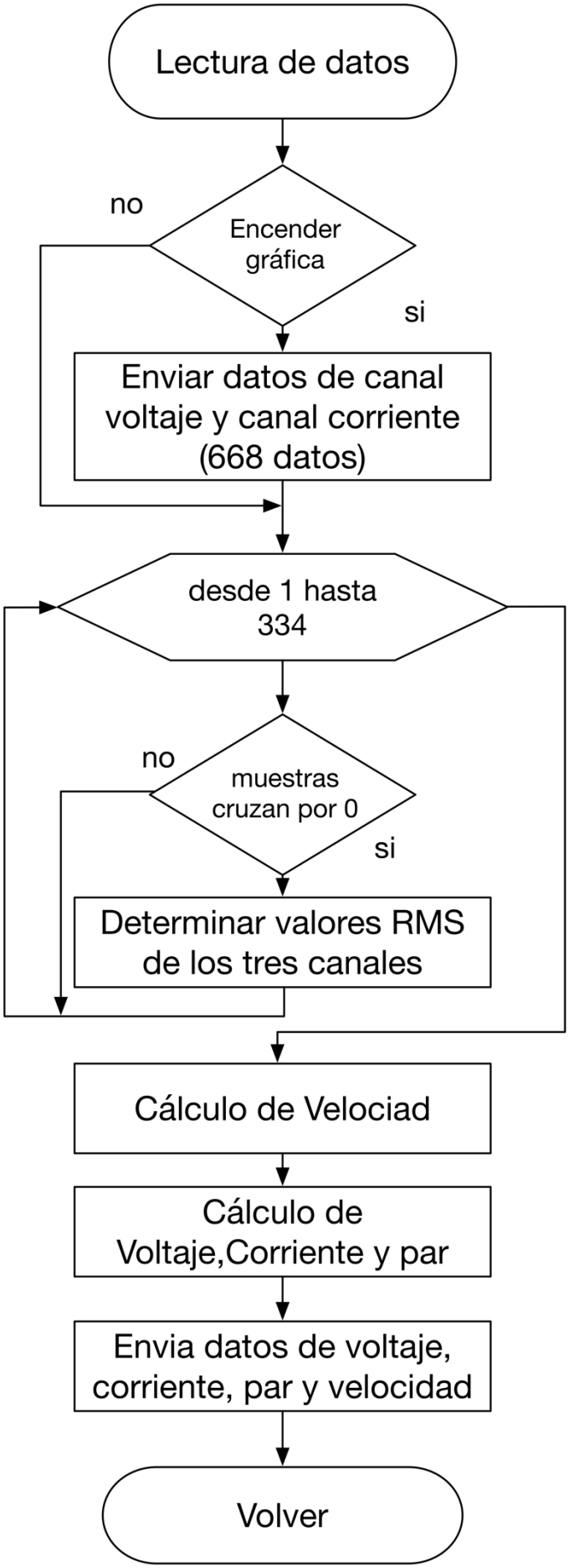



A
Inicializar Puerto Serie
Habilita Interrupciones Externas
Inicializar Interrupción por tiempo de 120us
Inicializar Interrupción externa por flanco de subida
Lectura analógica
Bucle Infinito


Interrupción externa
Incrementa Pulsos
VOLVER
Interrupción por tiempo
Incrementa contador
VOLVER


Bucle Infinito
Designa valores de los 3 canales analógicos
no
contador >0
si
Disminuye contador
no
muesta <=334
si
Desactiva interrupción externa
Almacena Pulsos
Lectura de datos
Activa interrupción externa
almacena valores analógicos
incrementa muestra
VOLVER


Lectura Analógica
Activa recepción SPI
Determina cantidad de bits
Lee canales analógicos
Almacena valores analógicos
Activa transmisión SPI
Retorna lectura ADC
Volver


Lectura de datos
no
Encender gráfica
si
Enviar datos de canal voltaje y canal corriente (668 datos)
desde 1 hasta 334
no
muestras cruzan por 0
si
Determinar valores RMS de los tres canales
Cálculo de Velociad
Cálculo de Voltaje,Corriente y par
Envia datos de voltaje, corriente, par y velocidad
Volver

10. Montaje, Puesta en Marcha y pruebas

10.1 Montaje

Una vez adquiridos todos los materiales para el funcionamiento se procede a realizar la implementación del trabajo fin de master.

10.1.1 Unidad lineal

Una vez adquiridas las unidades lineales se procede a realizar mediciones para comprobar que todos los datos según su diseño de selección estén a corde, se toma en cuenta que el diámetro para la colocación del motor en cada unidad lineal es de 8mm y que su recorrido es de 2.70m.

Figura 96 Adquisición de la unidad lineal

Tal y como se realizó en el diseño en SolidWorks, se implementó la estructura de sujeción en la unidad lineal, a su vez, esta estructura sujeta el motor a inducción, debido a sus dimensiones, en el caso de la estructura para el servomotor, no es necesario sujetarlo a la estructura ya que posee bridas de sujeción en su propio diseño.

Figura 97 Estructura de sujeción para la unidad lineal y los motores

Con la ayuda de tornillos de métrica 12 y un nivel para la verticalidad, se sujetó en la pared las unidades lineales, utilizando todas las normas de seguridad estas se encuentran fijas de acuerdo a las especificaciones del laboratorio, con el fin de que los estudiantes puedan utilizarlas para prácticas.

Figura 98 Unidades lineales montadas en la pared

10.1.2 Motores

Con la llegada del motor a inducción y el servomotor, se analiza la placa característica correspondiente a cada una de estas, verificando que ambos sean idóneos para el accionamiento de las unidades lineales, de la misma forma se procede a colocar los acoples para constatar que fueron diseñados correctamente como se aprecia en la siguiente figura.

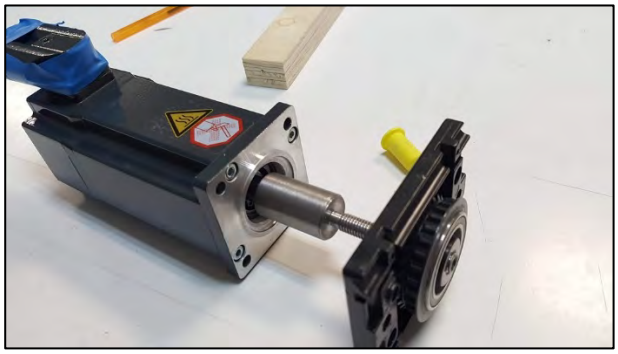


Figura 99 Prueba de servomotor-acople-unidad lineal

Procedemos a colocar el motor a inducción en la parte inferior de la unidad lineal, en la estructura de sujeción se diseñó un descanso para el motor a inducción en donde permanecerá unido junto con el encoder y el acople en la caja reductora. Templamos la correa de la unidad lineal para que quede correctamente sujeta evitando el deslizamiento que se pueda producir en el movimiento.

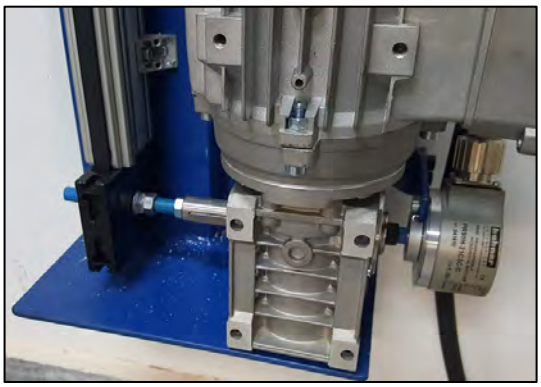


Figura 100 Montaje del motor asincrónico de inducción en la unidad lineal.

Para el caso del servomotor, el eje de giro no posee chaflanado por lo que sujetaremos mediante un prisionero de presión evitando perdidas de deslizamiento en la transmisión del movimiento a la unidad lineal.

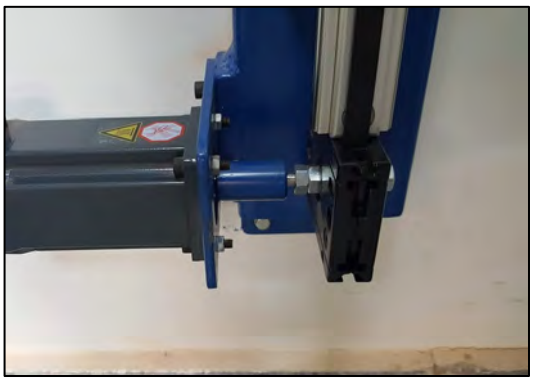


Figura 101 Montaje del motor sincrónico de imanes permanentes a la unidad lineal.

10.1.3 Sistemas de control

Las tarjetas de control Arduino y Raspberry son susceptibles al ambiente por lo que se decide hermetizar el control en cajas, para que se pueda visualizar el control evitando una mala manipulación de los equipos. En el caso de los variadores de frecuencia, poseen normas IP56 de fabricación por lo que se pueden conectar directamente al proceso. Utilizando los esquemas de conexión los dispositivos Arduino y Raspberry quedan implementados como se aprecia en la siguiente imagen.

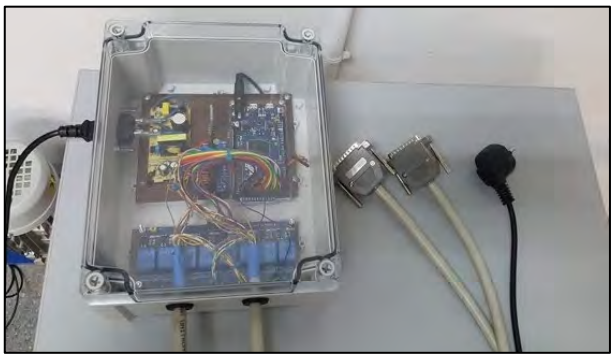


Figura 102 Montaje de los sistemas de control

10.1.4 Sensores y protección a la red

Como se había calculado inicialmente, se coloca el diferencial y el automático a la red eléctrica trifásica de 400V como se ve en la figura.

Figura 103 Montaje de protección a la red eléctrica

Utilizamos un armario de protecciones en donde estarán todas las conexiones eléctricas que puedan causar algún riesgo al usuario, además de los dispositivos de protección eléctrico. Finalmente queda todo asegurado para que al usuario pueda encender el sistema con facilidad.

Figura 104 Armario de protecciones

En el caso de los sensores inductivos y finales de carrera para el posicionamiento del elevador, son previamente probados, verificando su óptimo funcionamiento, para colocar en la estructura de sujeción se utilizan brocas de cobalto de métrica 3,8,12 y cónicas para insertar los sensores en la estructura como se puede apreciar en la siguiente imagen.

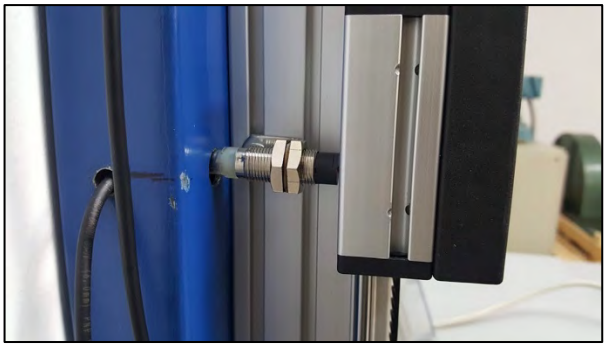


Figura 105 Montaje de los sensores de posicionamiento

10.2 Puesta en Marcha

Para iniciar la puesta en marcha conectamos los motores sin carga a la red, comprobando el correcto funcionamiento del control. Lo que se pretende evitar es estropear en caso de un mal funcionamiento las unidades lineales.

Figura 106 Puesta en marcha proceso y control del motor de inducción

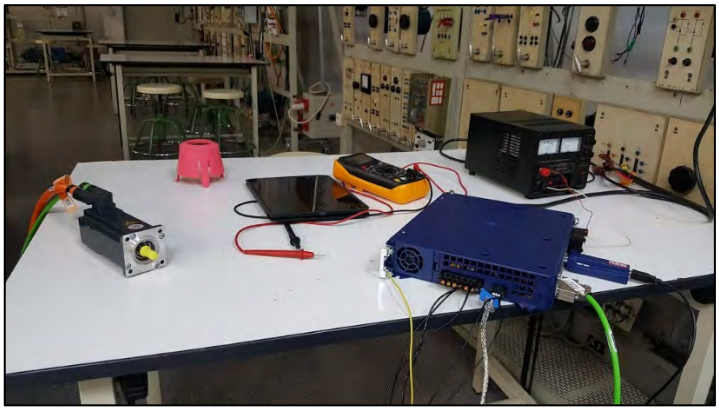


Figura 107 Puesta en marca proceso y control del servomotor

Los dispositivos de control y sus accionamientos funcionan adecuadamente, es necesario repasar el esquema de conexión y los manuales de uso de cada variador de frecuencia para no cometer equivocaciones de alimentación o control.

10.3 Pruebas

10.3.1 Control motor de inducción

Alimentamos el variador de frecuencia a 220V, conectamos las tarjetas Arduino y Raspberry para realizar pruebas de calidad y verificamos su funcionamiento.

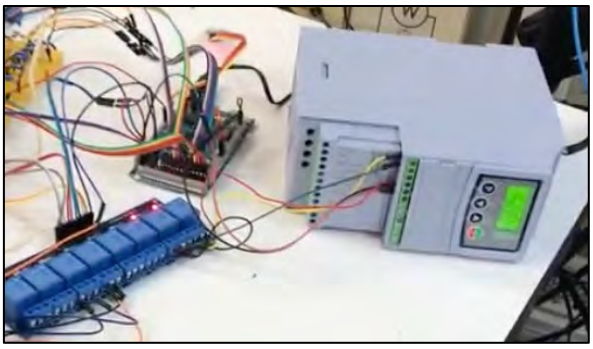


Figura 108 Prueba del control en el motor de inducción con Arduino

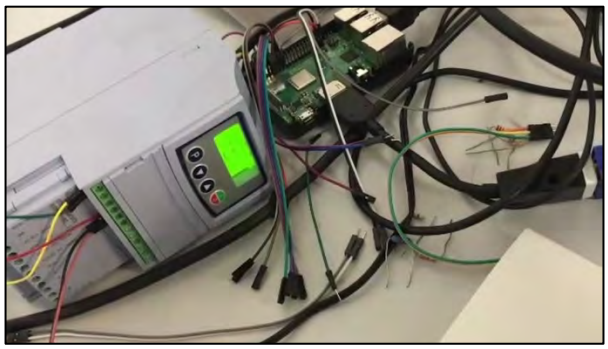


Figura 109 Prueba del control en el motor de inducción con Raspberry

Como se puede observar en las imágenes, tanto el Arduino como la Raspberry están conectadas a la regleta de relés, y estos a los pines digitales de entrada del variador de frecuencia, la señal PWM se puede apreciar en la pantalla HMI del variador de frecuencia, el sistema funciona correctamente en modo manual a las órdenes de paro marcha, sentido de giro y designación de distintas velocidades por lo que estaría adecuado para su instalación.

10.3.2 Control motor de imanes permanentes

Alimentamos el servo drive a 400V, para verificar el estado del mismo se utiliza el software de monitorización y accionamiento Combivis 6, aquí verificamos en qué estado se encuentra el servo drive ya que este no posee una pantalla HMI integrada como el variador de frecuencia, conectamos las tarjetas Arduino y Raspberry para realizar pruebas de calidad y verificamos su funcionamiento.

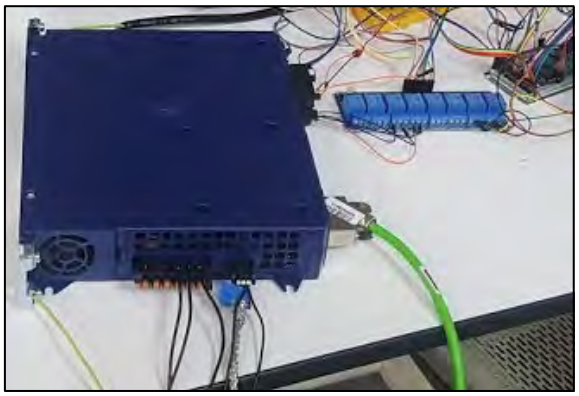


Figura 110 Prueba del control en el servo drive con Arduino

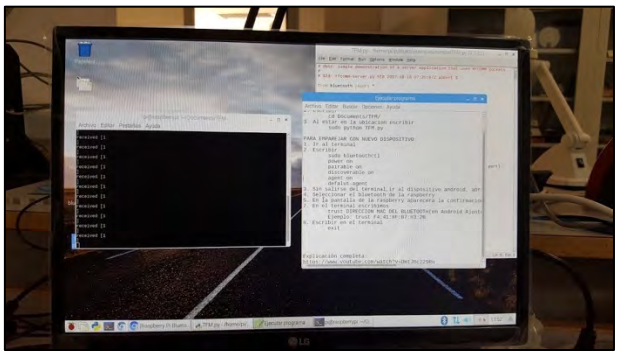


Figura 111 Prueba del control en el servo drive con Raspberry

10.3.3 Control de ascensor

Para el control del ascensor se utilizan los sensores, final de carrera y de inducción, inicialmente estos son colocados en una mesa de trabajo, en los laterales son colocados los motores para ver su comportamiento de giro y determinar si el control es adecuado.

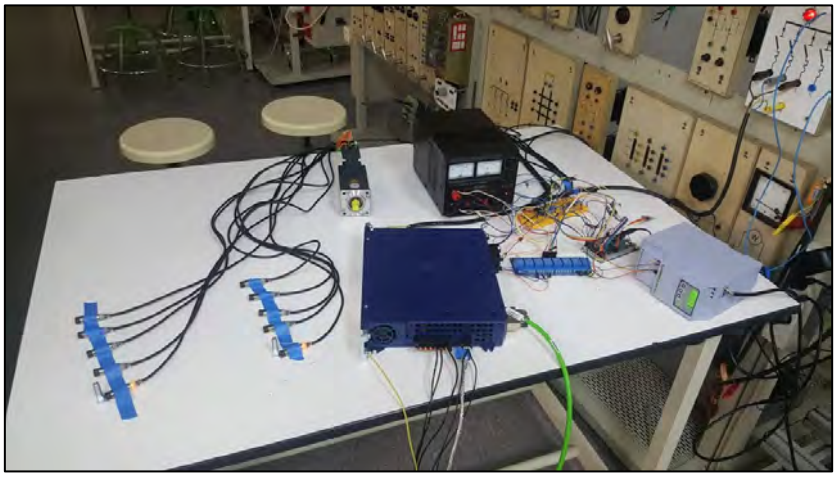


Figura 112 Prueba base en mesa de trabajo

En la imagen anterior se pueden apreciar los sensores de los 5 pisos encada unidad lineal, así mismo la conexión de los sensores y las tarjetas de control al variador de frecuencia y al servo drive, la respuesta en tiempo real es automática, posee un buen campo de funcionamiento, los tiempos de envío y recepción de datos se realizan a 9600 baudios por lo que el sistema funciona en óptimas condiciones como se puede apreciar en la siguiente figura.

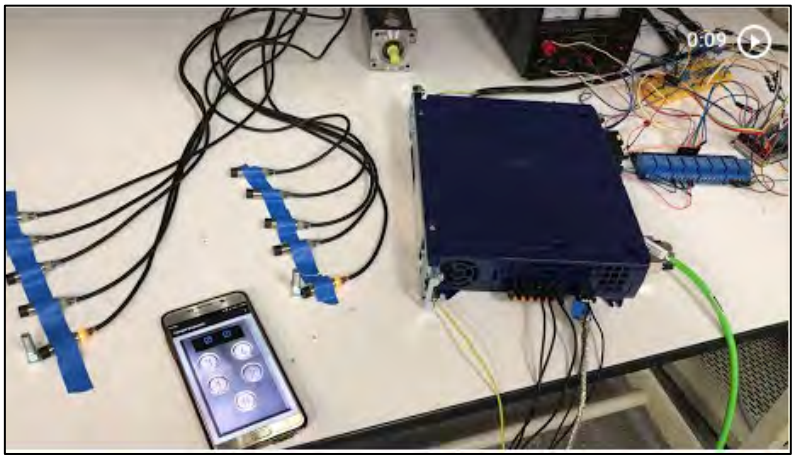


Figura 113 Funcionamiento del ascensor en mesa de trabajo

Una vez funciona en la mesa de trabajo se procede a implementar en las unidades lineales, a su vez se realizan pruebas de continuidad, testeo de objetos, revisión de esquemas de conexiones y dispositivos de seguridad para constatar que todo se encuentra correctamente conectado antes de energizar el sistema. Cuando energizamos el sistema procedemos a utilizar la app móvil y en modo manual y modo automático (selección de pisos por el usuario) comprobamos el correcto funcionamiento del proyecto.

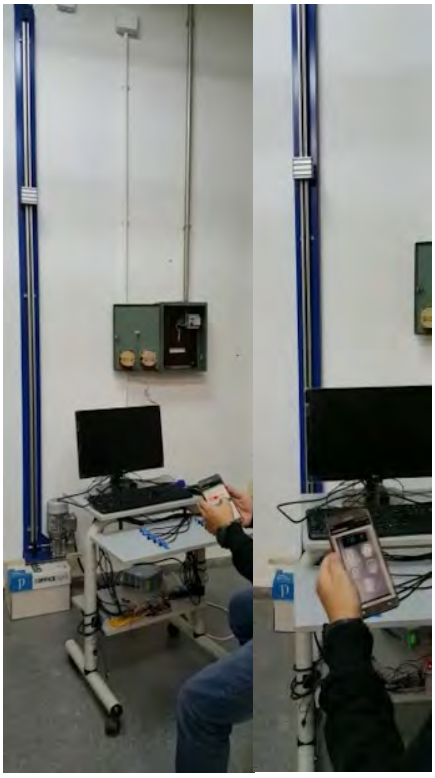


Figura 114 Control automático y manual en la Unidad Lineal

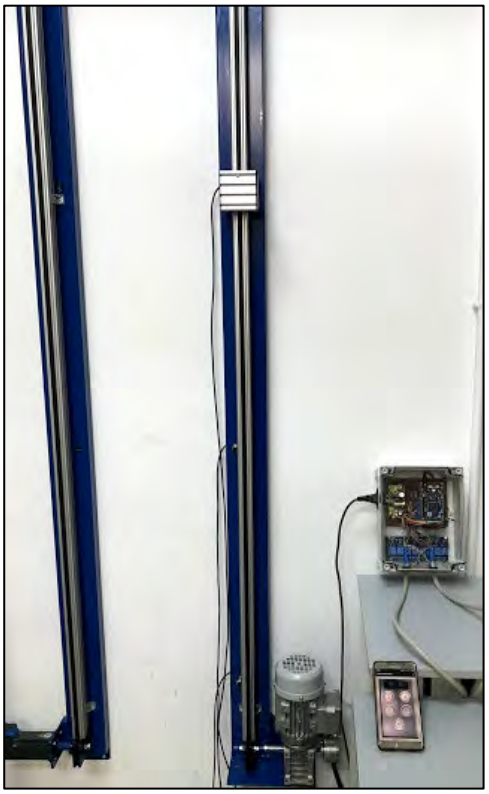


Figura 115 Implementación completa del Trabajo fin de Máster

10.3.4 Aplicación móvil

Para verificar la versatilidad de la aplicación móvil, se procede a instalar la misma en dispositivos con diferentes versiones de Android. El programa es hecho con el IDE de Google por lo que hasta el momento de su última compilación tiene una compatibilidad desde la versión 4.0.1 hasta la 8.0 de Android. Así mismo se utilizan distintos tamaños de pantalla para verifica una correcta relación de acuerdo al tamaño del dispositivo.

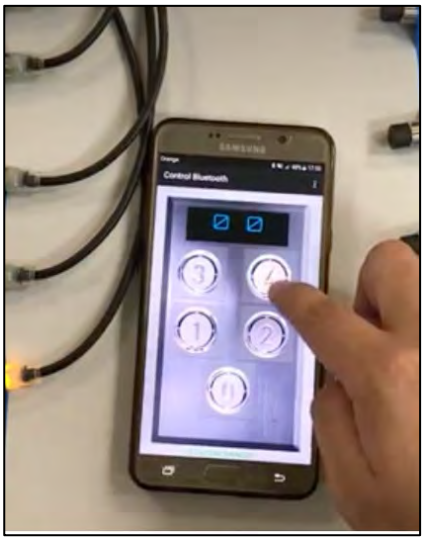


Figura 116 Uso de la aplicación móvil de control de las unidades lineales

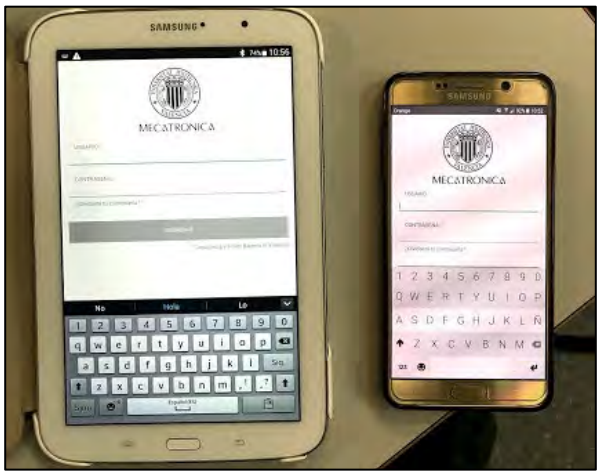


Figura 117 Adaptabilidad a distintos dispositivos y versiones Android

Como se puede apreciar en las figuras anteriores, la aplicación funciona correctamente indistintamente del dispositivo móvil, en la programación se implementaron los permisos bluetooth esto quiere decir que en caso de que el usuario no permita ocupar el bluetooth o el dispositivo carezca del mismo, la app indicará un mensaje explicando el error.

10.3.5 Osciloscopio

Con el uso del variador de frecuencias se procede a verificar si las gráficas implementadas son correctas, se coloca además un divisor de tensión para simular el comportamiento de una señal superior al voltaje necesario, como se pueden ver en las siguientes figuras, el dispositivo graficará 334 muestras tomadas cada 120us por muestra.

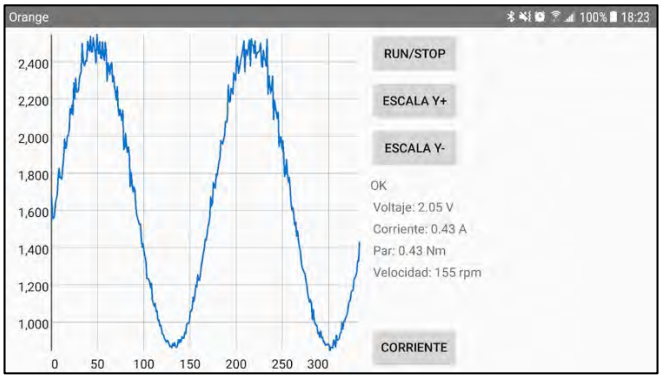


Figura 118 Onda senoidal representada en la aplicación de monitoreo

Como se ve en la figura anterior, verificamos que el sistema toma un número correcto de muestras, a partir del variador de frecuencia se envía una señal senoidal a 50Hz, la señal mostrada son 2 periodos de la onda, a través de los siguientes cálculos se verifica su funcionamiento.

$$t = \frac{1}{T}$$

$$t = \frac{1}{50} = 0.02s$$

Cada periodo de una onda senoidal de 50Hz, se representa en 0.02s.

$$t_{osciloscopio} = n_{muestras} * t_{muestreo}$$

$$t_{osciloscopio} = 334 * 120us$$

$$t_{osciloscopio} = 0.04s$$

$$T_{completos} = \frac{t_{osciloscopio}}{t_{senoidal\ 50\ HZ}} = \frac{0.04}{0.02} = 2$$

Con los que en la aplicación se vería 2 periodos completas de la onda senoidal, como se puede apreciar en la imagen anterior. Finalmente, para la implementación del osciloscopio en el monitoreo de los motores se procede a conectar los sensores de corriente y de voltaje en los motores.

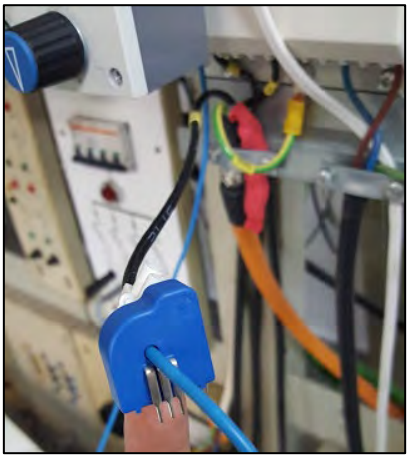


Figura 119 Sensores de corriente en una línea de alimentación al motor.

Una vez obtenida las muestras utilizamos el software de Matlab para graficar los resultados y analizar si cumple con el tiempo de muestreo por canal y como se puede ver, por cada canal se necesita un periodo de muestreo de 40us, como en el caso es de 3 canales el valor es de 120us.

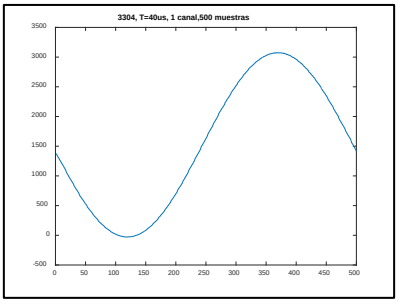


Figura 120 Análisis en Matlab de un canal a 40us de muestreo y 500 muestras

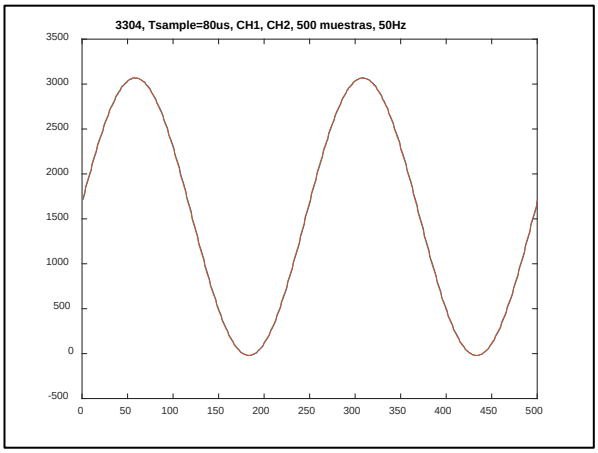


Figura 121 Análisis en Matlab de dos canales a 80us de muestreo y 500 muestras

Tabla 11 Análisis de velocidad con el osciloscopio

Puesto	PC	Arduino
1400	1398,4	1401
1410	1411,84	1411
1420	1417,66	1422
1430	1429,01	1429
1440	1438,04	1440
1450	1456,46	1452
1460	1459,51	1461
1470	1475,6	1469
1480	1478,81	1483
1490	1489,87	1492
1500	1498,14	1504
1510	1511,52	1509
1520	1517,32	1521
1530	1528,33	1532
1540	1538,12	1544
1550	1551,94	1551
1560	1558,04	1558
1570	1568,15	1573
1580	1575,08	1581
1590	1587,13	1590
1600	1592,86	1598

Comparamos la aplicación móvil de monitoreo con los valores de un osciloscopio portátil y el software del fabricante de monitoreo del variador y verificamos que los resultados arrojados son muy buenos. Cabe recalcar que los valores se ven al instante en la aplicación móvil de monitoreo, pero cuando se desea visualizar la gráfica es necesario un tiempo de refresco de 3 segundos debido a la limitación de comunicación bluetooth de 9600 baudios en donde al momento de graficar recibe 700 datos en menos de un segundo.

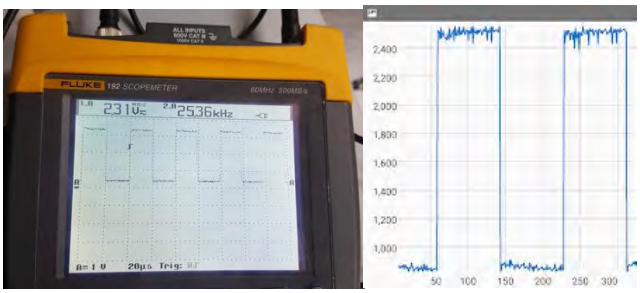


Figura 122 Comparación señal Encoder entre osciloscopio portátil y aplicación con Lolin I2C PRO

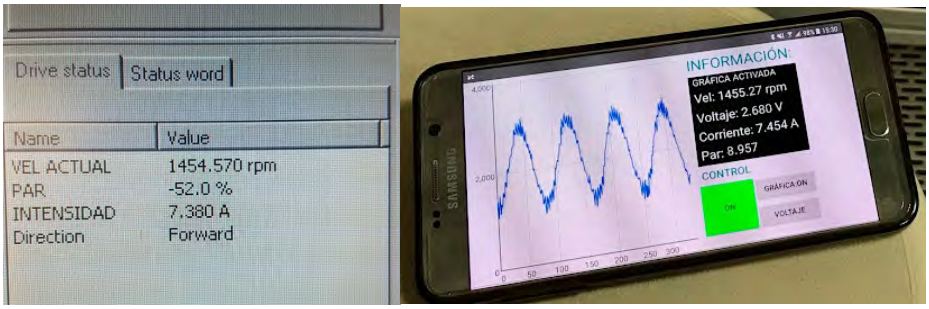


Figura 123 Comparación de medición de Corriente, Par, Velocidad y gráfica de corriente entre aplicación y monitoreo de fabricante de variador

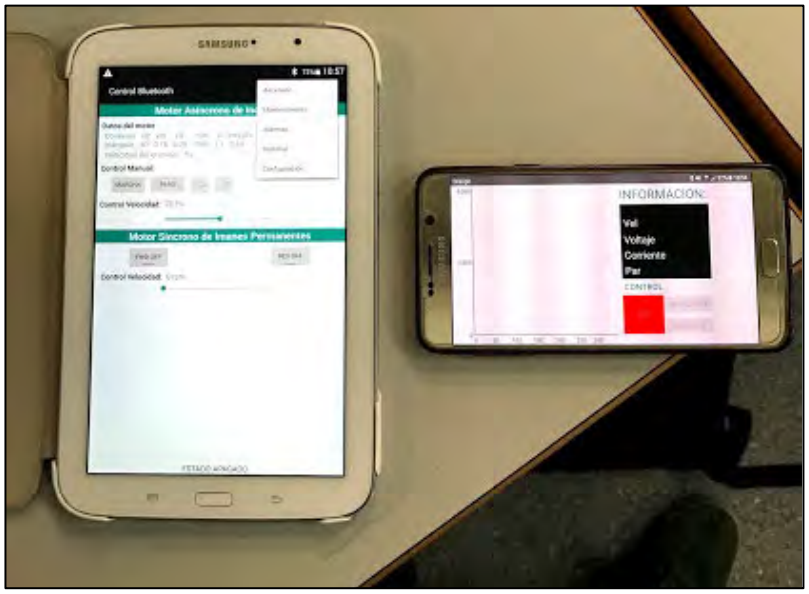


Figura 124 Aplicación de control y de monitoreo para ascensores montacargas

11. Presupuesto

En el este presupuesto no solo se incluye las horas de programación del sistema, sino todo lo necesario para su implantación y puesta en marcha en el prototipo ascensor

Se dividirá en 3 bloques: Procesos, Control y Servicios prestados. Partiendo de que este proyecto busca brindar información, apoyar el desarrollo y apertura a otras técnicas de accionamientos y control, se toma en cuenta los materiales implementados para este trabajo fin de master. El presupuesto fue dado en su totalidad por la Universitat Politècnica de València.

Bloque: Proceso			
Componentes	Precio Unidad	Cantidad	Precio Total
LRE5 D6ZU40R10	1.115,00 €	2	2.230,00 €
Encoder M23 HOHNER	315,00 €	1	315,00 €
Conector Hembra Encoder	11,14 €	1	11,14 €
Convertidor de Frecuencia WEG CFW300	198,00 €	1	198,00 €
Módulo Encoder para CFW300	99,00 €	1	99,00 €
Motor asíncrono de inducción REM	83,00 €	1	83,00 €
Caja Reductora MU R 1/5	227,00 €	1	227,00 €
Servo drive S6K	461,25 €	1	461,25 €
Cable para servomotor y resolver	147,50 €	1	147,50 €
Motor síncrono de imanes permanentes KEB	358,43 €	1	358,43 €
Estructura y soporte	400,00 €	2	800,00 €
Acople UL-Servomotor	200,00 €	1	200,00 €
Acople UL-Motor de inducción	190,00 €	1	190,00 €
Sensores de Inducción	26,03 €	10	260,30 €
Sensores fin de carrera	11,02 €	4	44,08 €
Interruptor de protección automático	25,00 €	1	25,00 €
Interruptor diferencial	87,01 €	1	87,01 €
Adaptador Combicom USB-RS485HTL	150,00 €	1	150,00 €
Fuente de alimentación de 24V	29,77 €	1	29,77 €
		Total	5.916,48 €

Bloque: Control			
Componentes	Precio Unidad	Cantidad	Precio Total
ABB PLC AC500	616,43 €	1	616,43 €
ABB Fuente de 24 V	82,32 €	1	82,32 €
ABB DC541	414,44 €	1	414,44 €
Pulsadores	10,00 €	6	60,00 €
Arduino Due	47,00 €	3	141,00 €
Raspberry Pi3 B+	35,66 €	2	71,32 €
Módulo Relé 8 canales	10,99 €	2	21,98 €
Módulo bluetooth HC06	7,99 €	3	23,97 €
Lolin I2C pro	15,00 €	1	15,00 €
Sensor LTS6NP	25,00 €	1	25,00 €
MCP3304	3,11 €	12	37,32 €
Caja de ABS Fibox	21,96 €	3	65,88 €
Resistencias varias	0,25 €	35	8,75 €
Cables Hembra-Hembra	7,99 €	1	7,99 €
Cables Hembra-Macho	7,99 €	1	7,99 €
Cables Macho-Macho	7,99 €	1	7,99 €
Placa de Matriz con orificios	8,52 €	4	34,08 €
Conectores D-SUB25 Macho	2,29 €	4	9,16 €
Conectores D-SUB25 Hembra	2,29 €	2	4,58 €
Kit complementario Raspberry	30,00 €	1	30,00 €
Cables varios	50,00 €	1	50,00 €
Fuente de alimentación de 5V Arduino	23,40 €	1	23,40 €
		Total	1.735,20 €

Bloque: Servicios Prestados			
Servicios	Precio por hora	Horas	Precio Total
Programar Arduino DUE	20,00 €	80	1.600,00 €
Programar Raspberry	20,00 €	80	1.600,00 €
Programar PLC	20,00 €	25	500,00 €
Programar Aplicación Android	30,00 €	80	2.400,00 €
Programar Arduino I2C pro	20,00 €	80	1.600,00 €
Diseñar acople UL-servomotor	20,00 €	20	400,00 €
Diseñar acople UL-motor inducción	20,00 €	20	400,00 €
Instalar caja de control Arduino	25,00 €	8	200,00 €
Instalar caja de control Raspberry	25,00 €	8	200,00 €
Parametrizar variadores de frecuencia	30,00 €	50	1.500,00 €
	Total	451	10.400,00 €
TOTAL, DEL PROYECTO	18.051,68 €		

12. Conclusiones

I. A través del software de análisis de esfuerzos las unidades lineales del prototipo de ascensor-montacargas fueron sometidas a distintas cargas, cumpliendo con las condiciones de diseño y parámetros que un ascensor real debe cumplir, la selección del material de construcción de las unidades lineales es de suma importancia para que no exista una deformación o cortadura del material al momento de aplicarse fuerzas en el ascensor. Tomando en cuenta los valores de deformación determinados por el software, verificamos que sus puntos de corte serían en la parte superior de la unidad lineal, siempre y cuando la fuerza aplicada sea tres veces más que su carga máxima, aun así, la transmisión de desplazamiento por correa soporta una fuerza máxima de 150N por lo que la correa se ropería antes de que exista un esfuerzo en el material, evitando un desgaste o flexión de la unidad lineal.

II. Tanto los acoples de los motores y la estructura de sujeción de la unidad lineal fueron mecanizados de acuerdo a los planos de construcción realizados en el software de SolidWorks, el diseño por software es de suma importancia ya que hoy en día las grandes empresas utilizan estos programas para el diseño y construcción de distintas maquinarias y accesorios, obteniendo una confiabilidad de los datos de diseño para evitar la falla o colisión de los elementos mecánicos con un factor de seguridad que dependa del conocimiento y criterio de quien lo diseñe.

III. Tomando en cuenta que el proyecto fue instalado en el laboratorio de máquina eléctricas de la Universitat Politècnica de València, se implementaron las conexiones de alimentación de los accionamientos y de la sensorización del prototipo siguiendo la normalización de los esquemas eléctricos, separando al prototipo y al usuario de la red eléctrica a través de un armario de protección en donde se encuentran las conexiones eléctricas tanto del proceso como del control.

IV. La esquematización eléctrica fue basada en conocimientos adquiridos a lo largo del máster tanto de redes eléctricas, dispositivos de protección, conexión de

sensores, alimentación de motores, señales de control y cuadros eléctricos por lo que fue de suma importancia adquirir todos esos conocimientos previos y valorar la importancia que tienen en una aplicación industrial real, ya que gracias a esto se pudo instalar adecuadamente la alimentación para el prototipo de ascensor – montacargas sin riesgos eléctricos.

V. Los sistemas de control de las unidades líneas, se realizaron utilizando software de código abierto, esto implica que el proyecto está abierto a posibles mejoras, aplicaciones, o adaptaciones según requiera el usuario final, la programación es multiplataforma ya que se implementó en Arduino, Autómata programable, Raspberry y Android, tomando en cuenta la adaptabilidad de programación y las ventajas que posee cada uno de los softwares.

VI. El uso de una aplicación móvil va de acuerdo a las nuevas tendencias tecnológicas, todos los desarrollos industriales implican un mayor confort para el usuario por lo que es necesario adaptarse a la evolución tecnológica, a través de la aplicación remota en Android, constatamos que el usuario pueda utilizar de forma confortable el prototipo de ascensor montacargas.

VII. Como se pudo ver en el presupuesto, el proyecto es implementado utilizando los mejores componentes tanto de accionamiento como de control, esto es con el fin de que el prototipo sea de uso didáctico, ofreciendo las mejores prestaciones y ayudando a que los nuevos estudiantes de grado y máster de la Universitat Politècnica de València, adquieran un mayor conocimiento de los ecosistema mecatrónico, comparando dos diferentes accionamientos a través de tres sistemas de control, lo cual ayuda a que no solo se destinen las aplicaciones industriales en un solo camino, es decir se abran paso nuevas tendencias de control para aplicaciones básicas en donde el uso de los autómatas programables no sea requerido.

VIII. Comparando los tres sistemas de control existe una diferencia clara en la calidad precio de las tarjetas, si se desea realizar aplicaciones industriales no hay duda que el autómata programable es la mejor opción sin embargo cuando la aplicación es didáctica, o son dispositivos de automatización de uso doméstico, las tarjetas de software y hardware abierto son una opción aceptable siempre y cuando el código de control sea adecuado.

13. Bibliografía

[1]. ArduinoSA. (2015). *Arduino. Arduino LLC.* StoreArduino.

[2]. Areny, R. P. (2014). *Sensores y acondicionadores de señal. Marcombo.*

[3]. Balcells, J. R. (2010). *Autómatas programables.* Marcombo.

[4]. Cho, K. V. (2014). *Learning phrase representations using RNN encoder-decoder for statistical machine translation.* arXiv preprint arXiv.

[5]. Company, O. E. (05 de 11 de 2018). *Otis España*. Obtenido de Otis España: https://www.otis.com/es/es/

[6]. Dorf, R. C. (2005). *Sistemas de control moderno.* Pearson Prentice Hall.

[7]. ElevatorDriveS.A. (25 de 06 de 2017). *Elevator Drive*. Obtenido de Elevator Drive: https://elevator-drives.es/producto/otis-ovf1-3/

[8]. Farnell. (16 de 11 de 2018). *Farnell España*. Obtenido de Farnell España: https://es.farnell.com/digilent/410-321/osciloscopio-usb-2-canales-30mhz/dp/2528523

[9]. Fernández, J. A. (21 de 01 de 2019). *Portal Electrozona*. Obtenido de Portal Electrozona: https://www.portalelectrozona.com/menuseccionplcomron/8-categoriacursoautomata/160-articulocableadodelautomata-3

[10]. Llamas, L. (21 de 09 de 2018). *LuisLlamas*. Obtenido de LuisLlamas: https://www.luisllamas.es/arduino-spi/

[11]. Moeller. (2015). Manual de esquemas Moeller. En Moeller, *Manual de esquemas Automatización y Distribución de Energía* (pág. 54). Moeller GmbH, Bonn.

[12]. montacarichi, D. A. (21 de 10 de 2018). *Daldoss* . Obtenido de Daldoss: https://daldoss.com/

[13]. Orona. (21 de 06 de 2017). *Orona España*. Obtenido de Orona España: https://www.orona.es/es-es

[14]. Pérez-Formoso, J. L.-F.-C.-T. (2010). *Dermatitis de contacto a acrilatos en una industria de fabricación de ascensores. A propósito de 8 casos.* Actas Dermo-Sifiliográficas.

[15]. PicoTech. (01 de 12 de 2018). *PicoTech*. Obtenido de PicoTech: https://www.picotech.com/products/oscilloscope

[16]. Sensor, S. (11 de 12 de 2018). *SICK España*. Obtenido de SICK España: https://www.sick.com/es/es/sectores/gestion-de-edificios/ascensor/supervision-de-velocidad-y-posicion-de-la-cabina-del-ascensor-con-sensores-de-medicion-lineales/c/p522268

[17]. Tedesco, C. F. (2011). *Ascensores electrónicos y variadores de velocidad.* Francisco Etchelecu.

[18]. Upton, E. &. (2014). *Raspberry Pi user guide.* John Wiley & Sons.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PROTOTIPO DE UN ASCENSOR Y SU CONTROL COMPARANDO TECNOLOGÍAS DIFERENTES DE ACCIONAMIENTO

ANEXOS

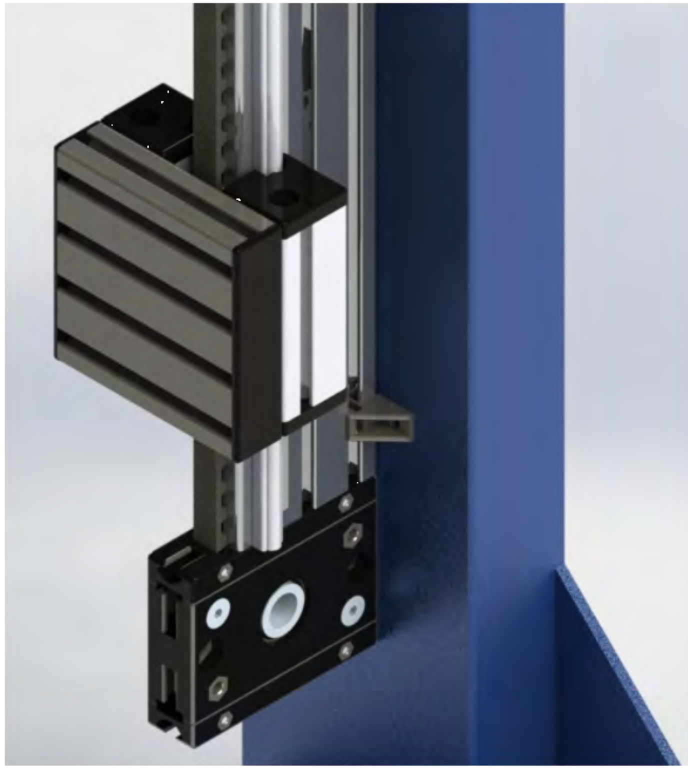
Índice:

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

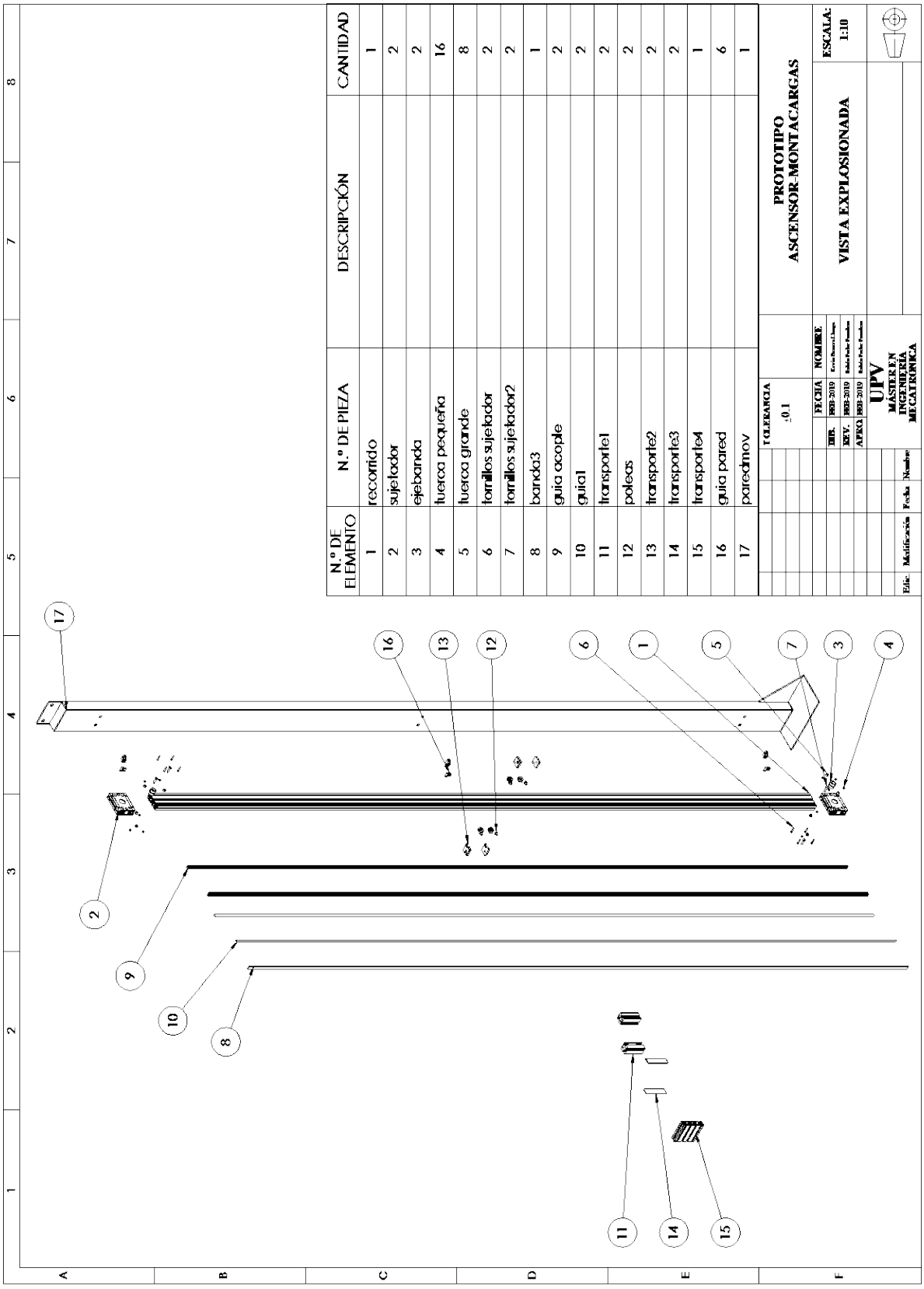
PROTOTIPO DE UN ASCENSOR Y SU CONTROL COMPARANDO TECNOLOGÍAS DIFERENTES DE ACCIONAMIENTO

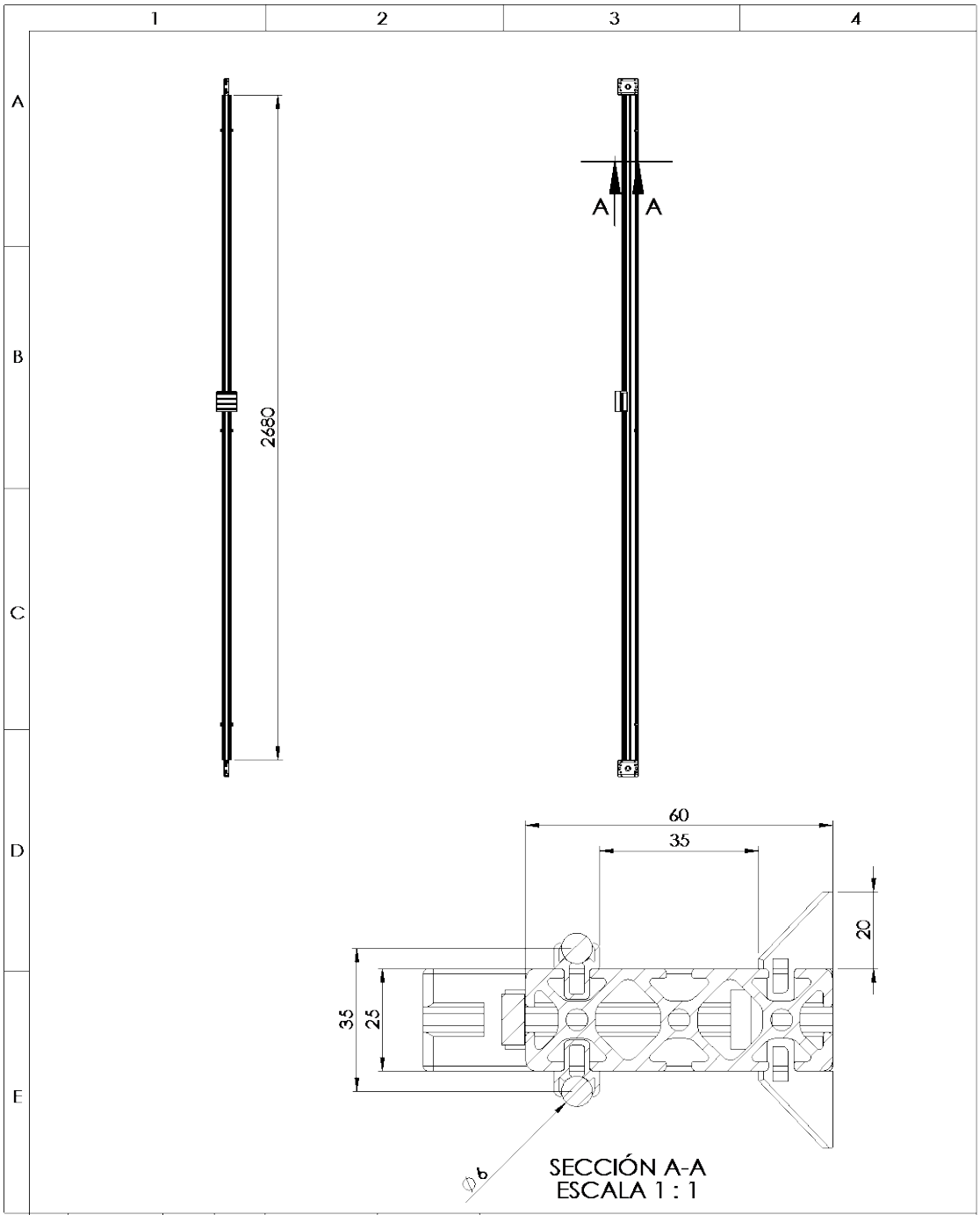
ANEXO I: PLANOS DE CONSTRUCCIÓN



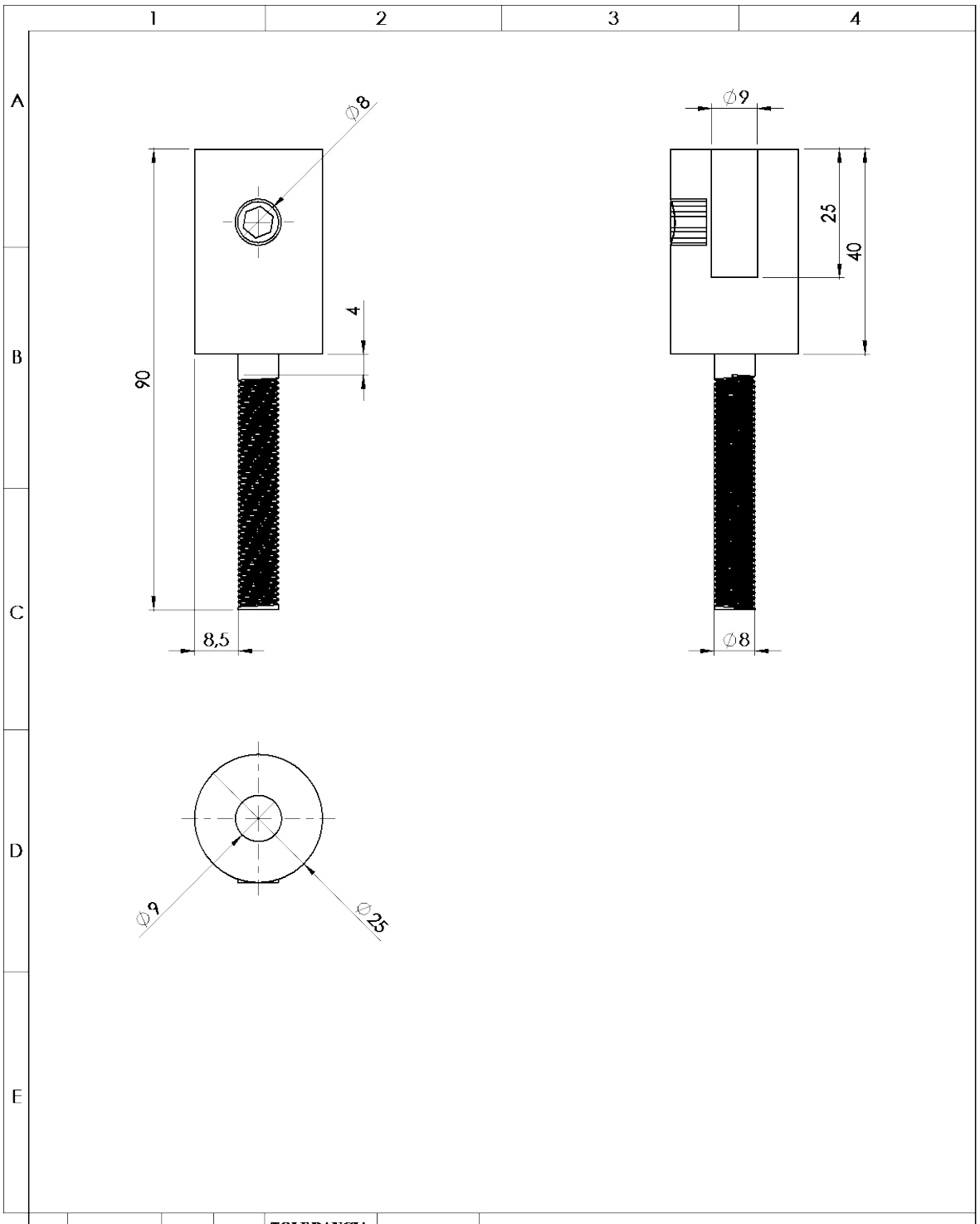
TOLERANCIA ±0.1			PROTOTIPO ASCENSOR-MONTACARGAS	
	FECHA	NOMBRE	DISEÑO COMPLETO	ESCALA: 1:20
DIB.	FEB-2019	[illegible]		
REV.	FEB-2019	[illegible]		
APRO.	FEB-2019	[illegible]		
UPV MÁSTER EN INGENIERÍA MECATRÓNICA				

Edic.	Modificación	Fecha	Nombre

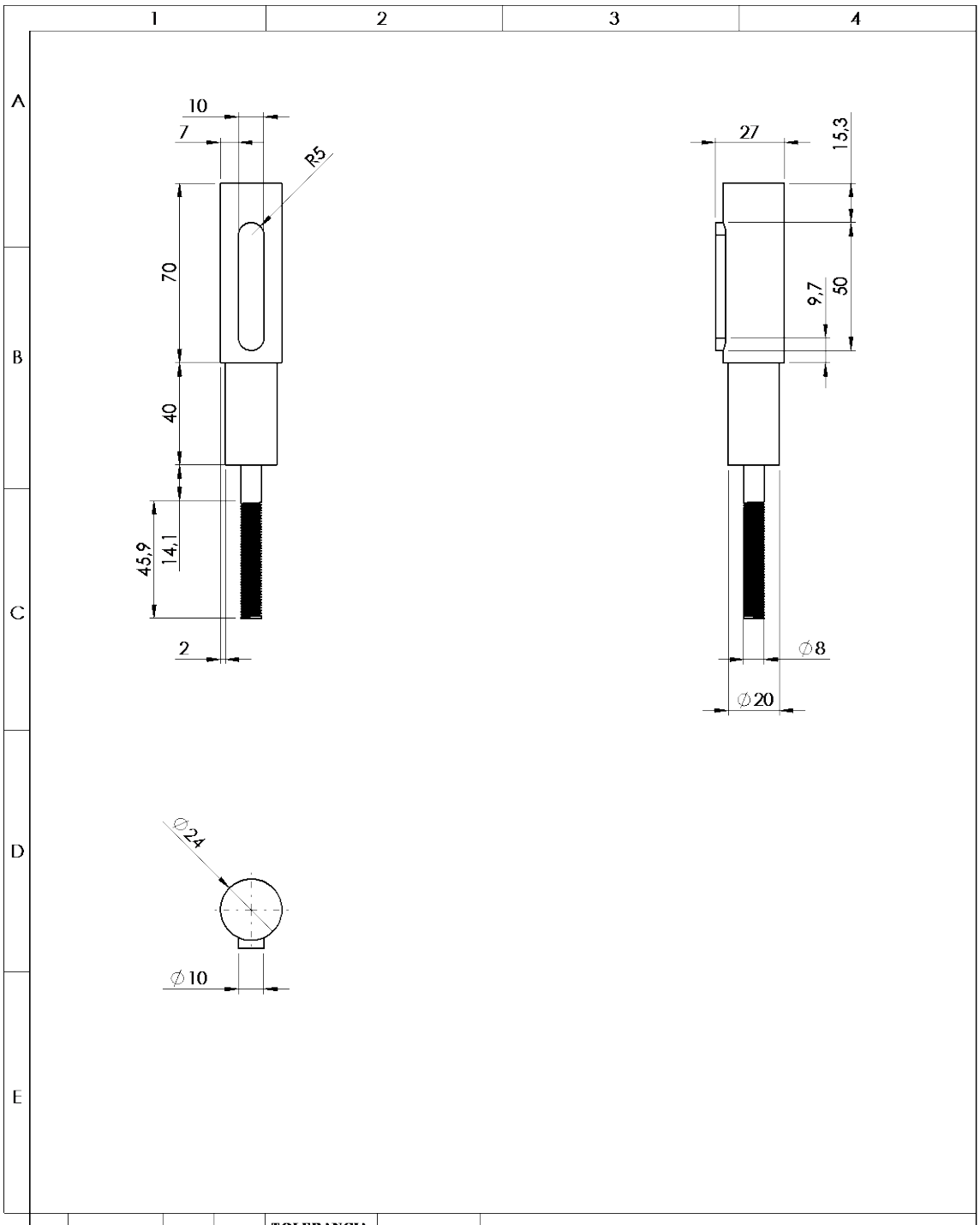

N.º DE ELEMENTO
N.º DE PIEZA
DESCRIPCIÓN
CANTIDAD
1 recorrido 1
2 sujetador 2
3 ejebarra 2
4 tuerca pequeña 16
5 tuerca grande 8
6 tornillos sujetador 2
7 tornillos sujetador2 2
8 barra3 1
9 guia acople 2
10 guia1 2
11 transporte1 2
12 poleas 2
13 transporte2 2
14 transporte3 2
15 transporte4 1
16 guia pared 6
17 paredmov 1
PROTOTIPO ASCENSOR-MONTACARGAS
TOLERANCIA ±0.1
DIB. FECHA NOMBRE
2020-2019
REV. 2020-2019
APRO. 2020-2019
UPV MÁSTER EN INGENIERÍA MECATRÓNICA
VISTA EXPLOSIONADA
ESCALA: 1:10
Edic. Modificación Fecha Nombre



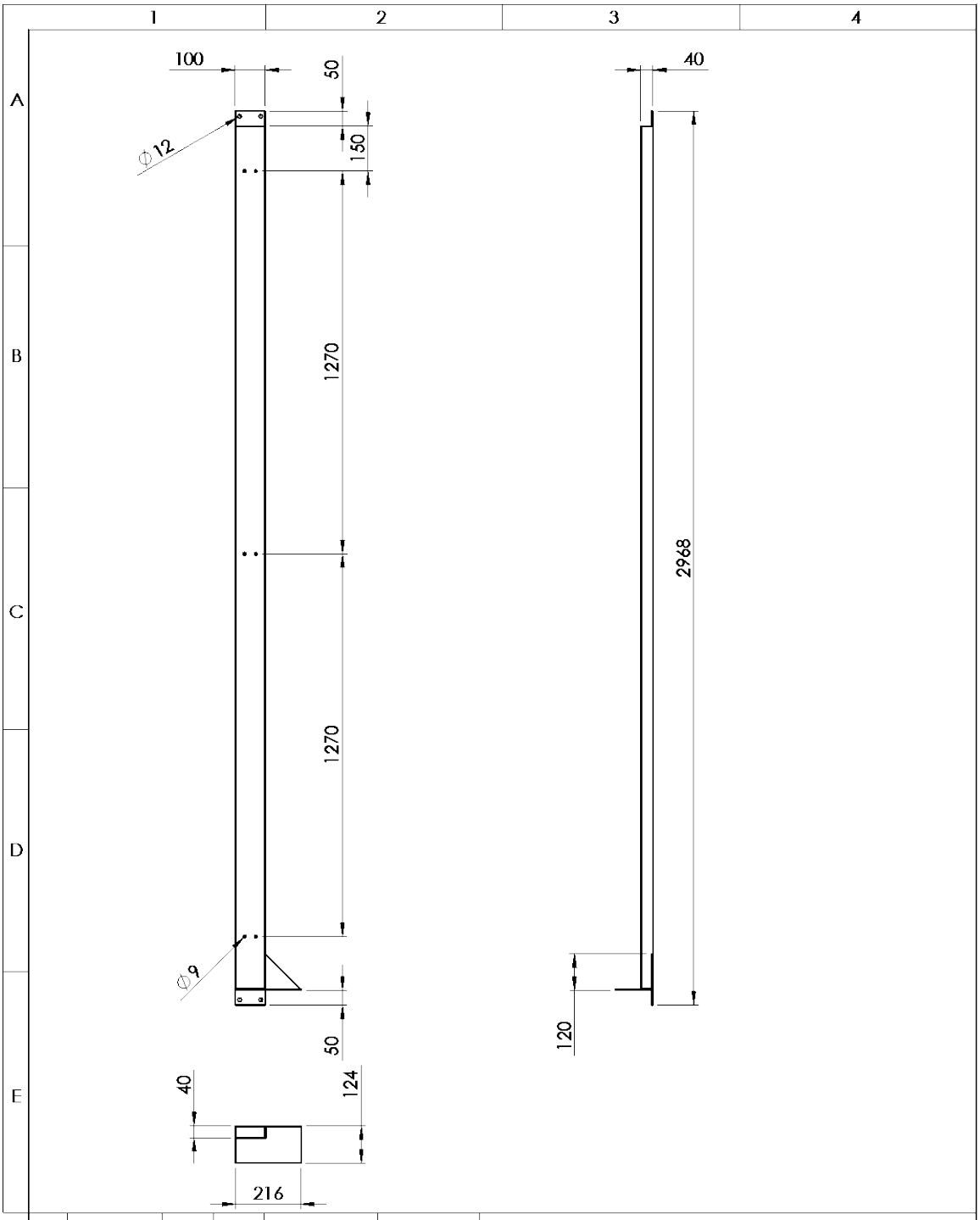
				TOLERANCIA ±0.1		PROTOTIPO ASCENSOR-MONTACARGAS
					FECHA / NOMBRE	
				DIB.	FEB-2019 Kevin Barrera Llauger	Area de trabajo — ESCALA: 1:10
				REV.	FEB-2019 Rubén Peche Panaleso	
				APRO.	FEB-2019 Rubén Peche Panaleso	
Edic.	Modificación	Fecha	Nombre	UPV MÁSTER EN INGENIERÍA MECATRÓNICA		



				TOLERANCIA ±0.1		PROTOTIPO ASCENSOR-MONTACARGAS
					FECHA / NOMBRE	
				DIB.	FEB-2019 Kevin Barrera Llanga	Acople Servomotor — ESCALA: 1:1
				REV.	FEB-2019 Rubén Pardo Pamaleo	
				APRO.	FEB-2019 Rubén Pardo Pamaleo	
Edic.	Modificación	Fecha	Nombre	UPV MÁSTER EN INGENIERÍA MECATRÓNICA		



				TOLERANCIA ±0.1		PROTOTIPO ASCENSOR-MONTACARGAS
					FECHA / NOMBRE	
				DIB.	FEB-2019 / Kevin Barrera Llanga	Acople caja reductora — ESCALA: 1:2
				REV.	FEB-2019 / Rubén Puche Panadero	
				APRO.	FEB-2019 / Rubén Puche Panadero	
Edic.	Modificación	Fecha	Nombre	UPV MÁSTER EN INGENIERÍA MECATRÓNICA		



				TOLERANCIA ±0.1		PROTOTIPO ASCENSOR-MONTACARGAS
					FECHA	NOMBRE
				DIB.	FEB-2019	Kevin Barrera Llanga
				REV.	FEB-2019	Rubén Puche Panadero
				APRO.	FEB-2019	Rubén Puche Panadero
Edic.	Modificación	Fecha	Nombre	UPV MÁSTER EN INGENIERÍA MECATRÓNICA		Soporte Pared Unidad Lineal

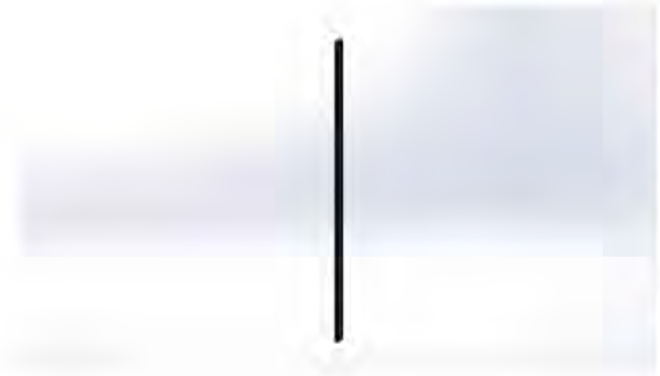
ESCALA: 1:17

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PROTOTIPO DE UN ASCENSOR Y SU CONTROL COMPARANDO TECNOLOGÍAS DIFERENTES DE ACCIONAMIENTO

ANEXO II: ANÁLISIS DE ESFUERZOS



Simulación de recorrido

Fecha: martes, 12 de febrero de 2019
Diseñador: Solidworks
Nombre de estudio: SimulationXpress Study
Tipo de análisis: Análisis estático

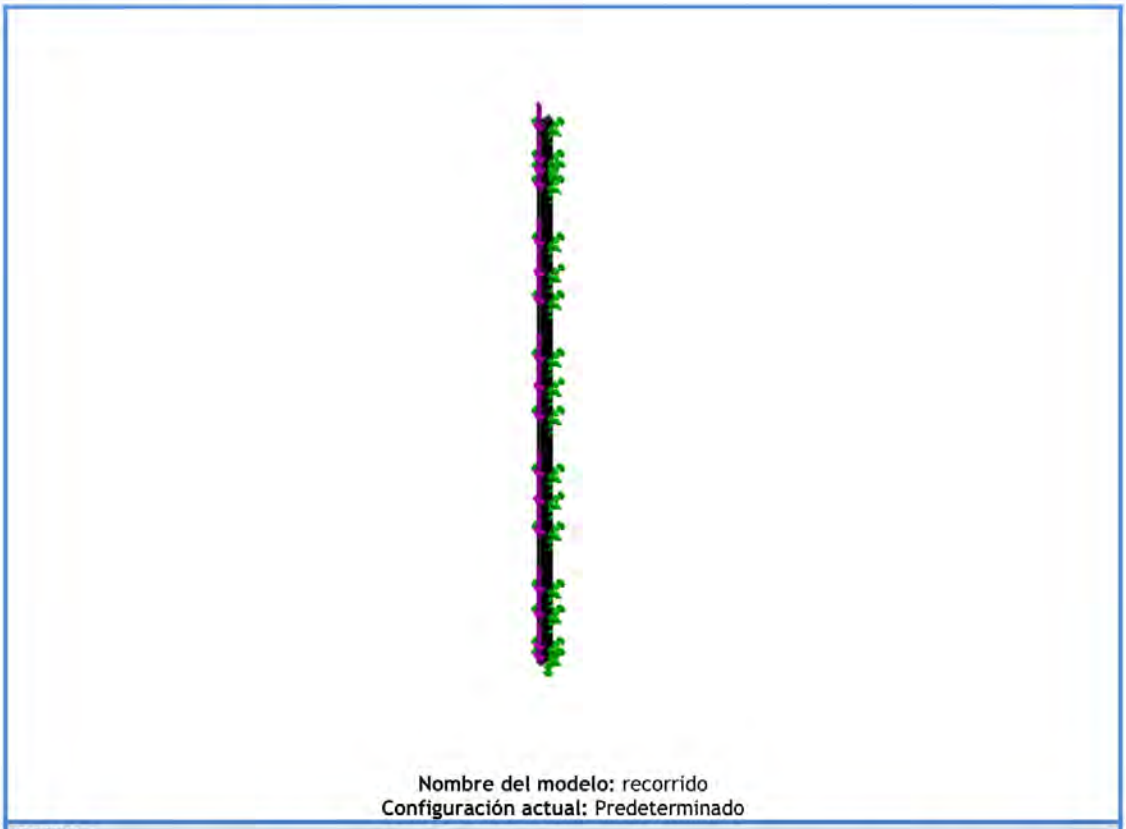
Tabla de contenidos

Descripción

Análisis barra deslizante Ascensor montacargas

Suposiciones

Información de modelo



Nombre del modelo: recorrido
Configuración actual: Predeterminado

Sólidos

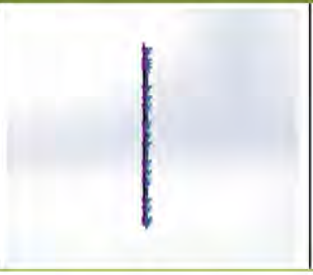
Nombre de documento y referencia	Tratado como	Propiedades volumétricas	Ruta al documento/Fecha de modificación

Saliente-Extruir1 	Sólido	**Masa:3.36641 kg** **Volumen:0.00124682 m^3** **Densidad:2700 kg/m^3** **Peso:32.9908 N**	**C:\Users\Kevin\Desktop\motor a induccion\recorrido.SLDPRT** Feb 12 18:39:08 2019

Propiedades de material

Referencia de modelo	Propiedades		Componentes
	**Nombre:** **Tipo de modelo:** **Criterio de error predeterminado:** **Limite elástico:** **Limite de tracción:**	Aleación 1060 Isotrópico elástico lineal Tensión de von Mises máx. 2.75742e+007 N/m^2 6.89356e+007 N/m^2	Sólido 1(Saliente-Extruir1)(recorrido)

Cargas y sujeciones

Nombre de sujeción	Imagen de sujeción	Detalles de sujeción
Fijo-2		**Entidades:** 2 cara(s) **Tipo:** Geometría fija

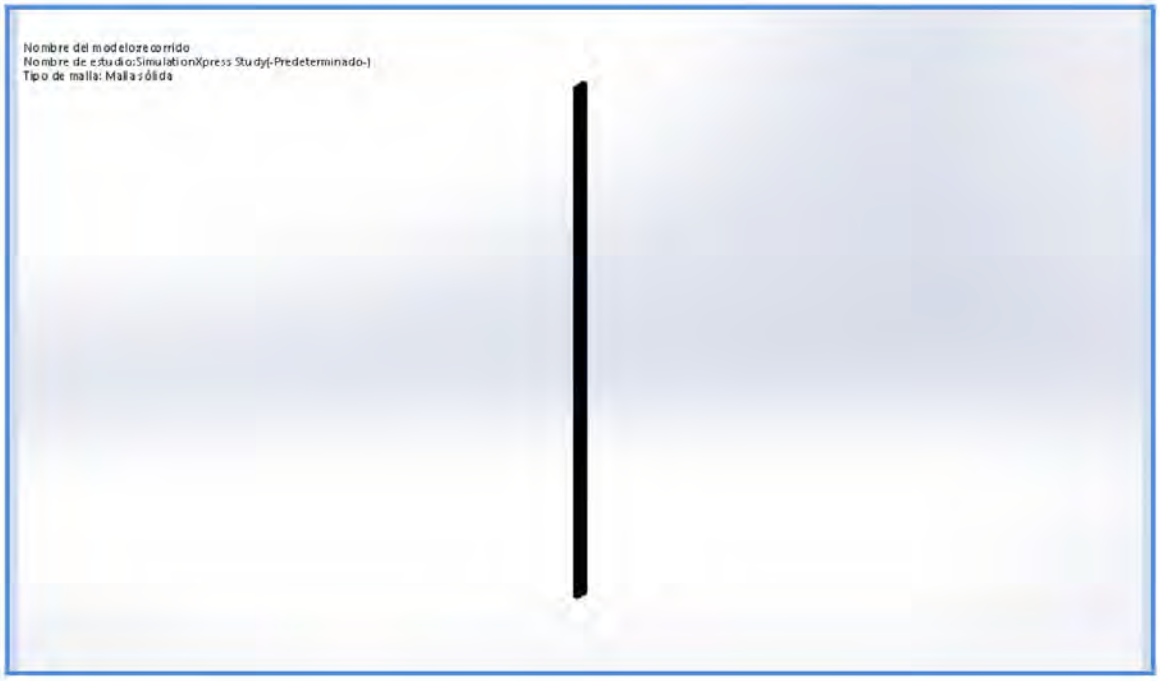
Nombre de carga	Cargar imagen	Detalles de carga
Fuerza-1		**Entidades:** 2 cara(s), 1 plano(s) **Referencia:** Planta **Tipo:** Aplicar fuerza **Valores:** ---, ---, -300 N

Información de malla

Tipo de malla	Malla sólida
Mallador utilizado:	Malla estándar
Transición automática:	Desactivar
Incluir bucles automáticos de malla:	Desactivar
Puntos jacobianos	4 Puntos
Tamaño de elementos	21.4386 mm
Tolerancia	1.07193 mm
Trazado de calidad de malla	Elementos cuadráticos de alto orden

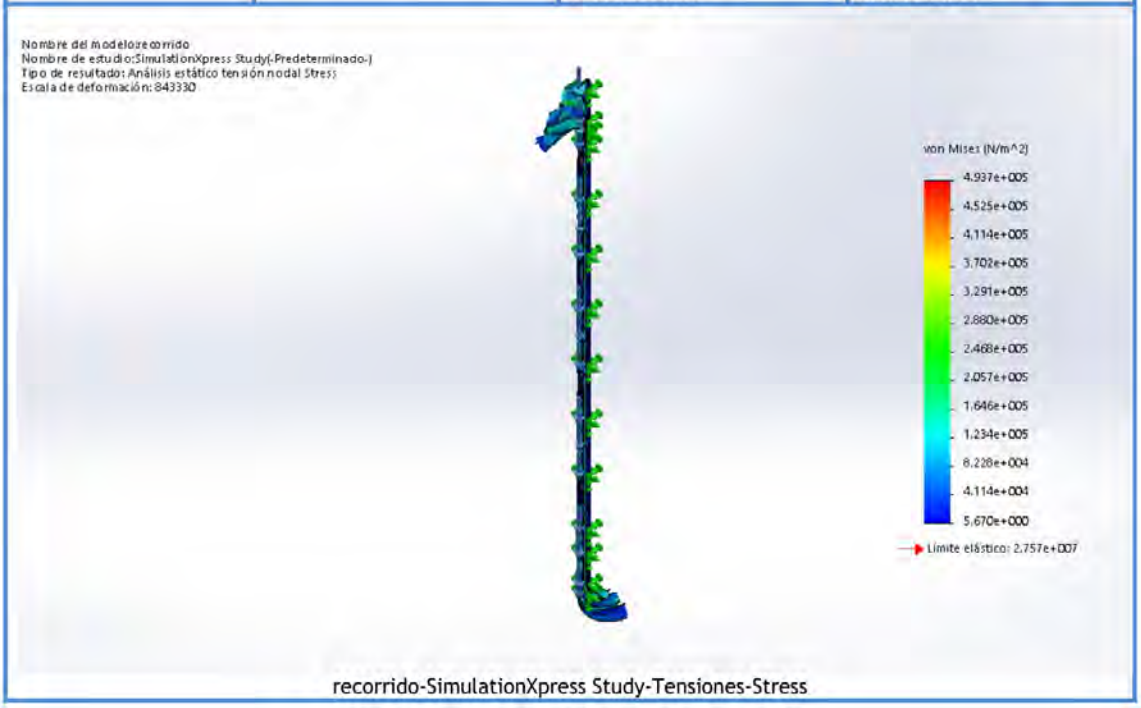
Información de malla - Detalles

Número total de nodos	122834
Número total de elementos	71741
Cociente máximo de aspecto	98.008
% de elementos cuyo cociente de aspecto es < 3	2.25
% de elementos cuyo cociente de aspecto es > 10	63.2
% de elementos distorsionados (Jacobiana)	0
Tiempo para completar la malla (hh;mm;ss):	00:01:25
Nombre de computadora:	


Nombre del modelo:recorrido
Nombre de estudio:SimulationXpress Study(-Predeterminado-)
Tipo de malla: Malla sólida

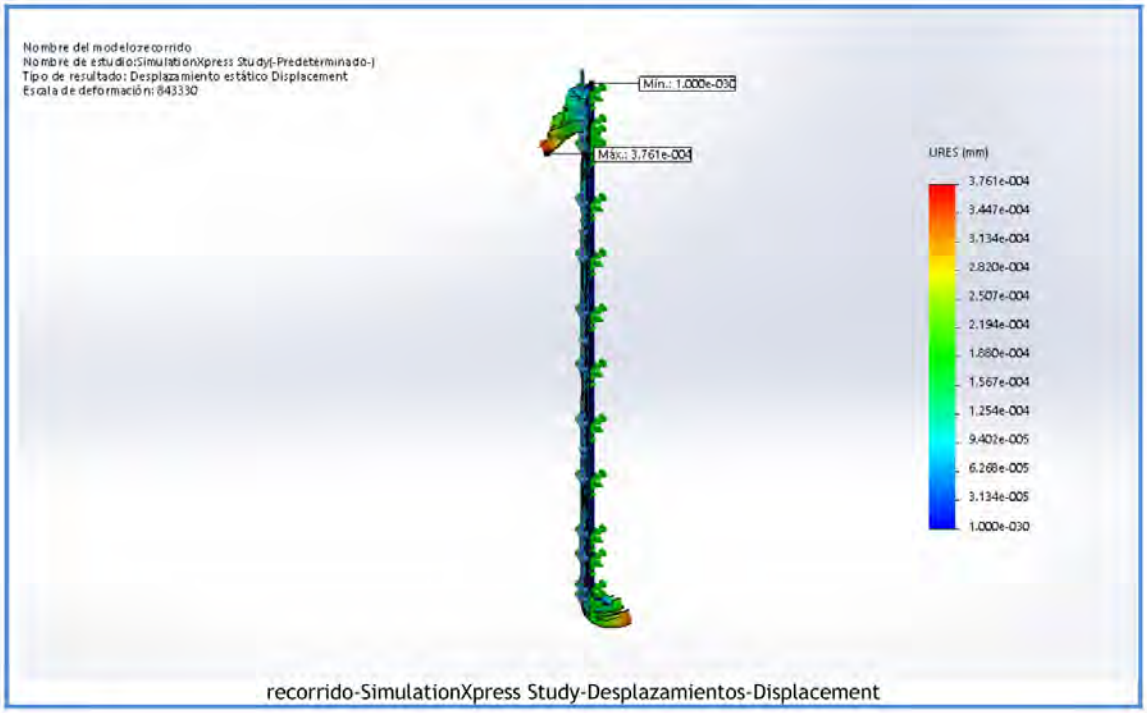
Resultados del estudio

Nombre	Tipo	Mín.	Máx.
Stress	VON: Tensión de von Mises	5.670e+000N/m^2 Nodo: 100712	4.937e+005N/m^2 Nodo: 48710



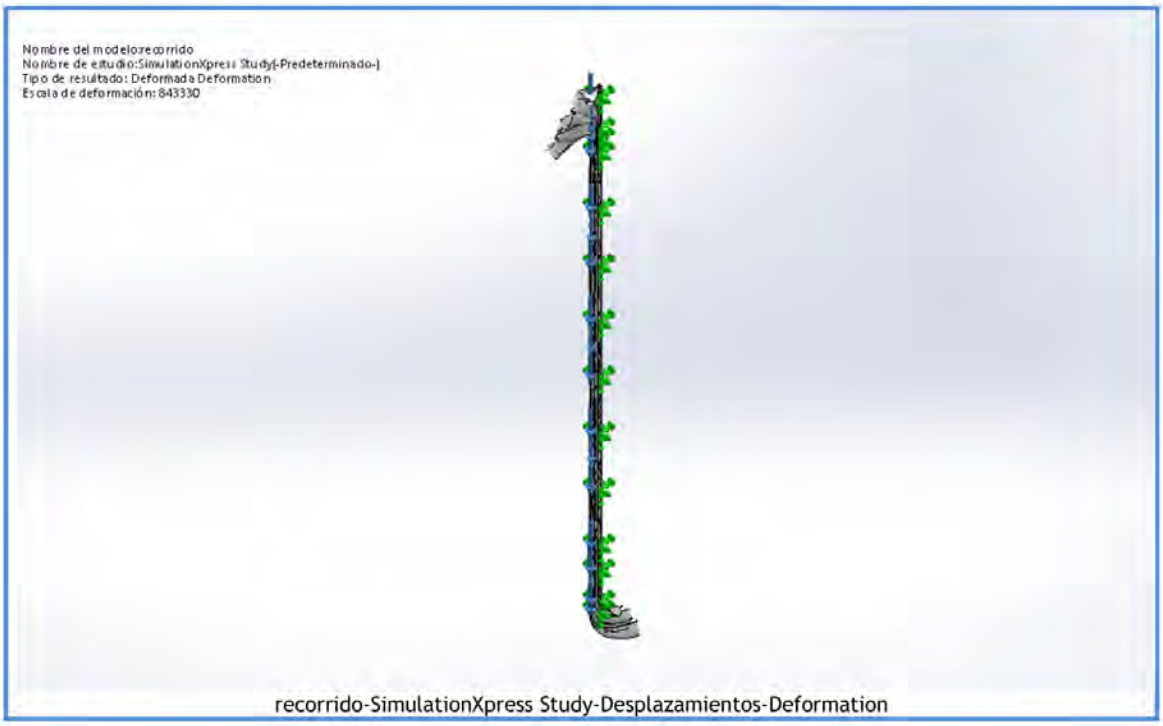
recorrido-SimulationXpress Study-Tensiones-Stress

Nombre	Tipo	Mín.	Máx.
Displacement	URES: Desplazamientos resultantes	0.000e+000mm Nodo: 4053	3.761e-004mm Nodo: 7154



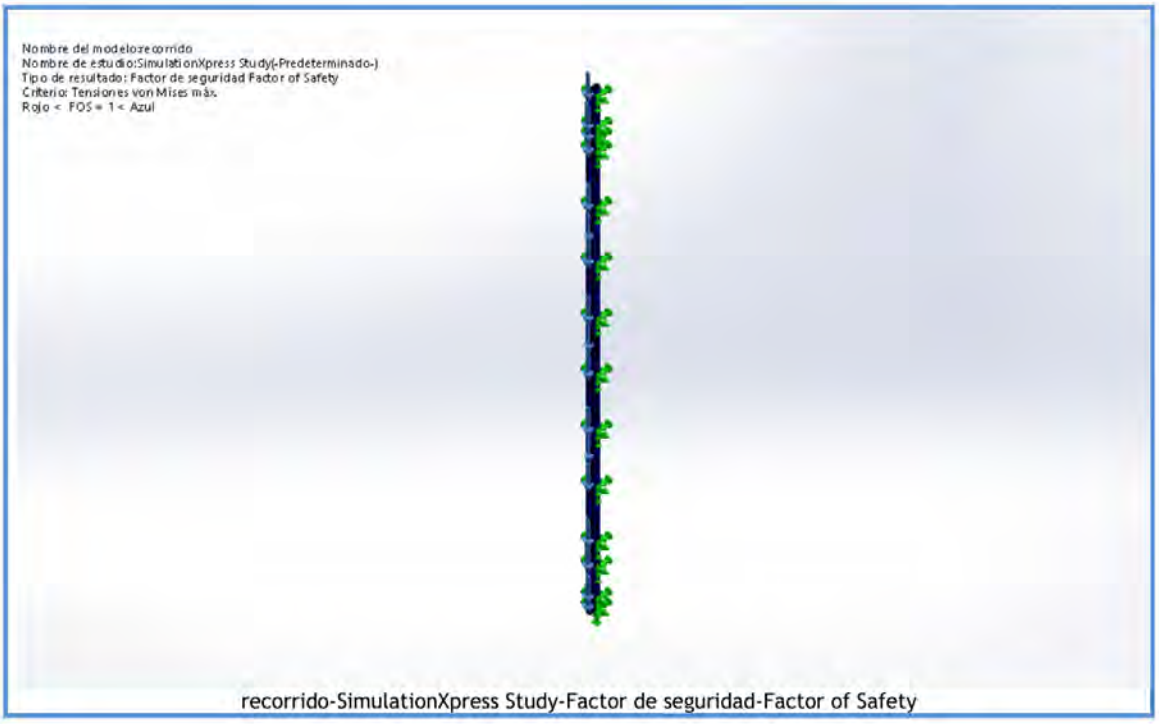
recorrido-SimulationXpress Study-Desplazamientos-Displacement

Nombre	Tipo
Deformation	Deformada



recorrido-SimulationXpress Study-Desplazamientos-Deformation

Nombre	Tipo	Mín.	Máx.
Factor of Safety	Tensión de von Mises máx.	5.586e+001 Nodo: 48710	4.863e+006 Nodo: 100712



recorrido-SimulationXpress Study-Factor de seguridad-Factor of Safety

Simulación de acople_induccion

Fecha: miércoles, 13 de febrero de 2019
Diseñador: Solidworks
Nombre de estudio: SimulationXpress Study
Tipo de análisis: Análisis estático

Tabla de contenidos

Descripción

Análisis de esfuerzos acople inducción

Suposiciones

Información de modelo

Nombre del modelo: acople_induccion
Configuración actual: Predeterminado

Sólidos

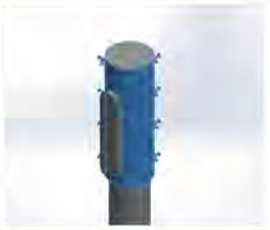
Nombre de documento y referencia	Tratado como	Propiedades volumétricas	Ruta al documento/Fecha de modificación

Rosca1 	Sólido	**Masa:0.383196 kg** **Volumen:4.85058e-005 m^3** **Densidad:7900 kg/m^3** **Peso:3.75532 N**	**C:\Users\Kevin\Desktop\motor a induccion\acople_induccion.SLDPRT** Feb 13 14:12:04 2019

Propiedades de material

Referencia de modelo	Propiedades	Componentes
	**Nombre:** AISI 1020 **Tipo de modelo:** Isotrópico elástico lineal **Criterio de error predeterminado:** Desconocido **Límite elástico:** 3.51571e+008 N/m^2 **Límite de tracción:** 4.20507e+008 N/m^2	Sólido 1(Rosca1)(acople_induccion)

Cargas y sujeciones

Nombre de sujeción	Imagen de sujeción	Detalles de sujeción
Fijo-1		**Entidades:** 1 cara(s) **Tipo:** Geometría fija

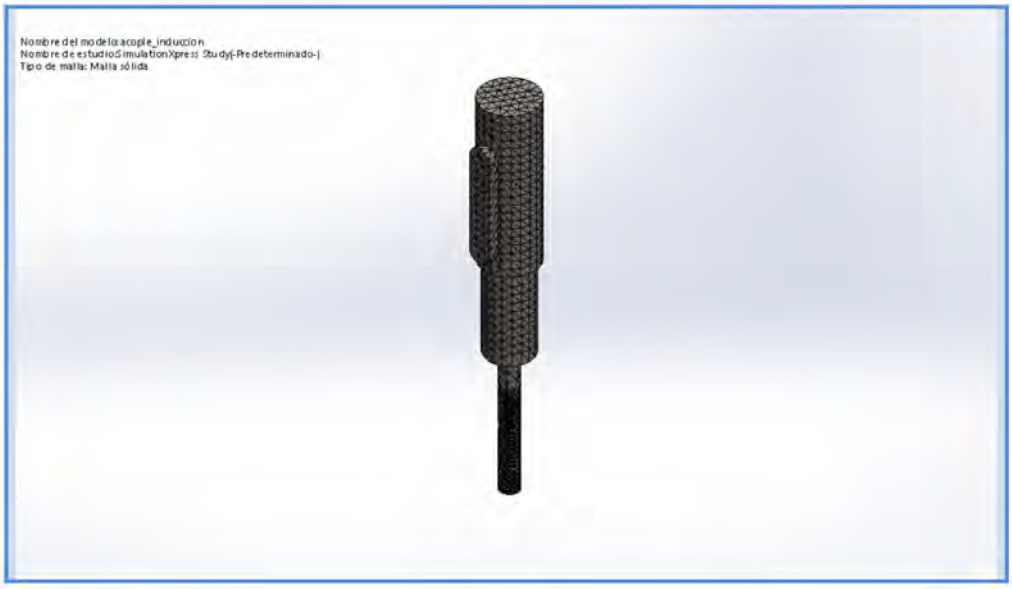
Nombre de carga	Cargar imagen	Detalles de carga
Fuerza-1		**Entidades:** 1 cara(s) **Tipo:** Aplicar fuerza normal **Valor:** 3 N

Información de malla

Tipo de malla	Malla sólida
Mallador utilizado:	Malla estándar
Transición automática:	Desactivar
Incluir bucles automáticos de malla:	Desactivar
Puntos jacobianos	4 Puntos
Tamaño de elementos	3.64811 mm
Tolerancia	0.182405 mm
Trazado de calidad de malla	Elementos cuadráticos de alto orden

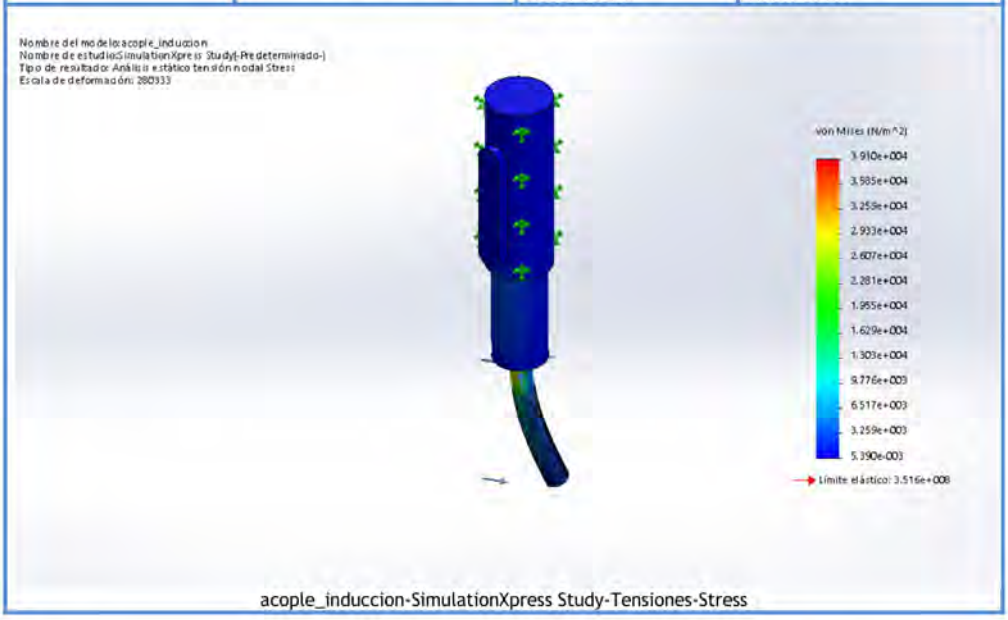
Información de malla - Detalles

Número total de nodos	31169
Número total de elementos	19464
Cociente máximo de aspecto	56.23
% de elementos cuyo cociente de aspecto es < 3	64.3
% de elementos cuyo cociente de aspecto es > 10	5.07
% de elementos distorsionados (Jacobiana)	0
Tiempo para completar la malla (hh;mm;ss):	00:00:48
Nombre de computadora:	


Nombre del modelo: acople_induccion
Nombre de estudio: SimulationXpress Study(-Predeterminado-)
Tipo de malla: Malla sólida

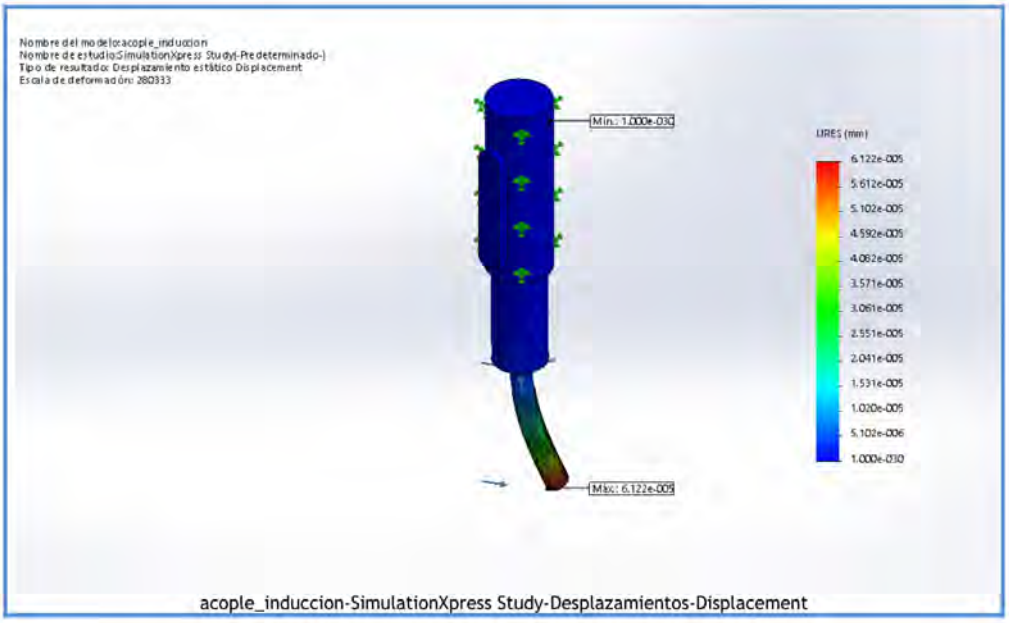
Resultados del estudio

Nombre	Tipo	Mín.	Máx.
Stress	VON: Tensión de von Mises	5.390e-003N/m^2 Nodo: 15578	3.910e+004N/m^2 Nodo: 30180



acople_induccion-SimulationXpress Study-Tensiones-Stress

Nombre	Tipo	Mín.	Máx.
Displacement	URES: Desplazamientos resultantes	0.000e+000mm Nodo: 1759	6.122e-005mm Nodo: 2222



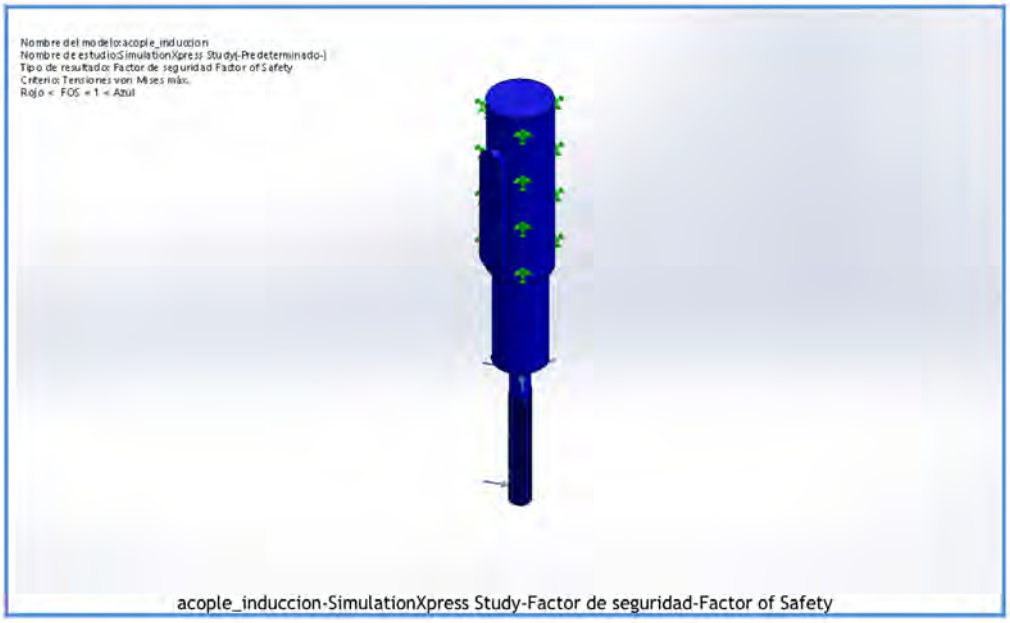
acople_induccion-SimulationXpress Study-Desplazamientos-Displacement

Nombre	Tipo
Deformation	Deformada

Nombre del modelo:acople_induccion
Nombre de estudio:Simulation Xpress Study(-Predeterminado-)
Tipo de resultado: Deformada Deformation
Escala de deformación: 280333

acople_induccion-SimulationXpress Study-Desplazamientos-Deformation

Nombre	Tipo	Mín.	Máx.
Factor of Safety	Tensión de von Mises máx.	8.991e+003 Nodo: 30180	6.523e+010 Nodo: 15578



acople_induccion-SimulationXpress Study-Factor de seguridad-Factor of Safety

Simulación de paredmov

Fecha: miércoles, 13 de febrero de 2019
Diseñador: Solidworks
Nombre de estudio: SimulationXpress Study
Tipo de análisis: Análisis estático

Tabla de contenidos

Descripción

Análisis de esfuerzos estructura de sujeción a la pared

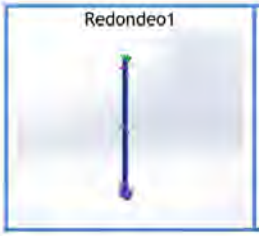
Suposiciones

Información de modelo

Nombre del modelo: paredmov
Configuración actual: Predeterminado

Sólidos

Nombre de documento y referencia	Tratado como	Propiedades volumétricas	Ruta al documento/Fecha de modificación

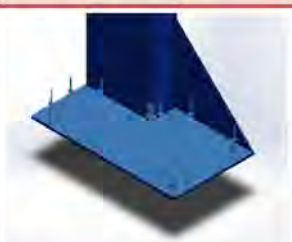
Redondeo1 	Sólido	Masa:25.3154 kg Volumen:0.00320448 m^3 Densidad:7900 kg/m^3 Peso:248.091 N	C:\Users\Kevin\Desktop\motor a induccion\paredmov.SLDPRT Feb 12 17:02:45 2019

Propiedades de material

Referencia de modelo	Propiedades	Componentes
	Nombre: AISI 1020 Tipo de modelo: Isotrópico elástico lineal Criterio de error predeterminado: Tensión de von Mises máx. Límite elástico: 3.51571e+008 N/m^2 Límite de tracción: 4.20507e+008 N/m^2	Sólido 1(Redondeo1)(paredmov)

Cargas y sujeciones

Nombre de sujeción	Imagen de sujeción	Detalles de sujeción
Fijo-1		**Entidades:** 4 cara(s) **Tipo:** Geometría fija

Nombre de carga	Cargar imagen	Detalles de carga
Fuerza-1		**Entidades:** 1 cara(s) **Tipo:** Aplicar fuerza normal **Valor:** 100 N
Fuerza-2		**Entidades:** 1 cara(s) **Tipo:** Aplicar fuerza normal **Valor:** 10 N

Fuerza-3		**Entidades:** 1 cara(s) **Tipo:** Aplicar fuerza normal **Valor:** 10 N
Fuerza-4		**Entidades:** 1 cara(s) **Tipo:** Aplicar fuerza normal **Valor:** 10 N
Fuerza-5		**Entidades:** 1 cara(s) **Tipo:** Aplicar fuerza normal **Valor:** 10 N
Fuerza-6		**Entidades:** 1 cara(s) **Tipo:** Aplicar fuerza normal **Valor:** 10 N

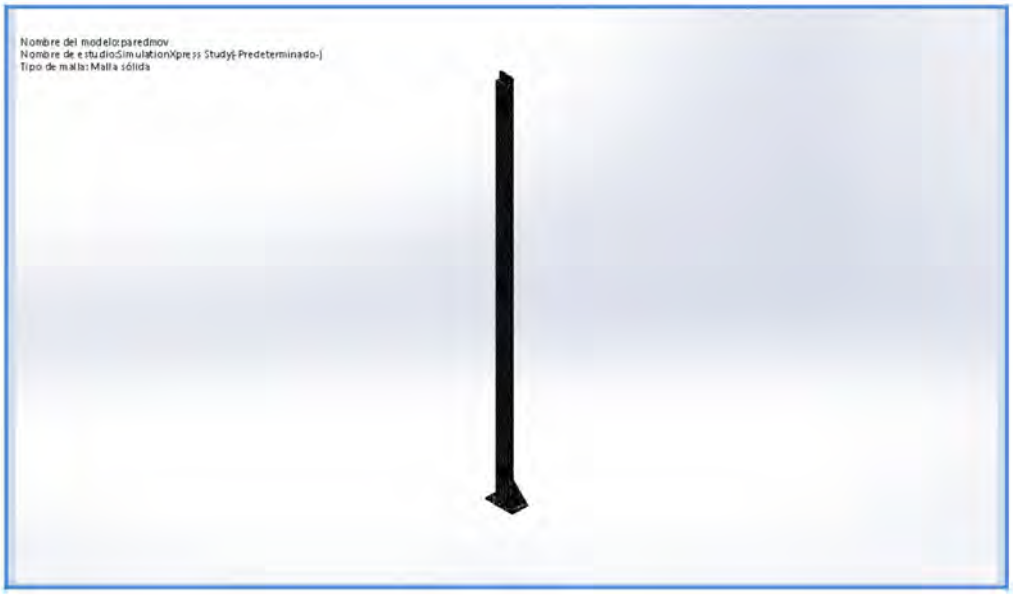
Fuerza-7		**Entidades:** 1 cara(s) **Tipo:** Aplicar fuerza normal **Valor:** 10 N

Información de malla

Tipo de malla	Malla sólida
Mallador utilizado:	Malla estándar
Transición automática:	Desactivar
Incluir bucles automáticos de malla:	Desactivar
Puntos jacobianos	4 Puntos
Tamaño de elementos	25.3188 mm
Tolerancia	1.26594 mm
Trazado de calidad de malla	Elementos cuadráticos de alto orden

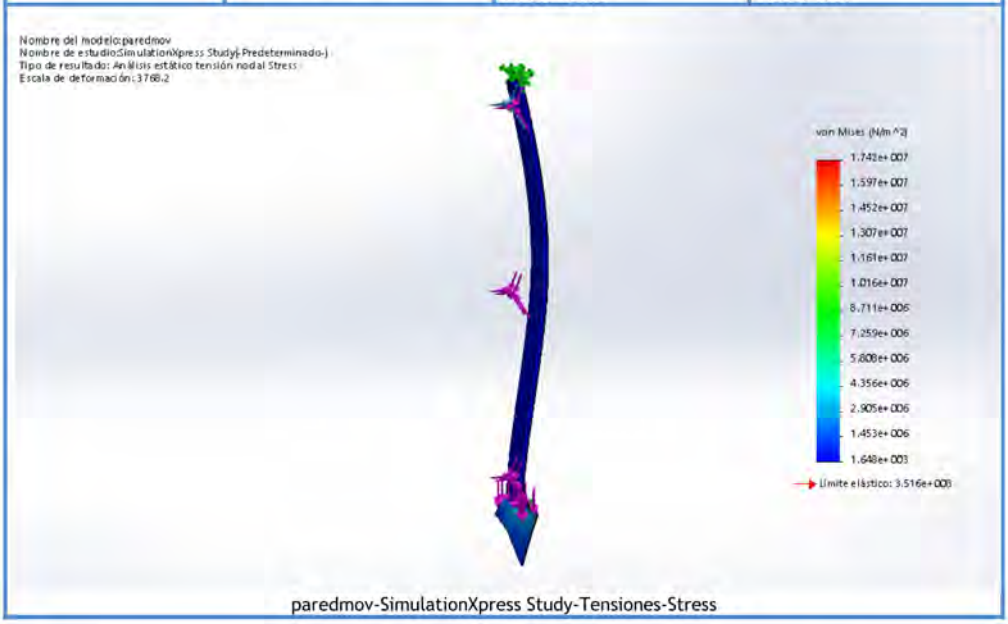
Información de malla - Detalles

Número total de nodos	21300
Número total de elementos	10598
Cociente máximo de aspecto	31.639
% de elementos cuyo cociente de aspecto es < 3	9.77
% de elementos cuyo cociente de aspecto es > 10	7.96
% de elementos distorsionados (Jacobiana)	0
Tiempo para completar la malla (hh;mm;ss):	00:00:03
Nombre de computadora:	


Nombre del modelo:paredmov
Nombre de estudio:SimulationXpress Study(-Predeterminado-)
Tipo de malla: Malla sólida

Resultados del estudio

Nombre	Tipo	Mín.	Máx.
Stress	VON: Tensión de von Mises	1.648e+003N/m^2 Nodo: 16383	1.742e+007N/m^2 Nodo: 3587

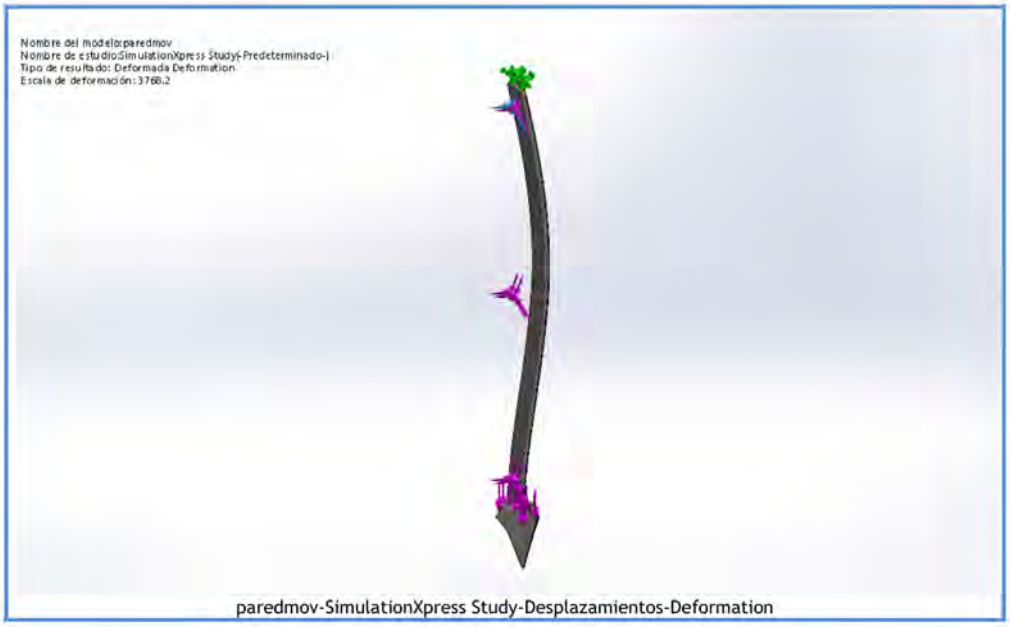


paredmov-SimulationXpress Study-Tensiones-Stress

Nombre	Tipo	Mín.	Máx.
Displacement	URES: Desplazamientos resultantes	0.000e+000mm Nodo: 73	7.879e-002mm Nodo: 4287

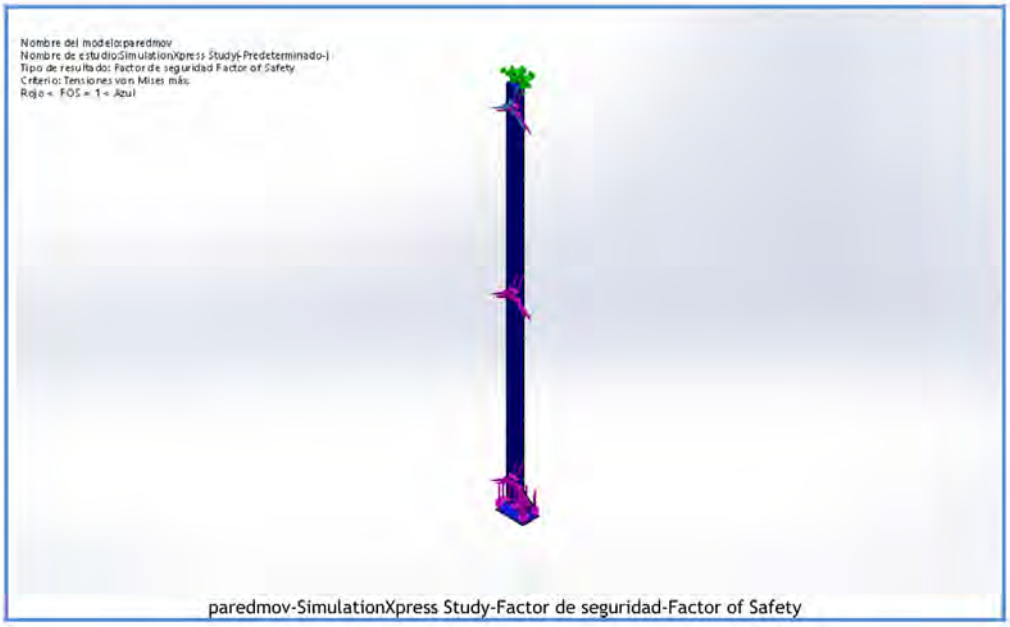
paredmov-SimulationXpress Study-Desplazamientos-Displacement

Nombre	Tipo
Deformation	Deformada



paredmov-SimulationXpress Study-Desplazamientos-Deformation

Nombre	Tipo	Mín.	Máx.
Factor of Safety	Tensión de von Mises máx.	2.018e+001 Nodo: 3587	2.133e+005 Nodo: 16383



paredmov-SimulationXpress Study-Factor de seguridad-Factor of Safety

Simulación de acople_servo

Fecha: miércoles, 13 de febrero de 2019
Diseñador: Solidworks
Nombre de estudio: SimulationXpress Study
Tipo de análisis: Análisis estático

Tabla de contenidos

Descripción

Análisis de esfuerzos acople servomotor

Suposiciones

Información de modelo

Nombre del modelo: acople_servo
Configuración actual: Predeterminado

Sólidos

Nombre de documento y referencia	Tratado como	Propiedades volumétricas	Ruta al documento/Fecha de modificación

Saliente-Extruir3 	Sólido	Masa:0.157659 kg Volumen:1.99568e-005 m^3 Densidad:7900 kg/m^3 Peso:1.54506 N	C:\Users\Kevin\Desktop\motor a induccion\acople_servo.SLDPRT Feb 13 13:52:43 2019

Propiedades de material

Referencia de modelo	Propiedades	Componentes
	**Nombre:** AISI 1020 **Tipo de modelo:** Isotrópico elástico lineal **Criterio de error predeterminado:** Desconocido **Límite elástico:** 3.51571e+008 N/m^2 **Límite de tracción:** 4.20507e+008 N/m^2	Sólido 1(Saliente-Extruir3)(acople_servo)

Cargas y sujeciones

Nombre de sujeción	Imagen de sujeción	Detalles de sujeción
Fijo-1		**Entidades:** 1 cara(s) **Tipo:** Geometría fija

Nombre de carga	Cargar imagen	Detalles de carga
Fuerza-1		**Entidades:** 1 cara(s) **Tipo:** Aplicar fuerza normal **Valor:** 3 N

Información de malla

Tipo de malla	Malla sólida
Mallador utilizado:	Malla estándar
Transición automática:	Desactivar
Incluir bucles automáticos de malla:	Desactivar
Puntos jacobianos	4 Puntos
Tamaño de elementos	2.7134 mm
Tolerancia	0.13567 mm
Trazado de calidad de malla	Elementos cuadráticos de alto orden

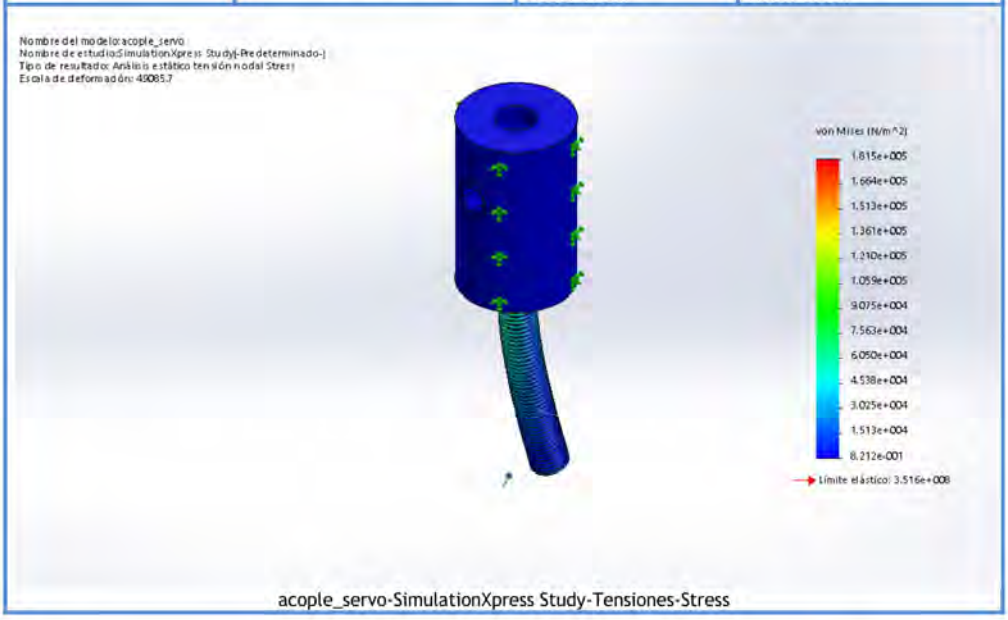
Información de malla - Detalles

Número total de nodos	30837
Número total de elementos	19273
Cociente máximo de aspecto	95.613
% de elementos cuyo cociente de aspecto es < 3	77.7
% de elementos cuyo cociente de aspecto es > 10	0.202
% de elementos distorsionados (Jacobiana)	0
Tiempo para completar la malla (hh;mm;ss):	00:00:49
Nombre de computadora:	

Nombre del modelo: acople_servo
Nombre de estudio SimulationXpress Study(-Predeterminado-)
Tipo de malla: Malla sólida

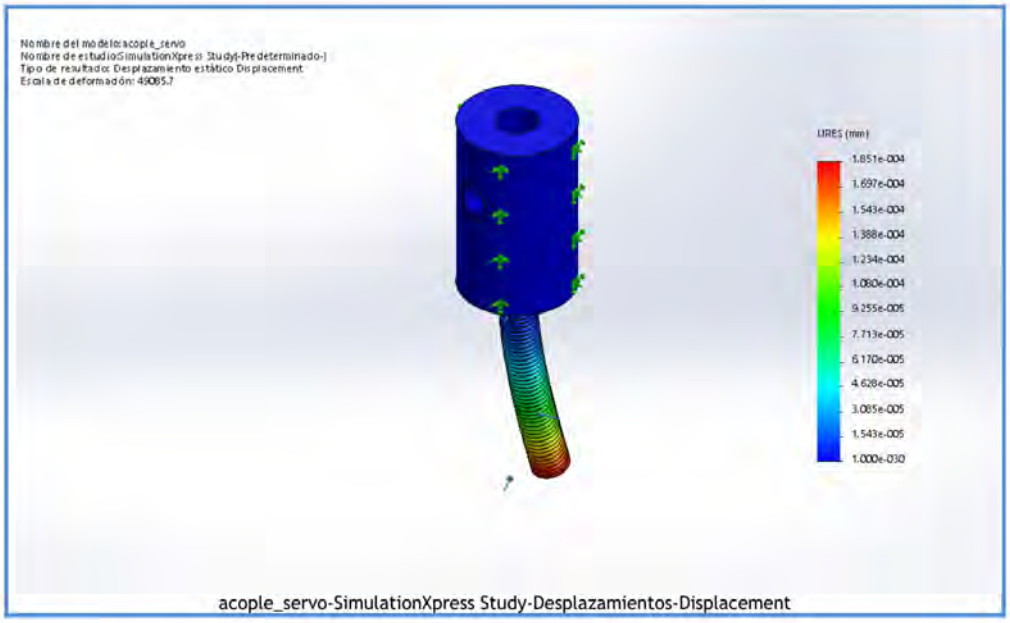
Resultados del estudio

Nombre	Tipo	Mín.	Máx.
Stress	VON: Tensión de von Mises	8.212e-001N/m^2 Nodo: 19825	1.815e+005N/m^2 Nodo: 30651



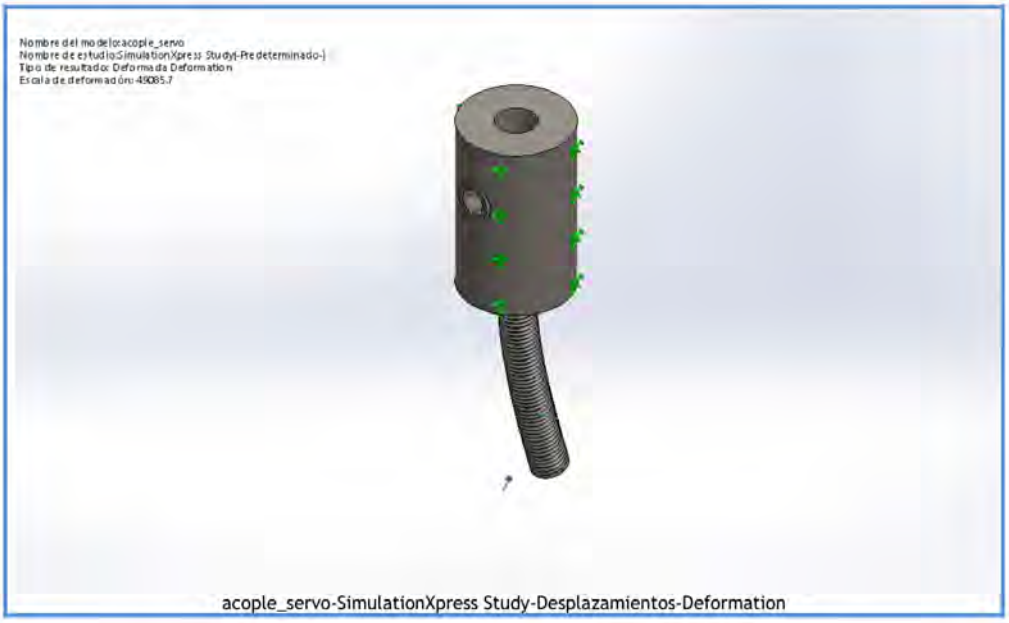
acople_servo-SimulationXpress Study-Tensiones-Stress

Nombre	Tipo	Mín.	Máx.
Displacement	URES: Desplazamientos resultantes	0.000e+000mm Nodo: 1787	1.851e-004mm Nodo: 1908



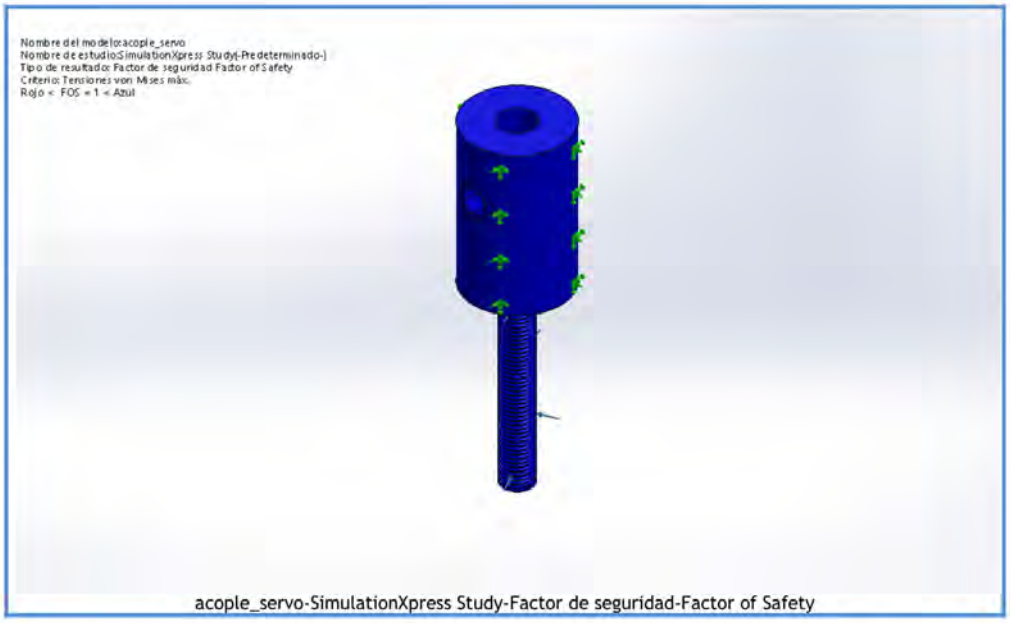
acople_servo-SimulationXpress Study-Desplazamientos-Displacement

Nombre	Tipo
Deformation	Deformada



acople_servo-SimulationXpress Study-Desplazamientos-Deformation

Nombre	Tipo	Mín.	Máx.
Factor of Safety	Tensión de von Mises máx.	1.937e+003 Nodo: 30651	4.281e+008 Nodo: 19825



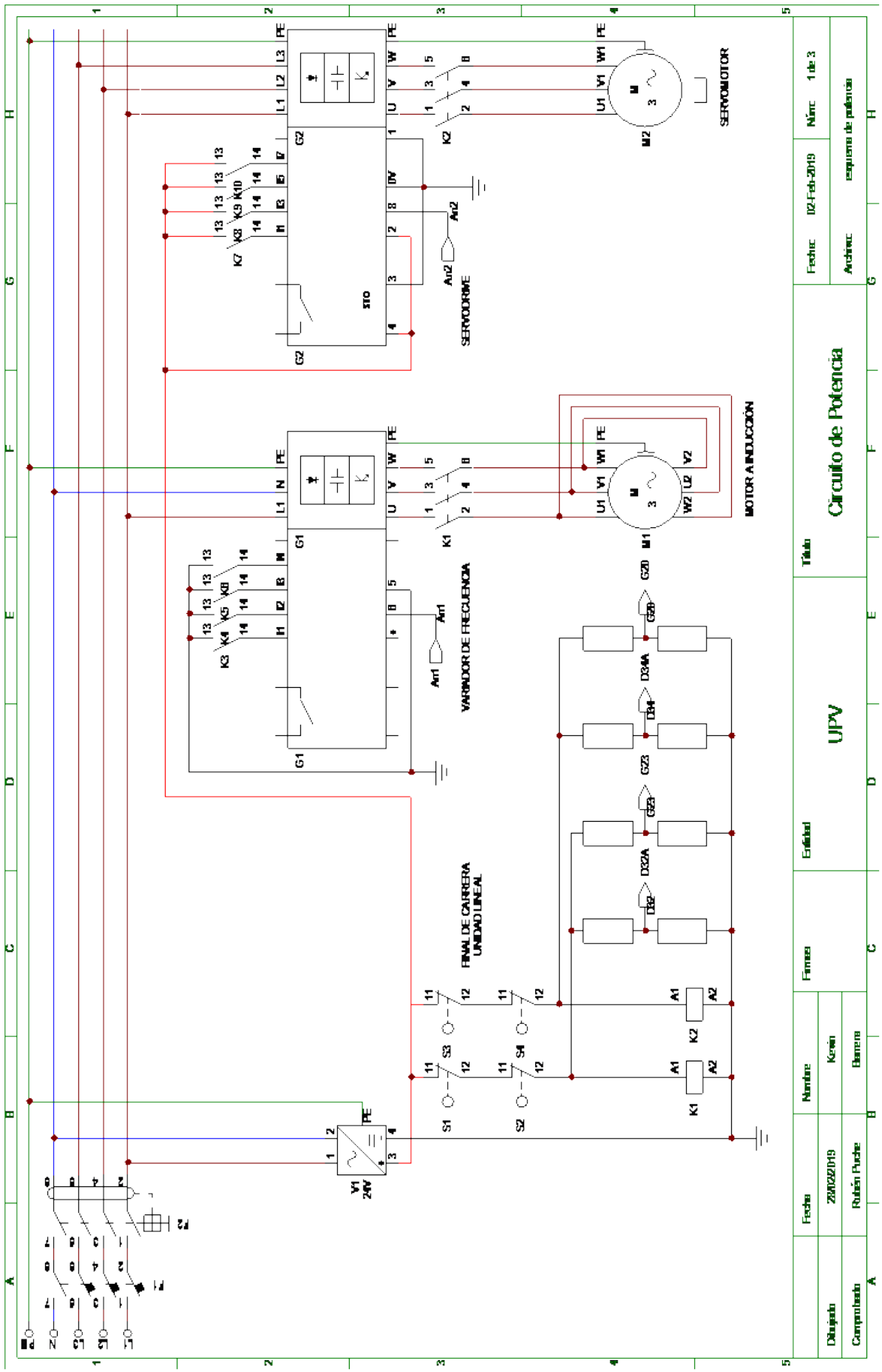
acople_servo-SimulationXpress Study-Factor de seguridad-Factor of Safety

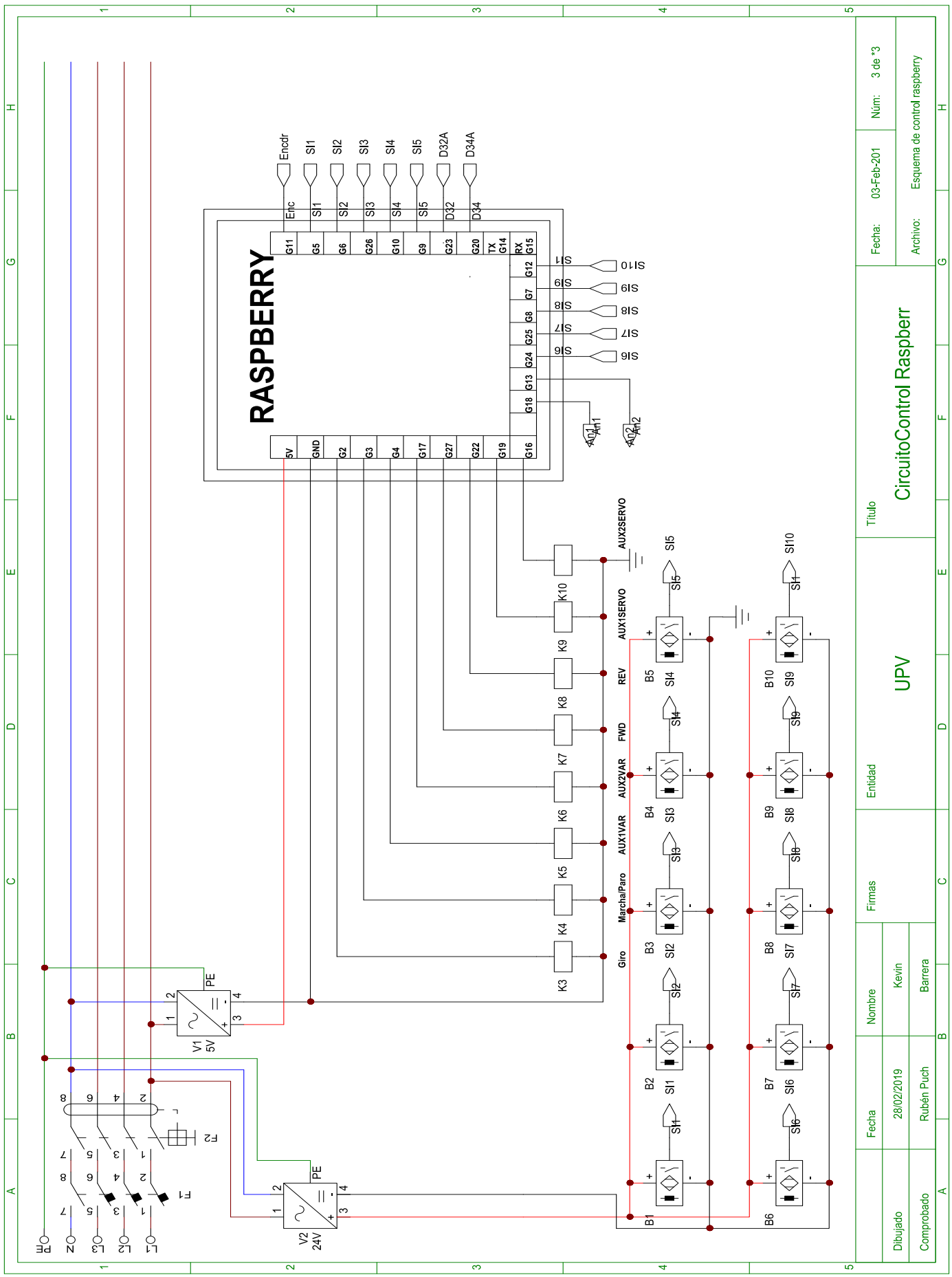
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

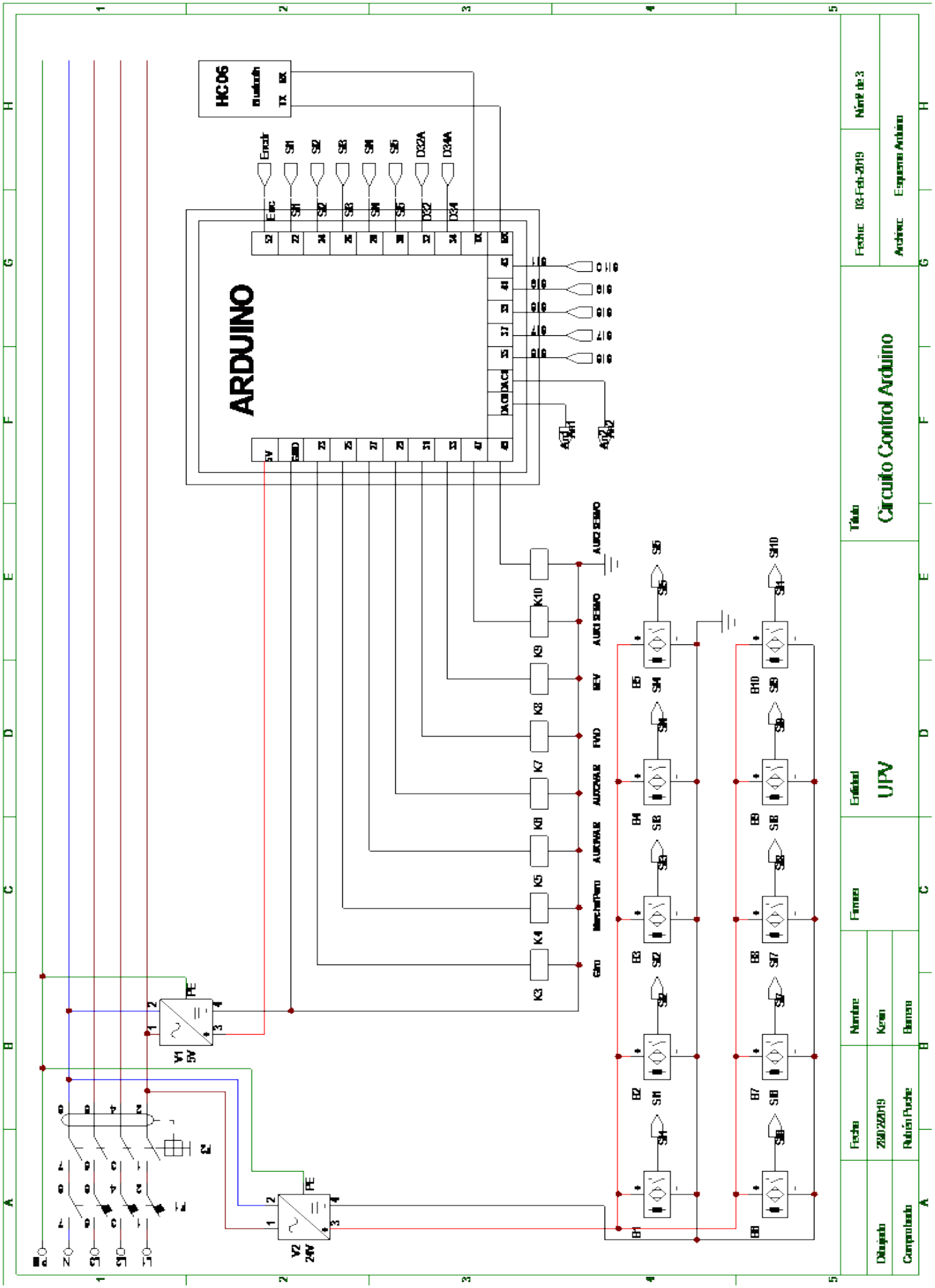
Escuela Técnica Superior de Ingeniería del Diseño

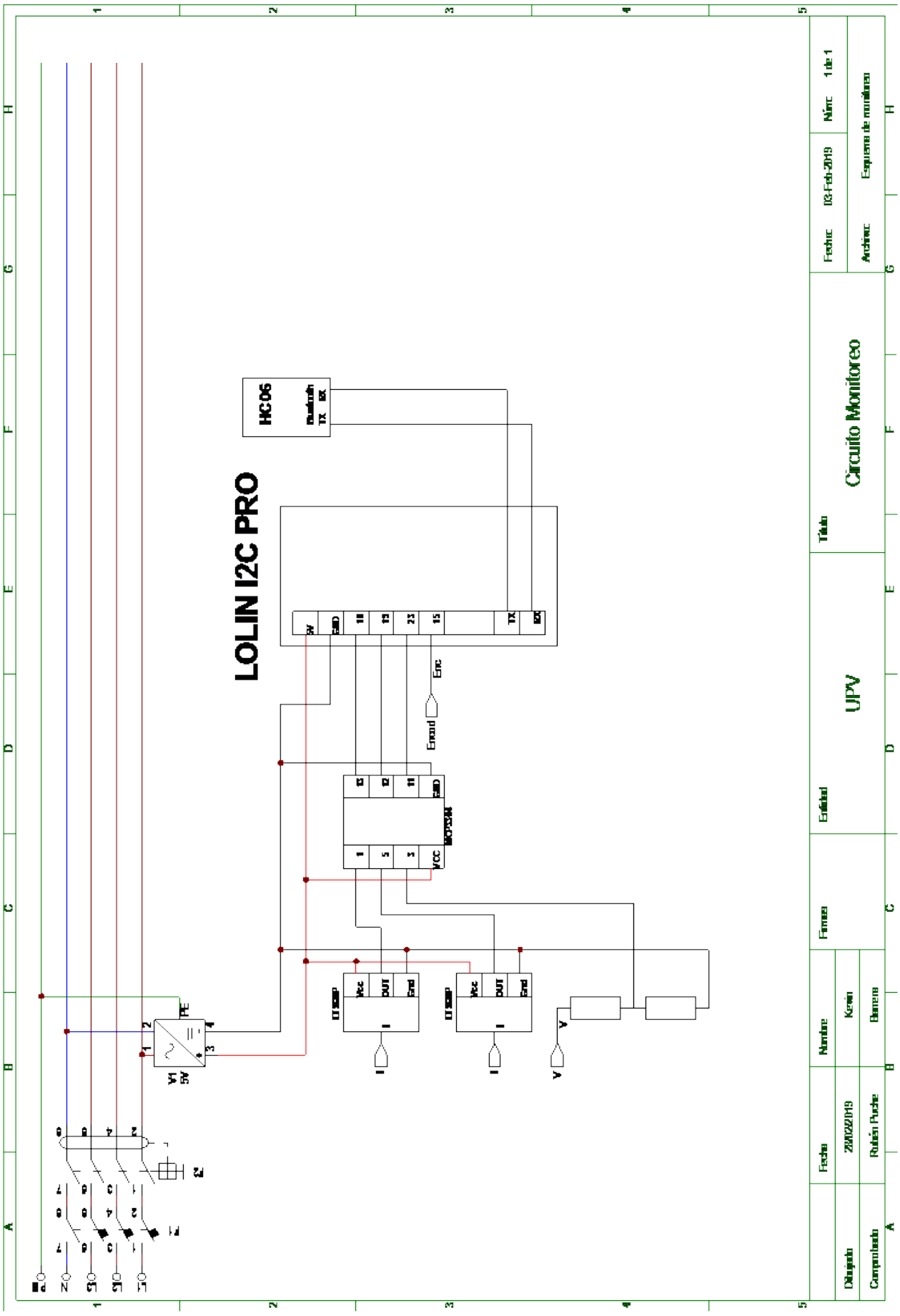
PROTOTIPO DE UN ASCENSOR Y SU CONTROL COMPARANDO TECNOLOGÍAS DIFERENTES DE ACCIONAMIENTO

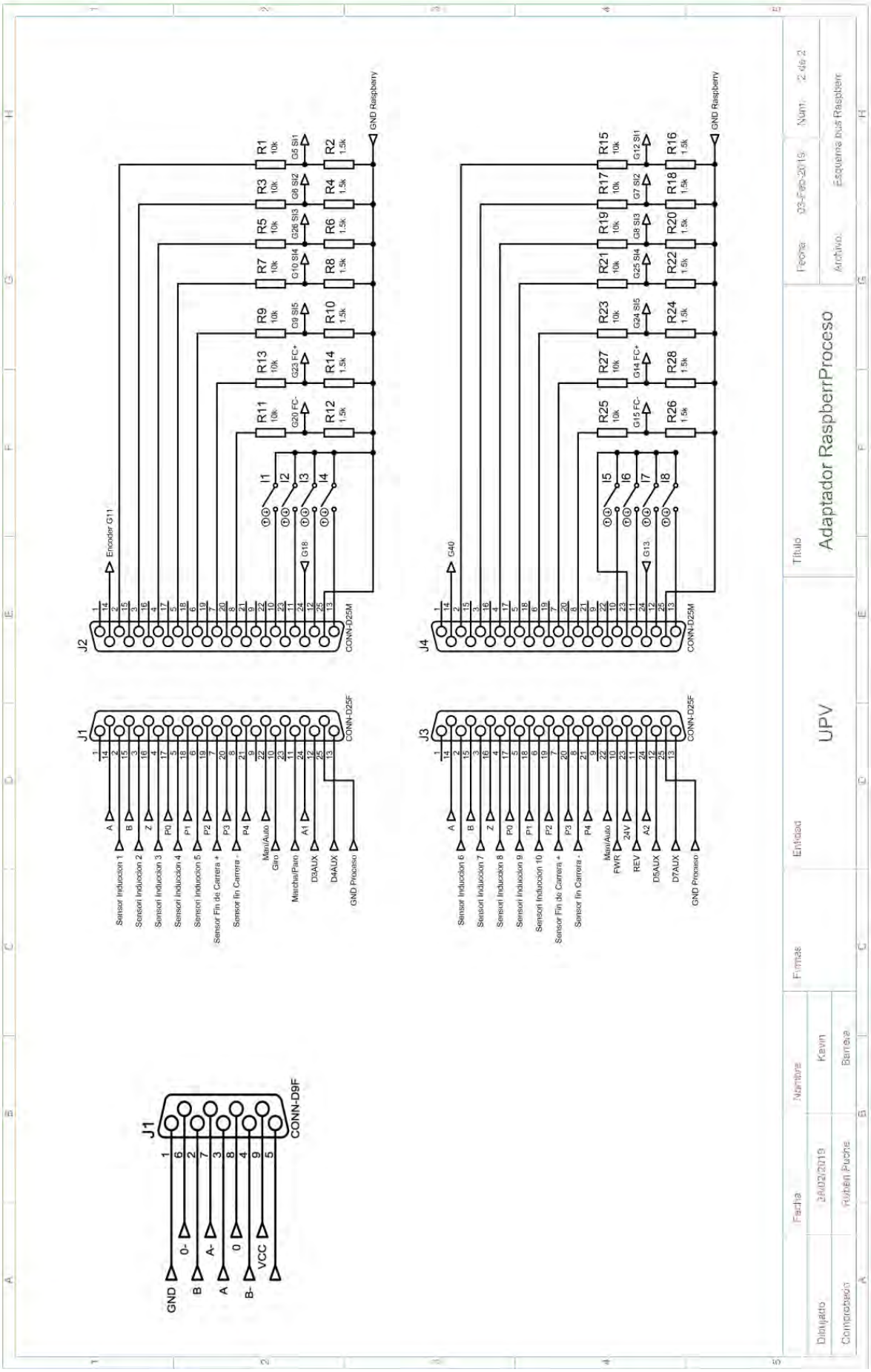
ANEXO III: ESQUEMAS DE CONEXIÓN

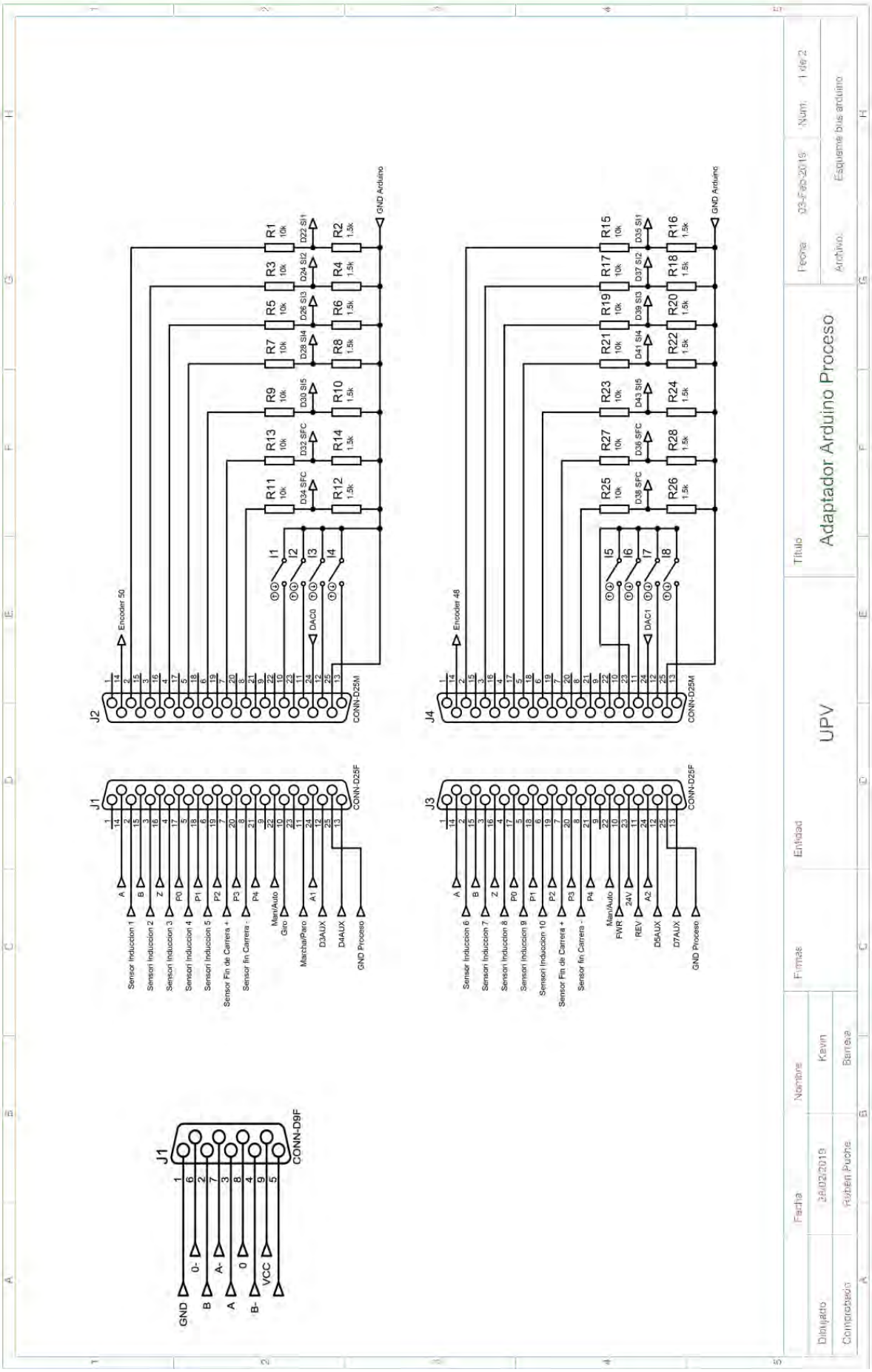

Circuito de Potencia
UPV
SERVOMOTOR
SERVODRIVE
MOTOR A INDUCCIÓN
VARIADOR DE FRECUENCIA
FINAL DE CARRERA UNIDAD LINEAL
Título
Entidad
Firmas
Nombre
Fecha
Dibujado
Comprobado
Robén Puche
23/02/2019
Archivo: esquema de potencia
Fecha: 02-Feb-2019
Núm: 1 de 3


RASPBERRY
Encdr
SI1
SI2
SI3
SI4
SI5
D32A
D34A
Enc
D32
D34
5V
GND
G2
G3
G4
G17
G27
G22
G19
G16
G18
G13
G24
G25
G8
G7
G12
G15
RX
G14
TX
G20
G23
G9
G10
G26
G6
G5
G11
SI10
SI9
SI8
SI7
SI6
SI1
AUX2SERVO
AUX1SERVO
REV
FWD
AUX2VAR
AUX1VAR
Marcha/Paro
Giro
K10
K9
K8
K7
K6
K5
K4
K3
V1
5V
V2
24V
PE
F1
F2
L1
L2
L3
N
PE
B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
Título
CircuitoControl Raspberr
Entidad
UPV
Firmas
Nombre
Kevin
Barrera
Fecha
28/02/2019
Rubén Puch
Dibujado
Comprobado
Fecha: 03-Feb-201
Núm: 3 de *3
Archivo: Esquema de control raspberry


ARDUINO
HC-06
Circuito Control Arduino
UPV
Fecha
29/03/2019
Dibujado
Comprobado
Nombre
Kevin
Rubén Puche
Bianca
Firmas
Entidad
Título
Archivo
Esquema Arduino
Fecha
03-Feb-2019
Hoja 1 de 3


LOLIN I2C PRO
HC06
Circuito Monitoreo
UPV
Título
Archivo:
Fecha:
03-Feb-2019
Núm:
1 de 1
Esquema de montaje
Empleado
Firmas
Nombre
Fecha
Diseñado
Comprobado
Kenan
Blanes
Rubén Puche
28/02/2019


J1
CONN-D9F
GND
B
A
B-
0-
A-
0
VCC
J2
CONN-D25M
Encoder G11
G18
GND Raspberry
J1
CONN-D25F
Sensor Inducción 1
Sensor Inducción 2
Sensor Inducción 3
Sensor Inducción 4
Sensor Inducción 5
Sensor Fin de Carrera +
Sensor fin Carrera -
Mani/Auto
Giro
Marcha/Paro
D3AUX
D4AUX
GND Proceso
J4
CONN-D25M
G40
G13
GND Raspberry
J3
CONN-D25F
Sensor Inducción 6
Sensor Inducción 7
Sensor Inducción 8
Sensor Inducción 9
Sensor Inducción 10
Sensor Fin de Carrera +
Sensor fin Carrera -
Mani/Auto
FWR
24V
REV
D5AUX
D7AUX
GND Proceso
R1 10k
R2 1.5k
R3 10k
R4 1.5k
R5 10k
R6 1.5k
R7 10k
R8 1.5k
R9 10k
R10 1.5k
R11 10k
R12 1.5k
R13 10k
R14 1.5k
R15 10k
R16 1.5k
R17 10k
R18 1.5k
R19 10k
R20 1.5k
R21 10k
R22 1.5k
R23 10k
R24 1.5k
R25 10k
R26 1.5k
R27 10k
R28 1.5k
I1
I2
I3
I4
I5
I6
I7
I8
G5 SI1
G6 SI2
G26 SI3
G10 SI4
G9 SI5
G23 FC+
G20 FC-
G12 SI1
G7 SI2
G8 SI3
G25 SI4
G24 SI5
G14 FC+
G15 FC-
Título
Adaptador RaspberrProceso
Fecha
03-Feb-2019
Núm.
2 de 2
Archivo:
Esquema bus Raspberr
Entidad
UPV
Firmas
Nombre
Fecha
Dibujado
Kevin
26/02/2019
Comprobado
Rubén Pucha
Barrera


J2
Encoder 50
CONN-D25M
GND Arduino
R1 10k
R2 1.5k
D22 SI1
R3 10k
R4 1.5k
D24 SI2
R5 10k
R6 1.5k
D26 SI3
R7 10k
R8 1.5k
D28 SI4
R9 10k
R10 1.5k
D30 SI5
R13 10k
R14 1.5k
D32 SFC
R11 10k
R12 1.5k
D34 SFC
I1
I2
I3
I4
DAC0
J4
Encoder 48
CONN-D25M
GND Arduino
R15 10k
R16 1.5k
D35 SI1
R17 10k
R18 1.5k
D37 SI2
R19 10k
R20 1.5k
D39 SI3
R21 10k
R22 1.5k
D41 SI4
R23 10k
R24 1.5k
D43 SI5
R27 10k
R28 1.5k
D36 SFC
R25 10k
R26 1.5k
D38 SFC
I5
I6
I7
I8
DAC1
J1
CONN-D25F
A
B
Z
P0
P1
P2
P3
P4
Sensor Inducción 1
Sensori Inducción 2
Sensori Inducción 3
Sensori Inducción 4
Sensori Inducción 5
Sensor Fin de Carrera +
Sensor fin Carrera -
Manj/Auto
Giro
Marcha/Paro
A1
D3AUX
D4AUX
GND Proceso
J3
CONN-D25F
Sensor Inducción 6
Sensori Inducción 7
Sensori Inducción 8
Sensori Inducción 9
Sensori Inducción 10
Sensor Fin de Carrera +
Sensor fin Carrera -
Manj/Auto
FWR
24V
REV
A2
D5AUX
D7AUX
GND Proceso
J1
CONN-D9F
GND
B
A
B-
0-
A-
0
VCC
Título
Adaptador Arduino Proceso
Fecha
03-Feb-2019
Num.
1 de 2
Archivo:
Esquema bus arduino
UPV
Entidad
Firmas
Nombre
Fecha
Dibujado
26/02/2019
Kevin
Comprobado
Rubén Pucha
Benlloa

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PROTOTIPO DE UN ASCENSOR Y SU CONTROL COMPARANDO TECNOLOGÍAS DIFERENTES DE ACCIONAMIENTO

ANEXO IV: PARAMETRIZACIÓN DE LOS VARIADORES DE FRECUENCIA

i. Configuración inicial y puesta en marcha del variador de frecuencia

A través del manual se realizan configuraciones iniciales que permitan un óptimo funcionamiento del sistema con el motor asincrónico, a continuación, se describen las instrucciones programadas.

P001: Lectura de referencia de velocidad, esta se encuentra por defecto al encender el variador, en el HMI se visualiza la frecuencia de referencia enviada a través de la aplicación.

P002: Lectura de velocidad de salida del motor.

P005: Lectura de frecuencia de salida del variador.

P007: Lectura de tensión de salida.

P012: Visualización de entradas digitales.

P202: Tipo de control= 0 (control escalar)

P204: Parámetros por defecto= 7 (configuraciones guardadas por el usuario)

P205: Parámetro principal de visualización= 1 (referencia de frecuencia)

P352: Ventilador=1 (Encendido)

PARÁMETROS DEL MOVIMIENTO

P100: Tiempo de aceleración= 1s

P100: Tiempo de desaceleración= 0,5s

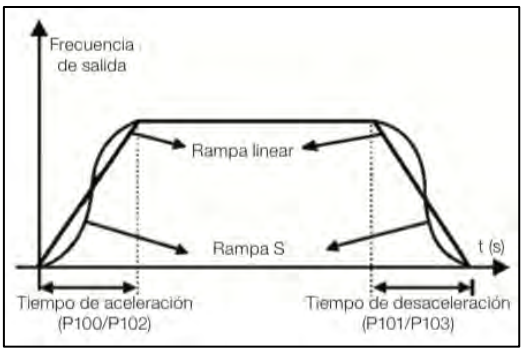


Figura 125 Tiempos de aceleración y desaceleración en el variador de frecuencia

P124: Multispeed 1= 10Hz

P125: Multispeed 2= 20Hz

P126: Multispeed 3= 30Hz

P127: Multispeed 4= 40Hz

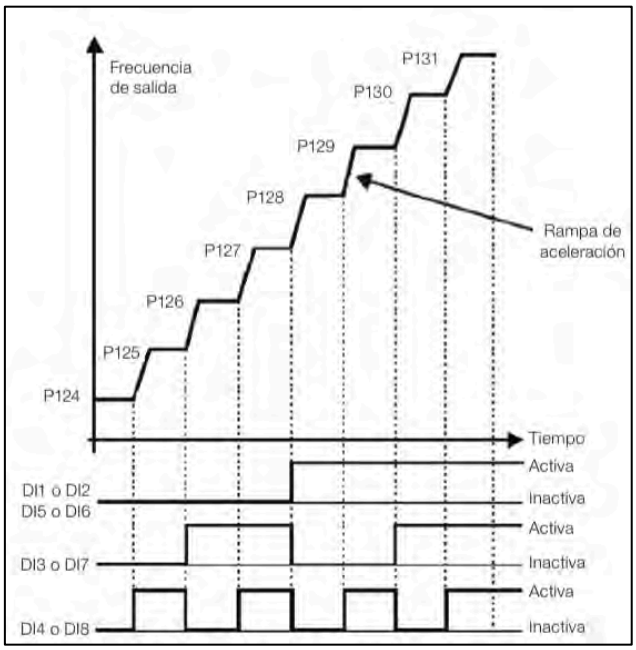


Figura 126 Referencia de frecuencias multispeed

P133: Referencia frecuencia mínima= 0 Hz

P134: Referencia frecuencia máxima= 50 Hz

P140: Compensación de Deslizamiento= 0,5s

P229: Parada del motor= 0 (por rampa)

PARÁMETROS DE CONTROL

P221: Referencia de frecuencia= 1 (Analógica 1)

P223: Sentido de Giro= 0 (horario)

P224: Start-Stop= 1(pin digital)

P231: Entrada Analógica 1= 0 (Referencia de frecuencia)

P232: Ganancia= 4,695 (amplificador de entrada analógica)

P234: Offset: -5% (se toma este valor debido a que la salida DAC del Arduino va de 0.55 a 2.63V)

P263: Digital DI1= 8 (sentido de giro)

P264: Digital DI2= 1 (gira/para)

P265: Digital DI3= 0

P266: Digital DI4= 0

P271: Ajuste NPN=0 (VL>10v= 0 VH<3v= 1)

PARÁMETROS DEL MOTOR

P295: Intensidad nominal del convertidor: 2 A

P296: Tensión Nominal de la red= 2 (220V)

P399: Rendimiento nominal del motor: 67%

P400: Tensión nominal del motor= 230V

P401: Intensidad nominal del motor= 1,1

P402: Velocidad nominal del motor = 1365 rpm

P403: Frecuencia nominal del motor= 50 Hz

P404: Potencia Nominal del motor= 1 (0,25 hp)

P407: Factor de potencia= 0,64

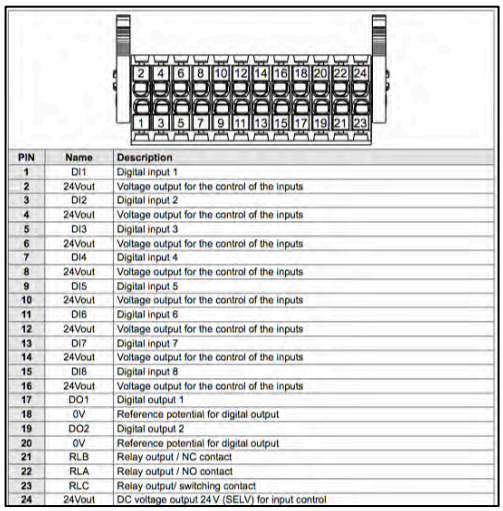
P409: Resistencia del estator= 20,31

Algunas consideraciones a tomarse para el proyecto:

- Los parámetros del motor se toman en cuenta los valores de la placa característica de datos del motor asincrónico a inducción.
- El ventilador del variador de frecuencia se deja encendido por defecto para evitar cortocircuitos por altas temperaturas.
- La referencia de velocidad está dada en hercios.
- La rampa de desaceleración es mucho menor con el fin de que el elevador cuando detecte el sensor se detenga en el punto lo antes posible.
- La tensión de alimentación del variador es monofásica.
- Las entradas digitales D3 Y D4 están conectadas, pero no son utilizadas.
- El valor del offset debe ser considerado en la programación de la Raspberry.
- Todos los parámetros se guardaron en el variador, si se desea cargar la configuración para este proyecto en caso de cambiar algún parámetro dirigirse al parámetro P204 en la opción 7.

ii. Configuración inicial y puesta en marcha del servo drive

Los pines de seguridad STO deben estar alimentados a 24V para que la tarjeta funcione, por lo que se deja conectado los pines en el servo accionamiento. Partiendo de los pines de la siguiente figura se utilizan 2 entradas digitales y una analógica para el control.



PIN	Name	Description
1	DI1	Digital input 1
2	24Vout	Voltage output for the control of the inputs
3	DI2	Digital input 2
4	24Vout	Voltage output for the control of the inputs
5	DI3	Digital input 3
6	24Vout	Voltage output for the control of the inputs
7	DI4	Digital input 4
8	24Vout	Voltage output for the control of the inputs
9	DI5	Digital input 5
10	24Vout	Voltage output for the control of the inputs
11	DI6	Digital input 6
12	24Vout	Voltage output for the control of the inputs
13	DI7	Digital input 7
14	24Vout	Voltage output for the control of the inputs
15	DI8	Digital input 8
16	24Vout	Voltage output for the control of the inputs
17	DO1	Digital output 1
18	0V	Reference potential for digital output
19	DO2	Digital output 2
20	0V	Reference potential for digital output
21	RLB	Relay output / NC contact
22	RLA	Relay output / NO contact
23	RLC	Relay output/ switching contact
24	24Vout	DC voltage output 24 V (SELV) for input control

Figura 127 Pines de Entrada/salida servo accionamiento

A través del manual se realizan configuraciones iniciales que permitan un óptimo funcionamiento del sistema con el motor sincrónico, a continuación, se implementan las siguientes configuraciones.

- **co31:** 32783: SO + EV + /QS + EO + MS15 (permite poner en RUN al servomotor)
- **vl21**: 2000rpm (valor máximo de velocidad servo drive)
- **an30:** ref INPUT AN1 (Referencia de velocidad a través de la entrada analógica)
- **an31:** vl20 (se designa a la entrada A1 a referencia de velocidad)
- **an32:** 0,7324 (porcentaje de amplificación setpoint/4096)
- **an08**: 0 (limite negativo AN1)
- **an09**: 100 (limite positivo AN1)
- **an00**: 0 10V (rango de entrada analógica)
- **di16**: I2 (se designa a I2 como entrada de giro FWD)
- **di17:** I1 (se designa a I2 como entrada de giro REV)

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

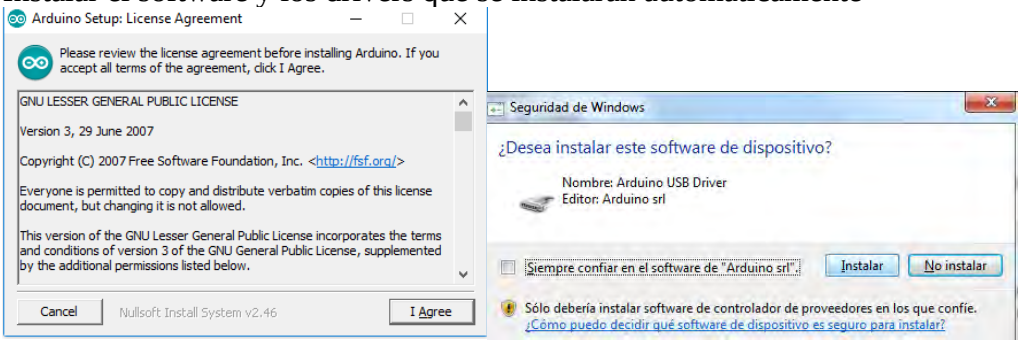
Escuela Técnica Superior de Ingeniería del Diseño

PROTOTIPO DE UN ASCENSOR Y SU CONTROL COMPARANDO TECNOLOGÍAS DIFERENTES DE ACCIONAMIENTO

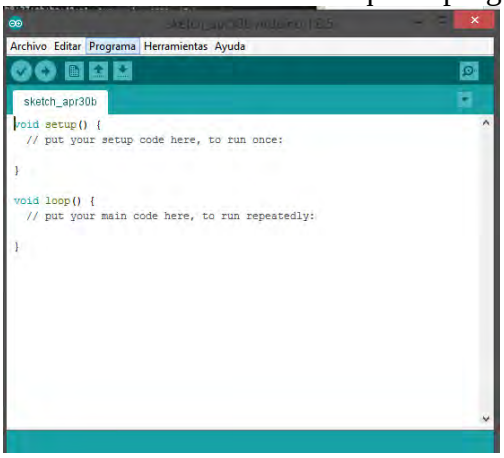
ANEXO V: CONFIGURACIÓN Y PUESTA EN MARCHA ARDUINO

INSTALAR ARDUINO DUE EN PC

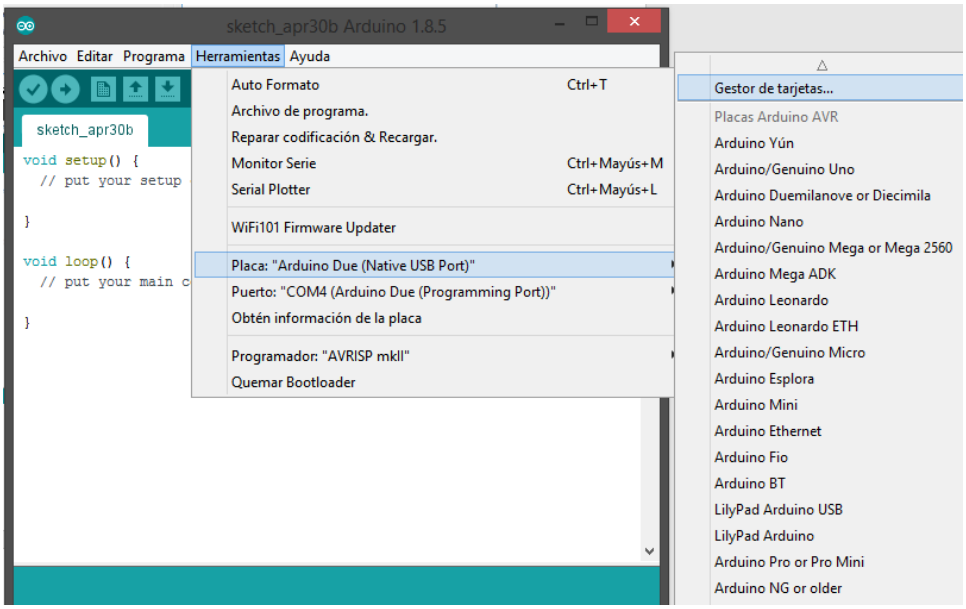
1. Descargar software Arduino de: https://www.arduino.cc/en/Main/Software
2. Instalar el software y los drivers que se instalaran automáticamente



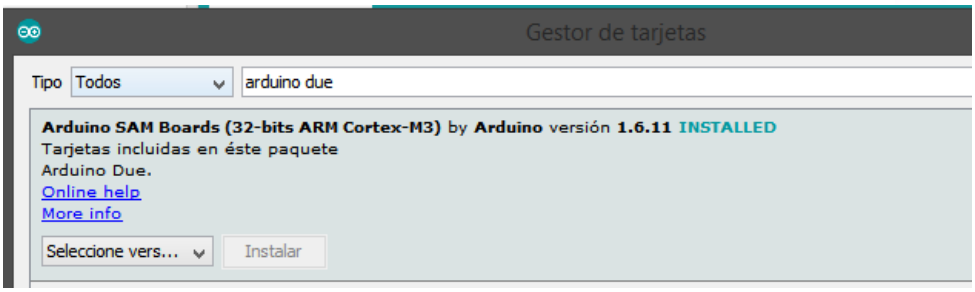
3. Una vez instalado verificar que el programa se abra como en la imagen



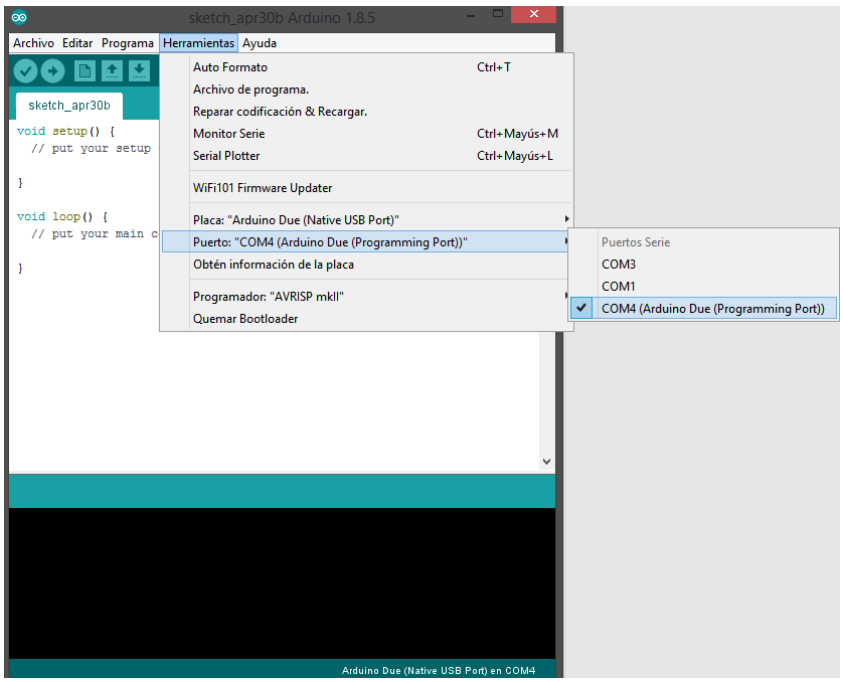
4. Dirigirse al Gestor de tarjetas en la barra de herramientas>>Placa>>Gestor de tarjetas



5. En el buscador colocar “Arduino DUE” e instalar la librería.



6. Verificar que el arduino Due sea reconocido en el puerto COM y en la Placa, ir a Herramientas>>Placa y Herramientas>>Puerto



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

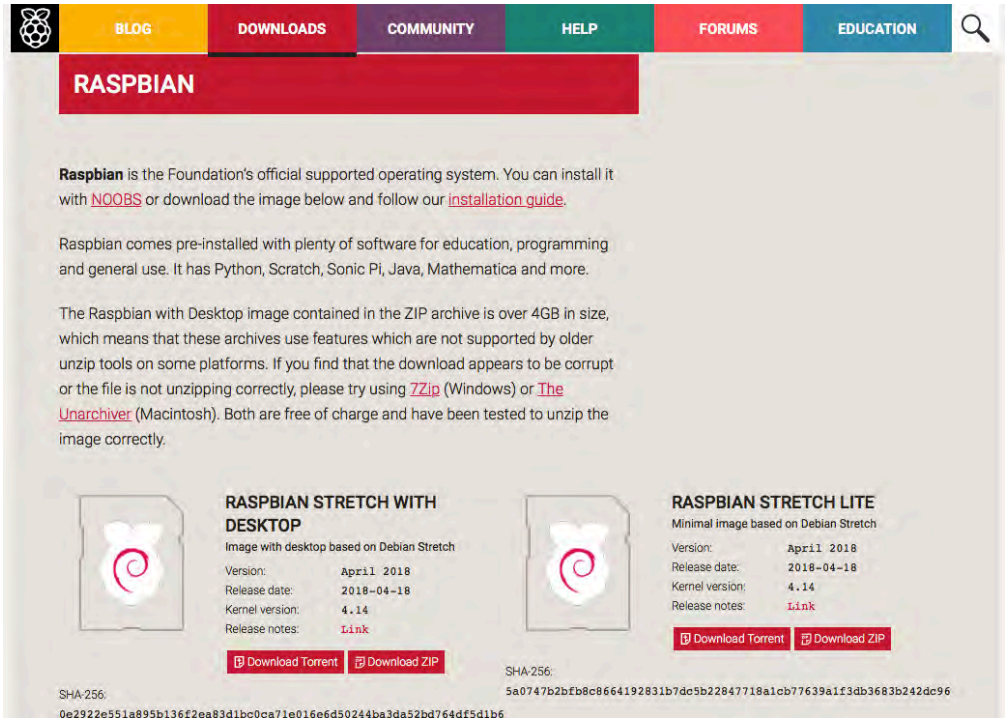
Escuela Técnica Superior de Ingeniería del Diseño

PROTOTIPO DE UN ASCENSOR Y SU CONTROL COMPARANDO TECNOLOGÍAS DIFERENTES DE ACCIONAMIENTO

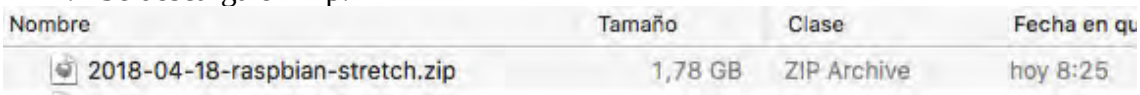
ANEXO VI: CONFIGURACIÓN Y PUESTA EN MARCHA RASPBERRY

1. Página de Descarga:

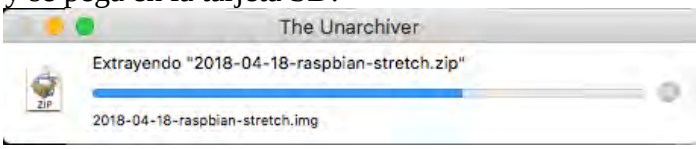
https://www.raspberrypi.org/downloads/raspbian/



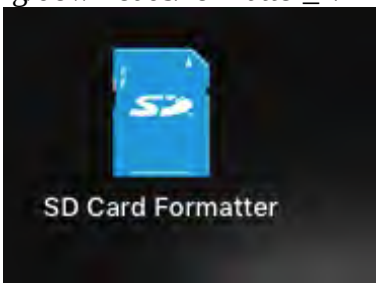
2. Se descarga en Zip:



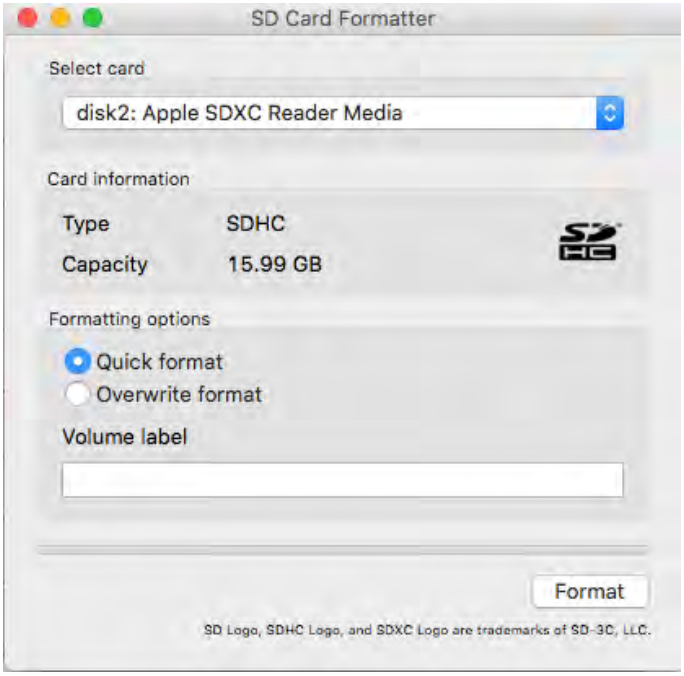
3. Se extrae y se pega en la tarjeta SD:



4. Formateamos la SD ocupando el software de la página: https://www. sdcard.org/downloads/formatter_4/



Formateamos la SD



5. Instalar el sistema operativo
   a) Entramos al terminal en MAC OSX y escribimos: df -h

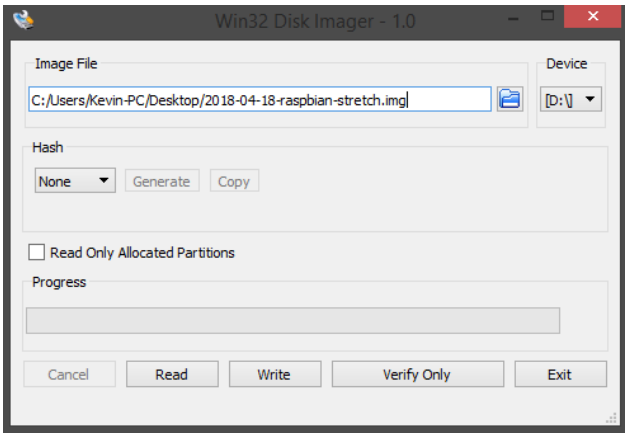
```
wifialu21-249:~ kevinbarrera$ df -h
Filesystem      Size   Used  Avail Capacity iused      ifree %iused  Mounted on
/dev/disk1     509Gi  237Gi  272Gi    47% 1370517 4293596762    0%   /
devfs          183Ki  183Ki    0Bi   100%     634          0  100%   /dev
map -hosts       0Bi    0Bi    0Bi   100%       0          0  100%   /net
map auto_home    0Bi    0Bi    0Bi   100%       0          0  100%   /home
/dev/disk0s4   188Gi  141Gi   47Gi    76% 1058065   48905323    2%   /Volumes/BO
OTCAMP
/dev/disk2s1    15Gi  2.4Mi   15Gi     1%       0          0  100%   /Volumes/RB
```

   Ahí en la última parte vemos el tamaño de la microSD y el nombre disk2
   b) Desmontamos la SD utilizando el comando:
      sudo diskutil unmount /dev/disk2s1
   c) Instalamos el archivo IMG, con la siguiente línea de comandos:
      sudo dd bs=1 if=**/Users/kevinbarrera/Desktop/2018-04-18-raspbian-stretch.img** of=/dev/**disk2**
   En donde en la primera parte colocamos la ubicación del archivo IMG, y en la siguiente colocamos el nombre del dispositivo SD.

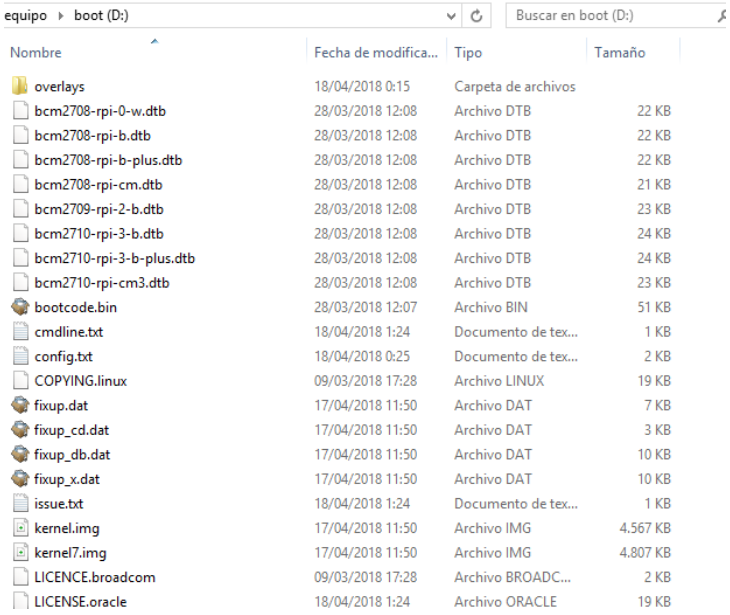
DESDE WINDOWS

1. Repetir los pasos anteriores 1,2 y 3.
2. Descargar Win32DiskImager:
   https://sourceforge.net/projects/win32diskimager/?source=typ_redirect

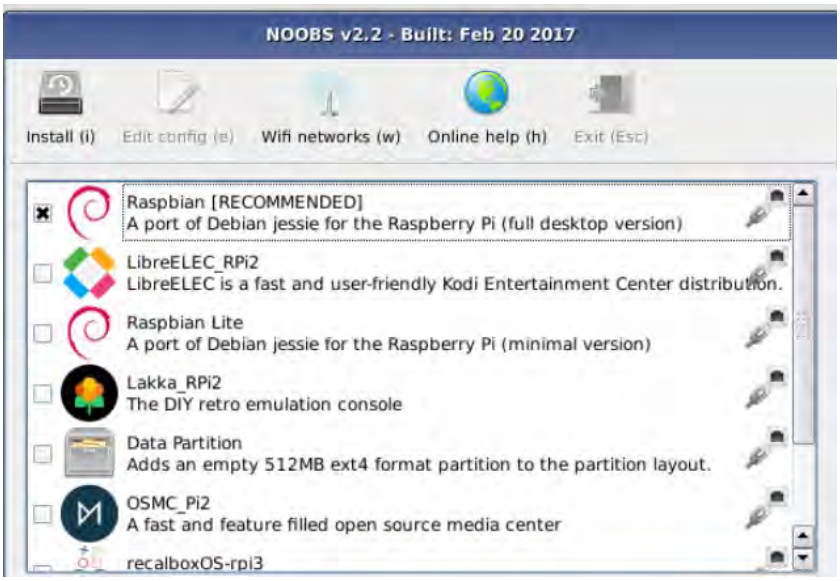
3. Abrir el programa y seleccionar las siguientes configuraciones



4. Al terminar el proceso verificamos que se encuentra instalado el sistema operativo.



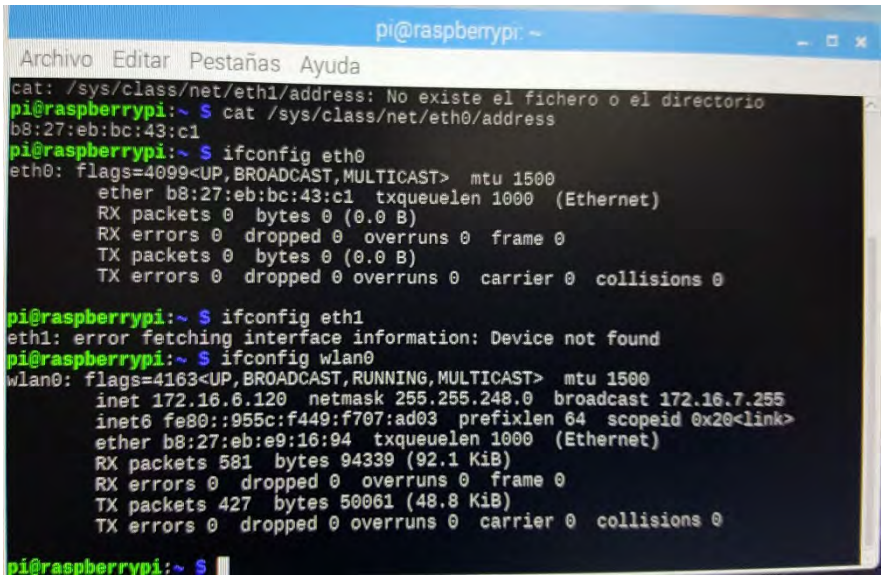
5. Encendemos la raspberry con la SD y aparece la pantalla de instalación, seleccionamos Raspbian, e instalamos.



6. Una vez instalo el SO aparecerá esta imagen y se procederá a configurar la fecha hora y ajustes relacionados con la interfaz.

CODIGOS PARA VER DIRECCION MAC EN RASPBERRY PI 3

Ifconfig eth0: aparece la dirección MAC de ethernet
b8:27:eb:bc:43:c1
Ifconfig wlan0: aparece la dirección MAC del wifi
b8:27:eb:e9:16:94



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PROTOTIPO DE UN ASCENSOR Y SU CONTROL COMPARANDO TECNOLOGÍAS DIFERENTES DE ACCIONAMIENTO

ANEXO VII:

CÓDIGO FUENTE ARDUINO

```
//variables--velocidad y pisos
```

```
int ref_vel;
String palabra;//palabra serial
char letra;//letra serial
int caso=0;
int palabra_entero=25;
int iniciar=0;
int salida_dac=4095;
int alarma_envio=0;
int alarma_envio2=0;
int piso_envio=0;
int piso_envio2=0;
int accion_envio=0;
int accion_envio2=0;
int ok=0;
int ok2=0;
//reles de arduino
//variador de frecuenca pin 27 y 29 salidas de rele3 y 4
//servodrive pin 47 y 49 salidas del rele 7 y 8
const int giro = 23;//PIN GIRO HORARIO Y ANTIHORARIO
const int mov_motor = 25;//PIN MUEVE O APAGA MOTOR ASINCRONICO
const int FWD= 31;//PIN GIRO HORARIO
const int REV= 33;//PIN GIRO ANTIHORARIO
const int sensor_primero = 22;
const int sensor_segundo = 24;
const int sensor_tercero = 26;
const int sensor_cuarto = 28;
const int sensor_quinto = 30;
const int sensor_fin_carrera = 32;
const int sensor_primero2 = 35;
const int sensor_segundo2 = 37;
const int sensor_tercero2 = 39;
const int sensor_cuarto2 = 41;
const int sensor_quinto2 = 43;
const int sensor_fin_carrera2 = 34;
//variables--parte medidor de velocidad
uint32_t pulsos, pulsosF, t_inicio, t_final, tiempo,pulsosA=0;
double rpm, nVueltas, resolucion = 1024, minuto=60000000.0;
const byte interruptPin = 50;//pin del encoder
void setup() {
  Serial.begin(9600);
  analogWriteResolution(12);//resolucion de 12 bits para DAC
  pinMode(mov_motor, OUTPUT);
  pinMode(giro, OUTPUT);
  pinMode(FWD, OUTPUT);
  pinMode(REV, OUTPUT);
  pinMode(27, OUTPUT);
  pinMode(29, OUTPUT);
  pinMode(47, OUTPUT);
  pinMode(49, OUTPUT);
  pinMode(sensor_primero, INPUT);
```

```
  pinMode(sensor_segundo, INPUT);
  pinMode(sensor_tercero, INPUT);
  pinMode(sensor_cuarto, INPUT);
  pinMode(sensor_quinto, INPUT);
  pinMode(sensor_fin_carrera, INPUT);
  pinMode(sensor_primero2, INPUT);
  pinMode(sensor_segundo2, INPUT);
  pinMode(sensor_tercero2, INPUT);
  pinMode(sensor_cuarto2, INPUT);
  pinMode(sensor_quinto2, INPUT);
  pinMode(sensor_fin_carrera2, INPUT);
  digitalWrite(mov_motor, HIGH);
  digitalWrite(giro, HIGH);
  digitalWrite(FWD, HIGH);
  digitalWrite(REV, HIGH);
  digitalWrite(27, HIGH);
  digitalWrite(29, HIGH);
  digitalWrite(47, HIGH);
  digitalWrite(49, HIGH);
  attachInterrupt(digitalPinToInterrupt(interruptPin), interrupcion, RISING);//Funcion
interrupcion en pin INPUT 52 en flanco ascendente.
  t_inicio=micros();  //Mide el tiempo de ciclo de scan.
}
void loop(){
  while (Serial.available()) {
    letra = Serial.read();  //lee el byte en el serial
    palabra += letra; //hacer la palabra sumando letras
    }
//==================posición del
elevador1============================================
if(digitalRead(sensor_primero)==HIGH){
  //Serial.println("sensor primera planta");
  piso_envio=0;
  }
  if(digitalRead(sensor_segundo)==HIGH){
  piso_envio=1;
  }
  if(digitalRead(sensor_tercero)==HIGH){
  piso_envio=2;
  }
  if(digitalRead(sensor_cuarto)==HIGH){
  piso_envio=3;
  }
  if(digitalRead(sensor_quinto)==HIGH){
  piso_envio=4;
  }
//==================fin posición del
elevador1============================================
//==================posición del
elevador2============================================
```

```
if(digitalRead(sensor_primero2)==HIGH){
  piso_envio2=0;
  }
  if(digitalRead(sensor_segundo2)==HIGH){
  piso_envio2=1;
  }
  if(digitalRead(sensor_tercero2)==HIGH){
  piso_envio2=2;
  }
  if(digitalRead(sensor_cuarto2)==HIGH){
  piso_envio2=3;
  }
  if(digitalRead(sensor_quinto2)==HIGH){
  piso_envio2=4;
  }
//==================fin posición del
elevador2==============================================
//==================Velocidad de
giro=============================================.
//Serial.println(ref_vel);
    if (palabra.length() >0 && letra!='i' && letra!='p' && letra!='u' && letra!='d' &&
letra!='t' && letra!='c' && letra!='q' && letra!='I' && letra!='P' && letra!='G' &&
letra!='g' && letra!='n' && letra!='N' && letra!='m' && letra!='M' && letra!='k' &&
letra!='K' && letra!='E' && letra!='B') {
      palabra_entero = palabra.toInt();  //convert readString into a number
      ref_vel=palabra_entero;
      palabra=""; //limpia la palabra
    } else
    { ref_vel=palabra_entero;
      palabra="";
      }
//==================FIN Velocidad de
giro============================================
//==================Variables bluetooth
recibidas=================================
    if (letra=='i') {
      iniciar=1;
      digitalWrite(REV, HIGH);//giro antihorario OFF
      digitalWrite(FWD, HIGH);//giro antihorario OFF
    }
    if (letra=='p') {
      iniciar=2;
    }
     if (letra=='u') {
      caso=1;
    }
    if (letra=='d') {
      caso=2;
    }
    if (letra=='t') {
```

```
   caso=3;
}
 if (letra=='c') {
   caso=4;
}
 if (letra=='q') {
   caso=5;
}
//para modo manual
if (letra=='I') {
  iniciar=3;
   digitalWrite(mov_motor, LOW);
}
if (letra=='P') {
   iniciar=3;
   digitalWrite(mov_motor, HIGH);
}
if (letra=='g') {
   digitalWrite(giro, LOW);//giro horario
}
if (letra=='G') {
   digitalWrite(giro, HIGH);//giro antihorario
}
if (letra=='m') {

   digitalWrite(FWD, LOW);//giro horario ON
}
if (letra=='M') {

   digitalWrite(FWD, HIGH);//giro horario OFF
}
if (letra=='n') {

   digitalWrite(REV, LOW);//giro antihorario ON
}
if (letra=='N') {

   digitalWrite(REV, HIGH);//giro antihorario OFF
}
 if (letra=='k') {
  analogWrite(DAC1,0);
  // digitalWrite(VX1, HIGH);
  // digitalWrite(VX2, HIGH);
}
if (letra=='K') {
  analogWrite(DAC1,1500);
   //digitalWrite(VX1, LOW);
   //digitalWrite(VX2, HIGH);
}
if (letra=='E') {
```

```
     analogWrite(DAC1,3000);
    //  digitalWrite(VX1, HIGH);
    //  digitalWrite(VX2, LOW);
    }
    if (letra=='B') {
      analogWrite(DAC1,4095);
     // digitalWrite(VX1, LOW);
     // digitalWrite(VX2, LOW);
    }
//==================fin Variables bluetooth
recibidas===================================
//==================mover
motor=============================================
 if(iniciar==1){
   salida_dac=map(ref_vel,0,50,0,4095);
   //Serial.println(salida_dac);
   analogWrite(DAC0,salida_dac);
   switch (caso) {
    case 5:
    //Ir a piso 4
      if(digitalRead(sensor_cuarto)==HIGH || digitalRead(sensor_tercero)==HIGH ||
digitalRead(sensor_segundo)==HIGH || digitalRead(sensor_primero)==HIGH)//esta en
el piso 4,3,2,1
          {
         digitalWrite(giro, LOW);//giro arriba
         accion_envio=1;
         ok=1;
          }
       if(digitalRead(sensor_quinto)==LOW && ok==1) //no esta quinto y verifica que
esta en cualquiera de los 4 pisos
          {
         digitalWrite(mov_motor, LOW);
          }else{
            digitalWrite(mov_motor, HIGH);
            accion_envio=0;
            ok=0;
            }
        // SEGUNDO ELEVADOR
       if(digitalRead(sensor_cuarto2)==HIGH || digitalRead(sensor_tercero2)==HIGH ||
digitalRead(sensor_segundo2)==HIGH || digitalRead(sensor_primero2)==HIGH)//esta
en el piso 4
          {
         accion_envio2=1;//GIRO PARA ARRIBA
         ok2=1;// gira hacia arriba
          }
       if(digitalRead(sensor_quinto2)==LOW && ok2==1) //no esta quinto y verifica que
esta en cualquiera de los 4 pisos
          {
         digitalWrite(FWD, LOW);//giro arriba on
         digitalWrite(REV, HIGH);//giro abajo off
```

```
        }else{
         if(digitalRead(sensor_quinto2)==LOW && ok2==2){
           digitalWrite(FWD, HIGH);//giro arriba off
           digitalWrite(REV, LOW);//giro abajo on
         }
           else
           {
            digitalWrite(FWD, HIGH);//giro arriba off
            digitalWrite(REV, HIGH);//giro abajo off
            accion_envio2=0;
            ok2=0;
           }
         }
   break;
    case 4:
     if(digitalRead(sensor_quinto)==HIGH )//esta en el piso 5
        {
       digitalWrite(giro, HIGH);//giro abajo
       accion_envio=2;
       ok=1;
        }
      if(digitalRead(sensor_tercero)==HIGH || digitalRead(sensor_segundo)==HIGH ||
digitalRead(sensor_primero)==HIGH )//esta en el piso 3,2,1
        {
       digitalWrite(giro, LOW);//giro arriba
       accion_envio=1;
       ok=1;
        }
     if(digitalRead(sensor_cuarto)==LOW && ok==1) //no esta cuarto y verifica que
esta en cualquiera de los 4 pisos
        {
       digitalWrite(mov_motor, LOW);
        }else{
         digitalWrite(mov_motor, HIGH);
          accion_envio=0;
         ok=0;
        }
      // SEGUNDO ELEVADOR
       if(digitalRead(sensor_quinto2)==HIGH )//esta en el piso 5
        {
       accion_envio2=2;//gira hacia abajo
       ok2=2;//gira hacia abajo
        }
      if(digitalRead(sensor_tercero2)==HIGH || digitalRead(sensor_segundo2)==HIGH
|| digitalRead(sensor_primero2)==HIGH )//esta en el piso 3,2,1
        {
       accion_envio2=1;//gira hacia arriba
       ok2=1;//gira hacia arriba
        }
```

```
        if(digitalRead(sensor_cuarto2)==LOW && ok2==1) //no esta cuarto y verifica
que esta en cualquiera de los 4 pisos
        {
       digitalWrite(FWD, LOW);//giro arriba on
       digitalWrite(REV, HIGH);//giro abajo off
        }else{
         if(digitalRead(sensor_cuarto2)==LOW && ok2==2){
          digitalWrite(FWD, HIGH);//giro arriba off
          digitalWrite(REV, LOW);//giro abajo on
         }
          else
          {
           digitalWrite(FWD, HIGH);//giro arriba off
           digitalWrite(REV, HIGH);//giro abajo off
           accion_envio2=0;
           ok2=0;
          }
         }
   break;
   case 3:
     if(digitalRead(sensor_quinto)==HIGH || digitalRead(sensor_cuarto)==HIGH)//esta
en el piso 5,4
        {
       digitalWrite(giro, HIGH);//giro abajo
       accion_envio=2;
       ok=1;
        }
     if(digitalRead(sensor_segundo)==HIGH ||
digitalRead(sensor_primero)==HIGH)//esta en el piso 2,1
        {
       digitalWrite(giro, LOW);//giro arriba
       accion_envio=1;
       ok=1;
        }
      if(digitalRead(sensor_tercero)==LOW && ok==1) //no esta tercero y verifica que
esta en cualquiera de los 4 pisos
        {
       digitalWrite(mov_motor, LOW);
        }else{
         digitalWrite(mov_motor, HIGH);
         accion_envio=0;
         ok=0;
        }
      // SEGUNDO ELEVADOR
      if(digitalRead(sensor_quinto2)==HIGH ||
digitalRead(sensor_cuarto2)==HIGH)//esta en el piso 5,4
        {
       accion_envio2=2;//hacia abajo
       ok2=2;//hacia abajo
        }
```

```
      if(digitalRead(sensor_segundo2)==HIGH ||
digitalRead(sensor_primero2)==HIGH)//esta en el piso 2,1
        {
        accion_envio2=1;//hacia arriba
        ok2=1;//hacia arriba
        }
       if(digitalRead(sensor_tercero2)==LOW && ok2==1) //no esta tercero y verifica
que esta en cualquiera de los 4 pisos
        {
        digitalWrite(FWD, LOW);//giro arriba on
        digitalWrite(REV, HIGH);//giro abajo off
        }else{
          if(digitalRead(sensor_tercero2)==LOW && ok2==2){
            digitalWrite(FWD, HIGH);//giro arriba off
            digitalWrite(REV, LOW);//giro abajo on
          }
            else
            {
             digitalWrite(FWD, HIGH);//giro arriba off
             digitalWrite(REV, HIGH);//giro abajo off
             accion_envio2=0;
             ok2=0;
            }
          }
   break;
   case 2:
     if(digitalRead(sensor_quinto)==HIGH || digitalRead(sensor_cuarto)==HIGH ||
digitalRead(sensor_tercero)==HIGH)//esta en el piso 5,4,3
        {
        digitalWrite(giro, HIGH);//giro abajo
        accion_envio=2;
        ok=1;
        }
       if(digitalRead(sensor_primero)==HIGH)//esta en el piso 1
           {
           digitalWrite(giro, LOW);//giro arriba
           accion_envio=1;
           ok=1;
           }
       if(digitalRead(sensor_segundo)==LOW && ok==1) //no esta segundo y verifica
que esta en cualquiera de los 4 pisos
           {
           digitalWrite(mov_motor, LOW);
           }else{
             digitalWrite(mov_motor, HIGH);
             accion_envio=0;
             ok=0;
          }
        // SEGUNDO ELEVADOR
```

```
      if(digitalRead(sensor_quinto2)==HIGH || digitalRead(sensor_cuarto2)==HIGH ||
digitalRead(sensor_tercero2)==HIGH)//esta en el piso 5,4,3
       {
       accion_envio2=2;
       ok2=2;//giro abajo
       }
      if(digitalRead(sensor_primero2)==HIGH)//esta en el piso 1
          {
         accion_envio2=1;
         ok2=1;//giro arriba
          }
      if(digitalRead(sensor_segundo2)==LOW && ok2==1) //no esta segundo y verifica
que esta en cualquiera de los 4 pisos
       {
       digitalWrite(FWD, LOW);//giro arriba on
       digitalWrite(REV, HIGH);//giro abajo off
       }else{
        if(digitalRead(sensor_segundo2)==LOW && ok2==2){
          digitalWrite(FWD, HIGH);//giro arriba off
          digitalWrite(REV, LOW);//giro abajo on
        }
          else
          {
           digitalWrite(FWD, HIGH);//giro arriba off
           digitalWrite(REV, HIGH);//giro abajo off
           accion_envio2=0;
           ok2=0;
          }
        }
   break;
    case 1:
     if(digitalRead(sensor_quinto)==HIGH || digitalRead(sensor_cuarto)==HIGH ||
digitalRead(sensor_segundo)==HIGH ||digitalRead(sensor_tercero)==HIGH)//esta en el
piso 5,4,3,2
       {
       digitalWrite(giro, HIGH);//giro abajo
       accion_envio=2;
       ok=1;
       }
      if(digitalRead(sensor_primero)==LOW && ok==1)//no esta primero y verifica que
esta en cualquiera de los 4 pisos
          {
         digitalWrite(mov_motor, LOW);
         }else{
           digitalWrite(mov_motor, HIGH);
            accion_envio=0;
           ok=0;
        }
      // SEGUNDO ELEVADOR
```

```
        if(digitalRead(sensor_quinto2)==HIGH || digitalRead(sensor_cuarto2)==HIGH ||
digitalRead(sensor_segundo2)==HIGH ||digitalRead(sensor_tercero2)==HIGH)//esta en
el piso 5,4,3,2
        {
       accion_envio2=2;//giro abajo
       ok2=2;//giro abajo
        }
      if(digitalRead(sensor_primero2)==LOW && ok2==1) //no esta segundo y verifica
que esta en cualquiera de los 4 pisos
        {
       digitalWrite(FWD, LOW);//giro arriba on
       digitalWrite(REV, HIGH);//giro abajo off
        }else{
         if(digitalRead(sensor_primero2)==LOW && ok2==2){
           digitalWrite(FWD, HIGH);//giro arriba off
           digitalWrite(REV, LOW);//giro abajo on
         }
           else
           {
            digitalWrite(FWD, HIGH);//giro arriba off
            digitalWrite(REV, HIGH);//giro abajo off
            accion_envio2=0;
            ok2=0;
           }
         }
  break;
  }
 }
   //pregunta si pulso stop, entonces caso es 0, se apaga el motor y la salida del dac es 0
  if(iniciar==2){
   analogWrite(DAC0,0);
   digitalWrite(mov_motor, HIGH);
   digitalWrite(FWD, HIGH);//giro arriba off
   digitalWrite(REV, HIGH);//giro abajo off
  // digitalWrite(VX1, HIGH);//VEL 1
  // digitalWrite(VX2, HIGH);//VEL 2
   analogWrite(DAC1,0);
   }
   if(iniciar==3){
  salida_dac=map(ref_vel,0,50,0,4095);
  analogWrite(DAC0,salida_dac);
   }

  if(digitalRead(sensor_fin_carrera)==HIGH)
  {
  alarma_envio=1;
  iniciar=2;
  accion_envio=0;
  analogWrite(DAC0,0);
  digitalWrite(mov_motor, HIGH);
```

```
  //Serial.println("no detecta sensores");
  }
  else{
  alarma_envio=0;
    }
  if(digitalRead(sensor_fin_carrera2)==HIGH)
  {
  alarma_envio2=1;
  iniciar=2;
  accion_envio2=0;
  digitalWrite(FWD, HIGH);//giro arriba off
  digitalWrite(REV, HIGH);//giro abajo off
  //digitalWrite(VX1, HIGH);//VEL 1
  //digitalWrite(VX2, HIGH);//VEL 2
  analogWrite(DAC1,0);
  }
  else{
  alarma_envio2=0;
    }
//==================fin mover
motor=============================================
//====================inicio
pulsos===================================
  delay(95);//100ms
  noInterrupts();//codigo sensible al tiempo
  pulsosF = pulsos*10;
  pulsos = 0;
  t_final=micros();
  interrupts();//codigo normal
  tiempo=t_final-t_inicio;
  t_inicio=t_final;
  nVueltas=pulsosF/resolucion;
  rpm = (nVueltas*minuto)/tiempo;
//====================fin
pulsos===================================
//================================Envia datos a la
App============================
Serial.print("X");
delay(5);
Serial.print("_");
Serial.print(piso_envio);//envia datos primera unidad lineal del 0 al 4
Serial.print("_");
Serial.print(piso_envio2);//envia datos segunda unidad lineal del 0 al 4
Serial.print("_");
Serial.print(alarma_envio);//alarma fin carrera primera unidad lineal 1 encendido 0
apagado
Serial.print("_");
Serial.print(alarma_envio2);//alarma fin carrera segunda unidad lineal 1 encendido 0
apagado
Serial.print("_");
```

```
Serial.print(accion_envio);//accion de subir 1 bajar 2 nada 0 primera unidad lineal
Serial.print("_");
Serial.print(accion_envio2);
//accion de subir 1 bajar 2 nada 0 segunda unidad lineal
Serial.print("_");
Serial.println(rpm);
//================================Fin Envia datos a la
App============================
}

//interrupcion del contador
void  interrupcion ()
{
  pulsos++;
}
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PROTOTIPO DE UN ASCENSOR Y SU CONTROL COMPARANDO TECNOLOGÍAS DIFERENTES DE ACCIONAMIENTO

ANEXO VIII: CÓDIGO FUENTE RASPBERRY

```
import time
import RPi.GPIO as GPIO#salidas de la raspberry gpio
import GS_timing as timing #modulo creado para los milisegundos
import sys
from bluetooth import *

#==================para
bluetooth=======================================
server_sock=BluetoothSocket( RFCOMM )
server_sock.bind(("",PORT_ANY))
server_sock.listen(1)

port = server_sock.getsockname()[1]

uuid = "94f39d29-7d6d-437d-973b-fba39e49d4ee"

advertise_service( server_sock, "SampleServer",
                   service_id = uuid,
                   service_classes = [ uuid, SERIAL_PORT_CLASS ],
                   profiles = [ SERIAL_PORT_PROFILE ],
#                    protocols = [ OBEX_UUID ]
                    )

print("Esperando a que el dispositivo de conecte a la raspberry %d" % port)
client_sock, client_info = server_sock.accept()
print("Conexion correcta de la direccion Bluetooth: ", client_info)
#==================fin para
bluetooth=======================================
#variables iniciales
#pines conectados blanco rele 1 orden 21 20 16 26 19 13 12 06
start = time.time()* 1000000
ref_vel=0;
palabra=""#palabra recibida de velocidad
letra="#accion recibida
data="
caso = 0#caso inicial posicion del elevador
caso2 = 0#caso inicial posicion del elevador2
palabra_entero = 25#velocidad por defecto
iniciar = 0#variable que inicia modo ascensor
salida_dac = 4095#salida del pin que controla la velocidad
alarma_envio = 0#alarma unidad lineal 1 en caso de fin de carrera
alarma_envio2 = 0#alarma unidad lineal 2 en caso de fin de carrera
piso_envio = 0#piso que indica sensores UL1
piso_envio2 = 0#piso que indica sensores UL2
accion_envio = 0#subir o bajar UL1
accion_envio2 = 0#subir o bajar UL2
ok = 0#verifica que detecta sensor UL1
ok2 = 0#verifica que detecta sensor UL2
giro = 2#pin giro hora y antihorario variador de frecuencia GPIO2
mov_motor = 3#pin mueve o apaga motor asincrono GPIO3
```

```
#GPIO04 y GPIO17 reservados del rele D3 Y D4 en variador
FWD=27#GIRO HORARIO SERVOMOTOR
REV=22#GIRO ANTIHORARIO SERVOMOTOR
#GPIO19 y GPIO16 reservados del rele D7 Y D8 en servomotor
sensor_primero = 5
sensor_segundo = 6
sensor_tercero = 26
sensor_cuarto = 10
sensor_quinto = 9
sensor_fin_carrera = 23
sensor_primero2 = 24
sensor_segundo2 = 25
sensor_tercero2 = 8
sensor_cuarto2 = 7
sensor_quinto2 = 12
sensor_fin_carrera2 = 20
#variables parte medidor de velocidad
nVueltas = 0
pulsosF = 0
pulsos = 0
resolucion = 1024
minuto = 60000000.0

#Void setup() {
GPIO.setmode(GPIO.BCM)     #programming the GPIO by BCM pin numbers. (like
PIN40 as GPIO21)
GPIO.setwarnings(False)
GPIO.setup(mov_motor,GPIO.OUT)  #pin de salida GPIO3
GPIO.setup(giro,GPIO.OUT)  #pin salida GPIO2
GPIO.setup(FWD,GPIO.OUT)  #pin salida GPIO27
GPIO.setup(REV,GPIO.OUT)  #pin salida GPIO22
GPIO.setup(4,GPIO.OUT)  #AUX1 variador
GPIO.setup(17,GPIO.OUT)  #AUX2 variador
GPIO.setup(19,GPIO.OUT)  #AUX 1 servomotor
GPIO.setup(16,GPIO.OUT)  #AUX 2 servomotor
GPIO.setup(sensor_primero, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  #pin
entrada GPIO5
GPIO.setup(sensor_segundo, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  #pin
entrada GPIO6
GPIO.setup(sensor_tercero, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  #pin
entrada GPIO26
GPIO.setup(sensor_cuarto, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  #pin
entrada GPIO14
GPIO.setup(sensor_quinto, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  #pin
entrada GPIO15
GPIO.setup(sensor_fin_carrera, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  #pin
entrada GPIO23
GPIO.setup(sensor_primero2, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  #pin
entrada GPIO24
```

```
GPIO.setup(sensor_segundo2, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  #pin
entrada GPIO25
GPIO.setup(sensor_tercero2, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  #pin
entrada GPIO8
GPIO.setup(sensor_cuarto2, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  #pin
entrada GPIO7
GPIO.setup(sensor_quinto2, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  #pin
entrada GPIO12
GPIO.setup(sensor_fin_carrera2, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  #pin
entrada GPIO20
GPIO.setup(18,GPIO.OUT)#pwm1
GPIO.setup(13,GPIO.OUT)#pwm2
PWM1=GPIO.PWM(18,100)
PWM2=GPIO.PWM(13,100)
GPIO.output(mov_motor,1)
GPIO.output(giro,1)
GPIO.output(FWD,1)
GPIO.output(REV,1)
GPIO.output(4,1)
GPIO.output(17,1)
GPIO.output(19,1)
GPIO.output(16,1)


GPIO.setup(11, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)#configurar pin de
interrupciones
#definir interrupcion
def inter(channel):
    global pulsos
    pulsos = pulsos+1
    #sys.stdout.write('%d' % pulsos)
    sys.stdout.flush()
#definir interrupcion
def nointer(channel):
    global pulsos
#Pin GPIO SENTRADA ENCODER FLANCO ASCENDENTE
GPIO.add_event_detect(11, GPIO.RISING, callback=inter)
#}
#Void loop()
try:
    while True:
        #delayMillisecond(1000)
        time.sleep(0.1)
        #NO INTERRUPTS
        callback=nointer
        pulsosF=pulsos*10
        pulsos=0
        #end = time.time()* 1000000
        #INTERRUPTS
        callback=inter
```

```
        time_elapsed = 1000000 #end - start
        time_in_miliseconds = time_elapsed
        data = client_sock.recv(1024)
        #client_sock.setblocking(False)
        #if len(data) == 0: break
        letra = data#lee el byte en el serial
        palabra+=letra#hace la palabra sumando las letras
        #print("Recibe: %s" % palabra)

            #client_sock.send(data)
    #=========================fin posicion elevador
2=======================================
     #=========================Velocidad de
giro=======================================
        if (len(palabra) > 0 and letra!= 'i' and letra!= 'p' and letra!= 'u' and letra!= 'd' and
letra!= 't'and letra!= 'c' and letra!= 'q' and letra!= 'I' and letra!= 'P' and letra!= 'G' and
letra!= 'g' and letra!= 'n' and letra!= 'N' and letra!= 'm' and letra!= 'M' and letra!= 'k' and
letra!= 'K' and letra!= 'E' and letra!= 'B'):
            palabra_entero=int(palabra)
            ref_vel=palabra_entero
            palabra=""
        else:
            ref_vel=palabra_entero
            palabra=""
    #=========================fin Velocidad de
giro=======================================
    #=========================Variables bluetooth
recibidas=======================================
        if (letra  == 'i'):
            iniciar=1
            GPIO.output(REV,1)#GIRO antihorario OFF servo
            GPIO.output(FWD,1)#GIRO horario OFF servo
        if (letra  == 'p'):
            iniciar=2
        if (letra  == 'u'):
            caso=1
            caso2=1
        if (letra  == 'd'):
            caso=2
            caso2=2
        if (letra  == 't'):
            caso=3
            caso2=3
        if (letra  == 'c'):
            caso=4
            caso2=4
        if (letra  == 'q'):
            caso=5
            caso2=5
        #para modo manual
```

```
        if (letra  == 'I'):
            iniciar=3
            GPIO.output(mov_motor,0)
        if (letra  == 'P'):
            iniciar=3
            GPIO.output(mov_motor,1)
        if (letra  == 'g'):
            GPIO.output(giro,0)
        if (letra  == 'G'):
            GPIO.output(giro,1)
        if (letra  == 'm'):
            GPIO.output(FWD,0)
        if (letra  == 'M'):
            GPIO.output(FWD,1)
        if (letra  == 'n'):
            GPIO.output(REV,0)
        if (letra  == 'N'):
            GPIO.output(REV,1)
        if (letra  == 'k'):
            PWM2.start(0)
        if (letra  == 'K'):
            PWM2.start(36)
        if (letra  == 'E'):
            PWM2.start(75)
        if (letra  == 'B'):
            PWM2.start(100)
    #=========================finVariables bluetooth
recibidas=======================================
        if(iniciar==1):
            PWM1.start(ref_vel*2)
            while(caso==5 or caso2==5):#ir al cuarto piso
                #datos enviados
                if GPIO.input(sensor_primero)==GPIO.HIGH:
                    piso_envio=0
                if GPIO.input(sensor_segundo)==GPIO.HIGH:
                    piso_envio=1
                if GPIO.input(sensor_tercero)==GPIO.HIGH:
                    piso_envio=2
                if GPIO.input(sensor_cuarto)==GPIO.HIGH:
                    piso_envio=3
                if GPIO.input(sensor_quinto)==GPIO.HIGH:
                    piso_envio=4
                if GPIO.input(sensor_primero2)==GPIO.HIGH:
                    piso_envio2=0
                if GPIO.input(sensor_segundo2)==GPIO.HIGH:
                    piso_envio2=1
                if GPIO.input(sensor_tercero2)==GPIO.HIGH:
                    piso_envio2=2
                if GPIO.input(sensor_cuarto2)==GPIO.HIGH:
                    piso_envio2=3
```

```
            if GPIO.input(sensor_quinto2)==GPIO.HIGH:
               piso_envio2=4
            client_sock.send("X")
            time.sleep(0.010)
            client_sock.send("_")
            client_sock.send(str(piso_envio))
            client_sock.send("_")
            client_sock.send(str(piso_envio2))
            client_sock.send("_")
            client_sock.send(str(alarma_envio))
            client_sock.send("_")
            client_sock.send(str(alarma_envio2))
            client_sock.send("_")
            client_sock.send(str(accion_envio))
            client_sock.send("_")
            client_sock.send(str(accion_envio2))
            client_sock.send("_")
            client_sock.send(str(pulsos))
            #Primer Elevador
            if GPIO.input(sensor_cuarto)==GPIO.HIGH or
GPIO.input(sensor_tercero)==GPIO.HIGH or
GPIO.input(sensor_segundo)==GPIO.HIGH or
GPIO.input(sensor_primero)==GPIO.HIGH:#esta en el 4,3,2,1
               GPIO.output(giro,0)
               accion_envio=1
               ok=1
            if GPIO.input(sensor_quinto)==GPIO.LOW and ok==1:#no esta en quinto y
verifica que todo esta bien
               GPIO.output(mov_motor,0)
            else:
               GPIO.output(mov_motor,1)
               accion_envio=0
               ok=0
               caso=0
            if GPIO.input(sensor_fin_carrera)==GPIO.HIGH or
GPIO.input(sensor_fin_carrera2)==GPIO.HIGH:
               alarma_envio=1
               iniciar=2
               accion_envio=0
               PWM1.start(0)
               GPIO.output(mov_motor,1)
               caso=0
               caso2=0
               print "sensor final de carrera"
            #Segundo Elevador
            if GPIO.input(sensor_cuarto2)==GPIO.HIGH or
GPIO.input(sensor_tercero2)==GPIO.HIGH or
GPIO.input(sensor_segundo2)==GPIO.HIGH or
GPIO.input(sensor_primero2)==GPIO.HIGH:#esta en el 4,3,2,1
               accion_envio2=1
```

```
            ok2=1
        if GPIO.input(sensor_quinto2)==GPIO.LOW and ok2==1:#no esta en quinto y
verifica que todo esta bien
            GPIO.output(FWD,0)
            GPIO.output(REV,1)
        else:
            if GPIO.input(sensor_quinto2)==GPIO.LOW and ok2==2:
                GPIO.output(FWD,1)
                GPIO.output(REV,0)
            else:
                GPIO.output(FWD,1)
                GPIO.output(REV,1)
                accion_envio2=0
                ok2=0
                caso2=0


    while(caso==4 or caso2==4):#ir al tercer piso
        #datos enviados
        if GPIO.input(sensor_primero)==GPIO.HIGH:
            piso_envio=0
        if GPIO.input(sensor_segundo)==GPIO.HIGH:
            piso_envio=1
        if GPIO.input(sensor_tercero)==GPIO.HIGH:
            piso_envio=2
        if GPIO.input(sensor_cuarto)==GPIO.HIGH:
            piso_envio=3
        if GPIO.input(sensor_quinto)==GPIO.HIGH:
            piso_envio=4
        if GPIO.input(sensor_primero2)==GPIO.HIGH:
            piso_envio2=0
        if GPIO.input(sensor_segundo2)==GPIO.HIGH:
            piso_envio2=1
        if GPIO.input(sensor_tercero2)==GPIO.HIGH:
            piso_envio2=2
        if GPIO.input(sensor_cuarto2)==GPIO.HIGH:
            piso_envio2=3
        if GPIO.input(sensor_quinto2)==GPIO.HIGH:
            piso_envio2=4
        client_sock.send("X")
        time.sleep(0.010)
        client_sock.send("_")
        client_sock.send(str(piso_envio))
        client_sock.send("_")
        client_sock.send(str(piso_envio2))
        client_sock.send("_")
        client_sock.send(str(alarma_envio))
        client_sock.send("_")
        client_sock.send(str(alarma_envio2))
        client_sock.send("_")
```

```
            client_sock.send(str(accion_envio))
            client_sock.send("_")
            client_sock.send(str(accion_envio2))
            client_sock.send("_")
            client_sock.send(str(pulsos))
            #Primer Elevador
            if GPIO.input(sensor_fin_carrera)==GPIO.HIGH or
GPIO.input(sensor_fin_carrera2)==GPIO.HIGH:
                alarma_envio=1
                iniciar=2
                accion_envio=0
                PWM1.start(0)
                GPIO.output(mov_motor,1)
                caso=0
                caso2=0
                print "sensor final de carrera"
            if GPIO.input(sensor_quinto)==GPIO.HIGH:#esta en el quinto piso
                GPIO.output(giro,1)
                accion_envio=2
                ok=1
            if GPIO.input(sensor_tercero)==GPIO.HIGH or
GPIO.input(sensor_segundo)==GPIO.HIGH or
GPIO.input(sensor_primero)==GPIO.HIGH:#esta en el 3,2,1
                GPIO.output(giro,0)
                accion_envio=1
                ok=1
            if GPIO.input(sensor_cuarto)==GPIO.LOW and ok==1:
                GPIO.output(mov_motor,0)
            else:
                GPIO.output(mov_motor,1)
                accion_envio=0
                ok=0
                caso=0
            #Segundo Elevador
            if GPIO.input(sensor_quinto2)==GPIO.HIGH:#esta en el quinto piso
                accion_envio2=2
                ok2=2
            if GPIO.input(sensor_tercero2)==GPIO.HIGH or
GPIO.input(sensor_segundo2)==GPIO.HIGH or
GPIO.input(sensor_primero2)==GPIO.HIGH:#esta en el 3,2,1
                accion_envio2=1
                ok2=1
            if GPIO.input(sensor_cuarto2)==GPIO.LOW and ok2==1:#no esta en cuarto y
verifica que todo esta bien
                GPIO.output(FWD,0)
                GPIO.output(REV,1)
            else:
                if GPIO.input(sensor_cuarto2)==GPIO.LOW and ok2==2:
                    GPIO.output(FWD,1)
                    GPIO.output(REV,0)
```

```
                else:
                    GPIO.output(FWD,1)
                    GPIO.output(REV,1)
                    accion_envio2=0
                    ok2=0
                    caso2=0
            while(caso==3 or caso2==3):#ir al tercer piso
                #datos enviados
                if GPIO.input(sensor_primero)==GPIO.HIGH:
                    piso_envio=0
                if GPIO.input(sensor_segundo)==GPIO.HIGH:
                    piso_envio=1
                if GPIO.input(sensor_tercero)==GPIO.HIGH:
                    piso_envio=2
                if GPIO.input(sensor_cuarto)==GPIO.HIGH:
                    piso_envio=3
                if GPIO.input(sensor_quinto)==GPIO.HIGH:
                    piso_envio=4
                if GPIO.input(sensor_primero2)==GPIO.HIGH:
                    piso_envio2=0
                if GPIO.input(sensor_segundo2)==GPIO.HIGH:
                    piso_envio2=1
                if GPIO.input(sensor_tercero2)==GPIO.HIGH:
                    piso_envio2=2
                if GPIO.input(sensor_cuarto2)==GPIO.HIGH:
                    piso_envio2=3
                if GPIO.input(sensor_quinto2)==GPIO.HIGH:
                    piso_envio2=4
                client_sock.send("X")
                time.sleep(0.010)
                client_sock.send("_")
                client_sock.send(str(piso_envio))
                client_sock.send("_")
                client_sock.send(str(piso_envio2))
                client_sock.send("_")
                client_sock.send(str(alarma_envio))
                client_sock.send("_")
                client_sock.send(str(alarma_envio2))
                client_sock.send("_")
                client_sock.send(str(accion_envio))
                client_sock.send("_")
                client_sock.send(str(accion_envio2))
                client_sock.send("_")
                client_sock.send(str(pulsos))
                #Primer Elevador
                if GPIO.input(sensor_fin_carrera)==GPIO.HIGH or
GPIO.input(sensor_fin_carrera2)==GPIO.HIGH:
                    alarma_envio=1
                    iniciar=2
                    accion_envio=0
```

```
                PWM1.start(0)
                GPIO.output(mov_motor,1)
                caso=0
                caso2=0
                print "sensor final de carrera"
            if GPIO.input(sensor_quinto)==GPIO.HIGH or
GPIO.input(sensor_cuarto)==GPIO.HIGH:#esta en el 5,4
                GPIO.output(giro,1)
                accion_envio=2
                ok=1
            if GPIO.input(sensor_segundo)==GPIO.HIGH or
GPIO.input(sensor_primero)==GPIO.HIGH:#esta en el 2,1
                GPIO.output(giro,0)
                accion_envio=1
                ok=1
            if GPIO.input(sensor_tercero)==GPIO.LOW and ok==1:
                GPIO.output(mov_motor,0)
            else:
                GPIO.output(mov_motor,1)
                accion_envio=0
                ok=0
                caso=0
            #segundo Elevador
            if GPIO.input(sensor_quinto2)==GPIO.HIGH or
GPIO.input(sensor_cuarto2)==GPIO.HIGH:#esta en el 5,4
                accion_envio2=2
                ok2=2
            if GPIO.input(sensor_segundo)==GPIO.HIGH or
GPIO.input(sensor_primero)==GPIO.HIGH:#esta en el 2,1
                accion_envio2=1
                ok2=1
            if GPIO.input(sensor_tercero2)==GPIO.LOW and ok2==1:#no esta en tercero
y verifica que todo esta bien
                GPIO.output(FWD,0)
                GPIO.output(REV,1)
            else:
                if GPIO.input(sensor_tercero2)==GPIO.LOW and ok2==2:
                    GPIO.output(FWD,1)
                    GPIO.output(REV,0)
                else:
                    GPIO.output(FWD,1)
                    GPIO.output(REV,1)
                    accion_envio2=0
                    ok2=0
                    caso2=0
        while(caso==2 or caso2==2):#ir al segundo piso
            #datos enviados
            if GPIO.input(sensor_primero)==GPIO.HIGH:
                piso_envio=0
            if GPIO.input(sensor_segundo)==GPIO.HIGH:
```

```
                piso_envio=1
            if GPIO.input(sensor_tercero)==GPIO.HIGH:
                piso_envio=2
            if GPIO.input(sensor_cuarto)==GPIO.HIGH:
                piso_envio=3
            if GPIO.input(sensor_quinto)==GPIO.HIGH:
                piso_envio=4
            if GPIO.input(sensor_primero2)==GPIO.HIGH:
                piso_envio2=0
            if GPIO.input(sensor_segundo2)==GPIO.HIGH:
                piso_envio2=1
            if GPIO.input(sensor_tercero2)==GPIO.HIGH:
                piso_envio2=2
            if GPIO.input(sensor_cuarto2)==GPIO.HIGH:
                piso_envio2=3
            if GPIO.input(sensor_quinto2)==GPIO.HIGH:
                piso_envio2=4
            client_sock.send("X")
            time.sleep(0.010)
            client_sock.send("_")
            client_sock.send(str(piso_envio))
            client_sock.send("_")
            client_sock.send(str(piso_envio2))
            client_sock.send("_")
            client_sock.send(str(alarma_envio))
            client_sock.send("_")
            client_sock.send(str(alarma_envio2))
            client_sock.send("_")
            client_sock.send(str(accion_envio))
            client_sock.send("_")
            client_sock.send(str(accion_envio2))
            client_sock.send("_")
            client_sock.send(str(pulsos))
            #Primer Elevador
            if GPIO.input(sensor_fin_carrera)==GPIO.HIGH or
GPIO.input(sensor_fin_carrera2)==GPIO.HIGH:
                alarma_envio=1
                iniciar=2
                accion_envio=0
                PWM1.start(0)
                GPIO.output(mov_motor,1)
                caso=0
                caso2=0
                print "sensor final de carrera"
            if GPIO.input(sensor_quinto)==GPIO.HIGH or
GPIO.input(sensor_cuarto)==GPIO.HIGH or
GPIO.input(sensor_tercero)==GPIO.HIGH:#esta en el 5,4,3
                GPIO.output(giro,1)
                accion_envio=2
                ok=1
```

```
            if GPIO.input(sensor_primero)==GPIO.HIGH:#esta en el 1
               GPIO.output(giro,0)
               accion_envio=1
               ok=1
            if GPIO.input(sensor_segundo)==GPIO.LOW and ok==1:
               GPIO.output(mov_motor,0)
            else:
               GPIO.output(mov_motor,1)
               accion_envio=0
               ok=0
               caso=0
            #segundo Elevador
            if GPIO.input(sensor_quinto2)==GPIO.HIGH or
GPIO.input(sensor_cuarto2)==GPIO.HIGH or
GPIO.input(sensor_tercero2)==GPIO.HIGH:#esta en el 5,4,3
               accion_envio2=2
               ok2=2
            if GPIO.input(sensor_primero2)==GPIO.HIGH:#esta en el 1
               GPIO.output(giro,0)
               accion_envio2=1
               ok2=1
            if GPIO.input(sensor_segundo2)==GPIO.LOW and ok2==1:#no esta en
tercero y verifica que todo esta bien
               GPIO.output(FWD,0)
               GPIO.output(REV,1)
            else:
               if GPIO.input(sensor_segundo2)==GPIO.LOW and ok2==2:
                  GPIO.output(FWD,1)
                  GPIO.output(REV,0)
               else:
                  GPIO.output(FWD,1)
                  GPIO.output(REV,1)
                  accion_envio2=0
                  ok2=0
                  caso2=0
         while(caso==1 or caso2==1):#ir al primer piso
            #datos enviados
            if GPIO.input(sensor_primero)==GPIO.HIGH:
               piso_envio=0
            if GPIO.input(sensor_segundo)==GPIO.HIGH:
               piso_envio=1
            if GPIO.input(sensor_tercero)==GPIO.HIGH:
               piso_envio=2
            if GPIO.input(sensor_cuarto)==GPIO.HIGH:
               piso_envio=3
            if GPIO.input(sensor_quinto)==GPIO.HIGH:
               piso_envio=4
            if GPIO.input(sensor_primero2)==GPIO.HIGH:
               piso_envio2=0
            if GPIO.input(sensor_segundo2)==GPIO.HIGH:
```

```
            piso_envio2=1
        if GPIO.input(sensor_tercero2)==GPIO.HIGH:
            piso_envio2=2
        if GPIO.input(sensor_cuarto2)==GPIO.HIGH:
            piso_envio2=3
        if GPIO.input(sensor_quinto2)==GPIO.HIGH:
            piso_envio2=4
        client_sock.send("X")
        time.sleep(0.010)
        client_sock.send("_")
        client_sock.send(str(piso_envio))
        client_sock.send("_")
        client_sock.send(str(piso_envio2))
        client_sock.send("_")
        client_sock.send(str(alarma_envio))
        client_sock.send("_")
        client_sock.send(str(alarma_envio2))
        client_sock.send("_")
        client_sock.send(str(accion_envio))
        client_sock.send("_")
        client_sock.send(str(accion_envio2))
        client_sock.send("_")
        client_sock.send(str(pulsos))
        #Primer Elevador
        if GPIO.input(sensor_fin_carrera)==GPIO.HIGH or
GPIO.input(sensor_fin_carrera2)==GPIO.HIGH:
            alarma_envio=1
            iniciar=2
            accion_envio=0
            PWM1.start(0)
            GPIO.output(mov_motor,1)
            caso=0
            caso2=0
            print "sensor final de carrera"
        if GPIO.input(sensor_quinto)==GPIO.HIGH or
GPIO.input(sensor_cuarto)==GPIO.HIGH or
GPIO.input(sensor_tercero)==GPIO.HIGH or
GPIO.input(sensor_segundo)==GPIO.HIGH:#esta en el 5,4,3,2
            GPIO.output(giro,1)
            accion_envio=2
            ok=1
        if GPIO.input(sensor_primero)==GPIO.LOW and ok==1:
            GPIO.output(mov_motor,0)
        else:
            GPIO.output(mov_motor,1)
            accion_envio=0
            ok=0
            caso=0
        #Primer Elevador
```

```
            if GPIO.input(sensor_quinto2)==GPIO.HIGH or
GPIO.input(sensor_cuarto2)==GPIO.HIGH or
GPIO.input(sensor_tercero2)==GPIO.HIGH or
GPIO.input(sensor_segundo2)==GPIO.HIGH:#esta en el 5,4,3,2
                accion_envio2=2
                ok2=2
            if GPIO.input(sensor_primero2)==GPIO.LOW and ok2==1:#no esta en
tercero y verifica que todo esta bien
                GPIO.output(FWD,0)
                GPIO.output(REV,1)
            else:
                if GPIO.input(sensor_primero2)==GPIO.LOW and ok2==2:
                    GPIO.output(FWD,1)
                    GPIO.output(REV,0)
                else:
                    GPIO.output(FWD,1)
                    GPIO.output(REV,1)
                    accion_envio2=0
                    ok2=0
                    caso2=0
        if(iniciar==2):
            PWM1.start(0)
            PWM2.start(0)
            GPIO.output(mov_motor,1)
            GPIO.output(FWD,1)
            GPIO.output(REV,1)
        if(iniciar==3):
            PWM1.start(ref_vel*2)
except IOError:
    pass
print("desconectado")
PWM1.start(0)
PWM2.start(0)
GPIO.output(mov_motor,1)
GPIO.output(FWD,1)
GPIO.output(REV,1)
GPIO.output(giro,1)
client_sock.close()
server_sock.close()
print("todo correcto")
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PROTOTIPO DE UN ASCENSOR Y SU CONTROL COMPARANDO TECNOLOGÍAS DIFERENTES DE ACCIONAMIENTO

ANEXO IX: CÓDIGO FUENTE ANDROID

```
package upv.kevinbarrera.controlbluetooth;
import android.annotation.SuppressLint;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.ImageButton;
import android.widget.ScrollView;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.Date;
import java.util.UUID;
import android.view.Menu;
import android.widget.ToggleButton;
import java.util.UUID;
public class control extends AppCompatActivity implements
SeekBar.OnSeekBarChangeListener{

    //inicializo variables normales
    Button bsalir,barriba,babajo,bdesconectarBT;
    Button bgiroHmanas,bgiroAmanas,bmarchmanas,bparomanas;
    ToggleButton FWD,REV;
    ImageButton bprimero,bsegundo,btercero,bcuarto,bquinto;
    TextView txtvelocidad;
    //inicializo variables bluetooth
    Handler bluetoothIn;
    final int handlerState=0;
    private BluetoothAdapter btAdapter=null;
    private BluetoothSocket btStocket=null;
    private StringBuilder DataStringIn=new StringBuilder();
    private ConnectedThread MyConexionBT;
    private static String readMessage;
    //INICIALIZO DIRECCION MAC
    private static final UUID BTMODULEUUID=UUID.fromString("00001101-0000-
1000-8000-00805F9B34FB");
```

```
    //private static final UUID BTMODULEUUID=UUID.fromString("94f39d29-7d6d-
437d-973b-fba39e49d4ee");
    private static String address=null;
    private static String dato1="25";
    private static String datorecibido="";
    private static String registro="";
    private static String ul1="";
    private static String ul2="";
    private static String historial="";
    private static String datocompleto="";
    private static String []cadena1={"0-0-0"};
    private static String piso="";
    private static String alarma="";
    private static String frecuencia="";
    private static String suba="";
    private static String piso2="";
    private static String alarma2="";
    private static String suba2="";
    private SeekBar barravelocidad;
    private SeekBar barravelocidadserv;
    private SeekBar barravelocidadservman;
    private SeekBar barravelocidadmanas;
    private SeekBar barrapisos;
    int progreso_vel=25;
    int progreso_vel_serv=0;
    int n_pisos=4;
    TextView txtbarra_pisos;
    TextView txtdato_barra2;
    TextView txtregistro;
    TextView txthistorial;
    TextView txtestado;
    TextView txtvelmanas;
    TextView ul12;
    TextView txtvelmanserv;


    @SuppressLint("WrongViewCast")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_control);

        time time=new time();
        time.execute();
        //inicializo y enlazo botones
        bsalir = (Button) findViewById(R.id.b_salir_control);//identificacion del boton
        barriba = (Button) findViewById(R.id.b_arriba);
        babajo = (Button) findViewById(R.id.b_abajo);
        bmarchmanas = (Button) findViewById(R.id.b_marcha_man_as);
        bparomanas = (Button) findViewById(R.id.b_paro_man_as);
```

```
        bgiroHmanas = (Button) findViewById(R.id.b_giroH_man_as);
        bgiroAmanas = (Button) findViewById(R.id.b_giroA_man_as);
        bprimero = (ImageButton) findViewById(R.id.img_ceropiso);
        bsegundo = (ImageButton) findViewById(R.id.img_primerpiso);
        btercero = (ImageButton) findViewById(R.id.img_segundopiso);
        bcuarto = (ImageButton) findViewById(R.id.img_tercerpiso);
        bquinto = (ImageButton) findViewById(R.id.img_cuartopiso);
        bdesconectarBT = (Button) findViewById(R.id.b_desconectarBT);
        barravelocidad = (SeekBar)findViewById(R.id.barra1);//llamada a HMI de barra
velocidad
        barravelocidad.setOnSeekBarChangeListener(this);//compilación para que barra
funcione en esta ventana
        barravelocidadmanas = (SeekBar)findViewById(R.id.barra_vel_man_as);//llamada
a HMI de barra velocidad manual asincrono
        barravelocidadmanas.setOnSeekBarChangeListener(this);//compilación para que
barra funcione en esta ventana
        barrapisos = (SeekBar)findViewById(R.id.barrapisos);//llamada a HMI de barra
pisos
        barrapisos.setOnSeekBarChangeListener(this);//compilación para que barra
funcione en esta ventana
        barravelocidadserv = (SeekBar)findViewById(R.id.barra2);//llamada a HMI de
velocidad servomotor
        barravelocidadserv.setOnSeekBarChangeListener(this);//compilación para que
barra funcione en esta ventana
        barravelocidadservman =
(SeekBar)findViewById(R.id.barra_vel_man_serv);//llamada a HMI de velocidad
manual servomotor
        barravelocidadservman.setOnSeekBarChangeListener(this);//compilación para que
barra funcione en esta ventana
        txtvelocidad = (TextView) findViewById(R.id.txtvelocidadbt);
        txtbarra_pisos=(TextView) findViewById(R.id.txtbarrapisos);
        txtdato_barra2=(TextView) findViewById(R.id.txtdatobarra2);
        txtregistro=(TextView) findViewById(R.id.txt_reg);
        txthistorial=(TextView) findViewById(R.id.txt_histo);
        txtestado=(TextView) findViewById(R.id.txt_estado);
        txtvelmanas=(TextView) findViewById(R.id.txt_vel_man);
        txtvelmanserv=(TextView) findViewById(R.id.txt_vel_man_serv);
        ul12=(TextView) findViewById(R.id.txt_vis);
        FWD= (ToggleButton) findViewById(R.id.tb_FWD);
        REV= (ToggleButton) findViewById(R.id.tb_REV);



        bluetoothIn = new Handler() {
            public void handleMessage(android.os.Message msg) {
                if (msg.what == handlerState) {
                    String readMessage = (String) msg.obj;
                    DataStringIn.append(readMessage);

                    int endOfLineIndex = DataStringIn.indexOf("#");
```

```
                //   if (endOfLineIndex > 0) {
                //   String dataInPrint = DataStringIn.substring(0, endOfLineIndex);
                //  txtbuffer.setText("Dato: " + endOfLineIndex);//<-<- PARTE A
MODIFICAR >->->
                //DataStringIn.delete(0, DataStringIn.length());
                //}
            }
        }
    };

    btAdapter = BluetoothAdapter.getDefaultAdapter();
    VerificarEstadoBT();

    bsalir.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            java.util.Date fecha = new Date();
            registro=fecha+" Salió de la App"+"\n"+registro;//error sensor fin de carrera
            //-------ir a seguridad----------
            Intent saltoaseguridad = new Intent(control.this, seguridad.class);
            startActivity(saltoaseguridad);
            //-------fin ir a seguridad----------
        }
    });


    barriba.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
          // enviar_buffer[0]="s";
            //---------historial--------------------
            java.util.Date fecha = new Date();
            historial=fecha+":Pulsó ENCENDIDO"+"\n"+historial;
            //---------texto estado--------------------
            txtestado.setText("ESTADO:ENCENDIDO");
            txtestado.setTextColor(Color.rgb(0, 150, 0));

            MyConexionBT.write("i");
            barriba.setBackgroundColor(Color.GREEN);
            babajo.setBackgroundColor(Color.LTGRAY);
        }
    });

    babajo.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //enviar_buffer[0]="p";
            //---------historial--------------------
            java.util.Date fecha = new Date();
```

```
        historial=fecha+":Pulsó APAGADO"+"\n"+historial;
        //---------texto estado--------------------
        txtestado.setText("ESTADO:APAGADO");
        txtestado.setTextColor(Color.rgb(0, 0, 0));
        MyConexionBT.write("p");
        babajo.setBackgroundColor(Color.RED);
        barriba.setBackgroundColor(Color.LTGRAY);
    }
});


bmarchmanas.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // enviar_buffer[0]="s";
        //---------historial--------------------
        java.util.Date fecha = new Date();
        historial=fecha+":Pulsó marcha manual"+"\n"+historial;
        //---------texto estado--------------------
        txtestado.setText("ESTADO:ENCENDIDO");
        txtestado.setTextColor(Color.rgb(0, 150, 0));
        MyConexionBT.write("I");
        bmarchmanas.setBackgroundColor(Color.GREEN);
        bparomanas.setBackgroundColor(Color.LTGRAY);
    }
});

bparomanas.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //enviar_buffer[0]="p";
        //---------historial--------------------
        java.util.Date fecha = new Date();
        historial=fecha+":Pulsó paro manual"+"\n"+historial;
        //---------texto estado--------------------
        txtestado.setText("ESTADO:APAGADO");
        txtestado.setTextColor(Color.rgb(0, 0, 0));
        MyConexionBT.write("P");
        bparomanas.setBackgroundColor(Color.RED);
        bmarchmanas.setBackgroundColor(Color.LTGRAY);
    }
});

bgiroHmanas.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //enviar_buffer[0]="p";
        //---------historial--------------------
        java.util.Date fecha = new Date();
        historial=fecha+":Dirección manual arriba"+"\n"+historial;
```

```
            //---------texto estado--------------------
            MyConexionBT.write("g");
        }
    });
    bgiroAmanas.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //enviar_buffer[0]="p";
            //---------historial--------------------
            java.util.Date fecha = new Date();
            historial=fecha+":Dirección manual abajo"+"\n"+historial;
            //---------texto estado--------------------
            MyConexionBT.write("G");
        }
    });
    bprimero.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //registro=fecha+":Pulso primer piso"+"\n"+registro;//---------------historial----
---------------
            //---------historial--------------------
            java.util.Date fecha = new Date();
            historial=fecha+":Pulsó Planta Baja"+"\n"+historial;
            //*****fin para historial*****************************************
            //enviar_buffer[2]="1";
            MyConexionBT.write("u");
           // bprimero.setBackgroundColor(Color.CYAN);
           // bsegundo.setBackgroundColor(Color.LTGRAY);
           // btercero.setBackgroundColor(Color.LTGRAY);
        }
    });

    bsegundo.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //enviar_buffer[2]="2";
            //---------historial--------------------
            java.util.Date fecha = new Date();
            historial=fecha+":Pulsó Primera Planta"+"\n"+historial;
            //*****fin para historial*****************************************
            MyConexionBT.write("d");
           // bprimero.setBackgroundColor(Color.LTGRAY);
           // bsegundo.setBackgroundColor(Color.CYAN);
           // btercero.setBackgroundColor(Color.LTGRAY);
        }
    });
    btercero.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
          // enviar_buffer[2]="3";
```

```
            //---------historial--------------------
            java.util.Date fecha = new Date();
            historial=fecha+":Pulsó Segunda Planta"+"\n"+historial;
            //*****fin para historial*****************************************
            MyConexionBT.write("t");
           //  bprimero.setBackgroundColor(Color.LTGRAY);
            // bsegundo.setBackgroundColor(Color.LTGRAY);
            // btercero.setBackgroundColor(Color.CYAN);
        }
    });

    bcuarto.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // enviar_buffer[2]="3";
            //---------historial--------------------
            java.util.Date fecha = new Date();
            historial=fecha+":Pulsó Tercera Planta"+"\n"+historial;
            //*****fin para historial*****************************************
            MyConexionBT.write("c");
            //  bprimero.setBackgroundColor(Color.LTGRAY);
            // bsegundo.setBackgroundColor(Color.LTGRAY);
            // btercero.setBackgroundColor(Color.CYAN);
        }
    });

    bquinto.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // enviar_buffer[2]="3";
            //---------historial--------------------
            java.util.Date fecha = new Date();
            historial=fecha+":Pulsó Cuarta planta"+"\n"+historial;
            //*****fin para historial*****************************************
            MyConexionBT.write("q");
            //  bprimero.setBackgroundColor(Color.LTGRAY);
            // bsegundo.setBackgroundColor(Color.LTGRAY);
            // btercero.setBackgroundColor(Color.CYAN);
        }
    });


    FWD.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
            if (isChecked) {
                MyConexionBT.write("m");
                //---------historial--------------------
                java.util.Date fecha = new Date();
```

```
                historial=fecha+":FWD on"+"\n"+historial;
                REV.setEnabled(false);
                //*****fin para historial*****************************************
            } else {
                //---------historial--------------------
                java.util.Date fecha = new Date();
                historial=fecha+":FWD off"+"\n"+historial;
                MyConexionBT.write("M");
                REV.setEnabled(true);
            }
        }
    });

    REV.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
            if (isChecked) {
                MyConexionBT.write("n");
                //---------historial--------------------
                java.util.Date fecha = new Date();
                historial=fecha+":REV on"+"\n"+historial;
                //
                FWD.setEnabled(false);
            } else {
                MyConexionBT.write("N");
                //---------historial--------------------
                java.util.Date fecha = new Date();
                historial=fecha+":REV off"+"\n"+historial;
                FWD.setEnabled(true);
            }
        }
    });



    bdesconectarBT.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            java.util.Date fecha = new Date();
            registro=fecha+" Desconexión de Bluetooth"+"\n"+registro;//error sensor fin
de carrera
            if (btStocket!=null)
            {
                try {btStocket.close();}
                catch (IOException e)
                { Toast.makeText(getBaseContext(), "Error",
Toast.LENGTH_SHORT).show();;}
            }
            finish();
```

```
        }
    });
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate( R.menu.control_menu,menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch(item.getItemId()){
        case R.id.ascensor:
            findViewById(R.id.HMI_config).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_alarmas).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_historial).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_mantenimiento).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_ascensor).setVisibility(View.VISIBLE);
            return true;
        case R.id.alarmas:
            findViewById(R.id.HMI_alarmas).setVisibility(View.VISIBLE);
            findViewById(R.id.HMI_config).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_historial).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_mantenimiento).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_ascensor).setVisibility(View.INVISIBLE);
            return true;
        case R.id.config:
            findViewById(R.id.HMI_alarmas).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_config).setVisibility(View.VISIBLE);
            findViewById(R.id.HMI_historial).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_mantenimiento).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_ascensor).setVisibility(View.INVISIBLE);
            return true;
        case R.id.historial:
            findViewById(R.id.HMI_alarmas).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_config).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_historial).setVisibility(View.VISIBLE);
            findViewById(R.id.HMI_mantenimiento).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_ascensor).setVisibility(View.INVISIBLE);
            return true;
        case R.id.mantenimiento:
            findViewById(R.id.HMI_alarmas).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_config).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_historial).setVisibility(View.INVISIBLE);
            findViewById(R.id.HMI_mantenimiento).setVisibility(View.VISIBLE);
            findViewById(R.id.HMI_ascensor).setVisibility(View.INVISIBLE);
            return true;
        default:
            return super.onOptionsItemSelected(item);
```

```
        }

    }

    private BluetoothSocket createBluetoothSocket(BluetoothDevice device) throws
IOException
    {
        //crea un conexion de salida segura para el dispositivo
        //usando el servicio UUID
        return device.createRfcommSocketToServiceRecord(BTMODULEUUID);



    }


    //==================tiempo de espera del periodo de
muestreo======================
    public void hilo() //llama a la funcion para que espere el llamado de 1 segundo
    {
        try {
            Thread.sleep(50);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

    }
    public void ejecutar()
    {
        time time=new time();
        time.execute();
        ((TextView)findViewById(R.id.txtdatobarra)).setText(progreso_vel+" Hz");
        txtvelmanas.setText(progreso_vel+" Hz");
        txtbarra_pisos.setText(n_pisos+"");
        txtregistro.setText(registro);
        txtdato_barra2.setText(progreso_vel_serv+" rpm");
        txtvelmanserv.setText(progreso_vel_serv+" rpm");
        if(registro.length()>2500)
        {
            registro="";
        }
        txthistorial.setText(historial);
        if(historial.length()>2500)
        {
            historial="";
        }
        switch(n_pisos){
            case 1:
                findViewById(R.id.img_cuartopiso).setVisibility(View.INVISIBLE);
```

```
            findViewById(R.id.img_tercerpiso).setVisibility(View.INVISIBLE);
            findViewById(R.id.img_segundopiso).setVisibility(View.INVISIBLE);
            findViewById(R.id.img_primerpiso).setVisibility(View.VISIBLE);
            return;
        case 2:
            findViewById(R.id.img_cuartopiso).setVisibility(View.INVISIBLE);
            findViewById(R.id.img_tercerpiso).setVisibility(View.INVISIBLE);
            findViewById(R.id.img_segundopiso).setVisibility(View.VISIBLE);
            findViewById(R.id.img_primerpiso).setVisibility(View.VISIBLE);
            return;
        case 3:
            findViewById(R.id.img_cuartopiso).setVisibility(View.INVISIBLE);
            findViewById(R.id.img_tercerpiso).setVisibility(View.VISIBLE);
            findViewById(R.id.img_segundopiso).setVisibility(View.VISIBLE);
            findViewById(R.id.img_primerpiso).setVisibility(View.VISIBLE);
            return;
        case 4:
            findViewById(R.id.img_cuartopiso).setVisibility(View.VISIBLE);
            findViewById(R.id.img_tercerpiso).setVisibility(View.VISIBLE);
            findViewById(R.id.img_segundopiso).setVisibility(View.VISIBLE);
            findViewById(R.id.img_primerpiso).setVisibility(View.VISIBLE);
            return;

    }


   /* for(int v=0;v<6;v++) {
        MyConexionBT.write(enviar_buffer[v]);
        System.out.println(enviar_buffer[v]);
    }*/
}
public class time extends AsyncTask<Void,Integer,Boolean>{

    @Override
    protected Boolean doInBackground(Void... voids) {
        //for (int i=1;i<2;i++)
        //{
        hilo();
        //}
        return true;
    }

    @Override
    protected void onPostExecute(Boolean aBoolean) {
        ejecutar();
        //Toast.makeText(control.this,"Actualizar",Toast.LENGTH_SHORT).show();
        cadena1 = datocompleto.split("_");
        //System.out.println(datocompleto);
        if (cadena1.length>=7) {
            piso = cadena1[1];//piso sensor1
```

```
            piso2 = cadena1[2];//piso sensor2
            alarma= cadena1[3];//alarma fallo en sensores1
            alarma2= cadena1[4];//alarma fallo en sensores2
            suba= cadena1[5];//accion unidad lineal1
            suba2= cadena1[6];//accion unidad lineal2
            frecuencia= cadena1[7];//frecuencia induccion
          }

          if(suba.equals("0")){
            ul1=piso;
            //((TextView) findViewById(R.id.txt_vis)).setText(piso);
          }
          if(suba.equals("1")){
            ul1="↑"+piso;
            //((TextView) findViewById(R.id.txt_vis)).setText("↑"+piso);
          }
          if(suba.equals("2")){
            ul1="↓"+piso;
            //((TextView) findViewById(R.id.txt_vis)).setText("↓"+piso);
          }
          if(alarma.equals("1")){
            ul1="F1";
            //((TextView) findViewById(R.id.txt_vis)).setText("FIN");
            java.util.Date fecha = new Date();
            registro=fecha+" Se activo sensor fin de carrera 1"+"\n"+registro;//error
sensor fin de carrera
            babajo.setBackgroundColor(Color.RED);
            barriba.setBackgroundColor(Color.LTGRAY);
            bparomanas.setBackgroundColor(Color.RED);
            bmarchmanas.setBackgroundColor(Color.LTGRAY);
            MyConexionBT.write("p");
          }

          if(suba2.equals("0")){
            ul2=piso2;
            //((TextView) findViewById(R.id.txt_vis2)).setText(piso2);
          }
          if(suba2.equals("1")){
            ul2="↑"+piso2;
            //((TextView) findViewById(R.id.txt_vis2)).setText("↑"+piso2);
          }
          if(suba2.equals("2")){
            ul2="↓"+piso2;
            //((TextView) findViewById(R.id.txt_vis2)).setText("↓"+piso2);
          }
          if(alarma2.equals("1")){
            ul2="F2";
            //((TextView) findViewById(R.id.txt_vis2)).setText("FIN");
            java.util.Date fecha = new Date();
```

```
            registro=fecha+" Se activo sensor fin de carrera 2"+"\n"+registro;//error
sensor fin de carrera
            babajo.setBackgroundColor(Color.RED);
            barriba.setBackgroundColor(Color.LTGRAY);
            bparomanas.setBackgroundColor(Color.RED);
            bmarchmanas.setBackgroundColor(Color.LTGRAY);
            MyConexionBT.write("p");
         }
         ul12.setText(ul2+"    "+ul1);
         ((TextView) findViewById(R.id.txtvelocidadbt)).setText(frecuencia+"Hz");

      }
   }
   //==================fin tiempo de espera del periodo de
muestreo======================

   public void onResume()
   {


      super.onResume();
      //Consigue la direccion MAC desde DeviceListActivity via intent
      Intent intent = getIntent();
      //Consigue la direccion MAC desde DeviceListActivity via EXTRA
      address =
intent.getStringExtra(dispositivosBT.EXTRA_DEVICE_ADDRESS);//<-<- PARTE A
MODIFICAR >->->
      //Setea la direccion MAC
      BluetoothDevice device = btAdapter.getRemoteDevice(address);

      try
      {
        btStocket = createBluetoothSocket(device);
      } catch (IOException e) {
         Toast.makeText(getBaseContext(), "La creacción del Socket fallo",
Toast.LENGTH_LONG).show();
      }
      // Establece la conexión con el socket Bluetooth.
      try
      {
         btStocket.connect();
      } catch (IOException e) {
         try {
            btStocket.close();
         } catch (IOException e2) {}
      }
      MyConexionBT = new ConnectedThread(btStocket);
      MyConexionBT.start();

   }
```

```
    @Override
    public void onPause()
    {
        super.onPause();
        try
        { // Cuando se sale de la aplicación esta parte permite
            // que no se deje abierto el socket
            btStocket.close();
        } catch (IOException e2) {}
    }

    //Comprueba que el dispositivo Bluetooth Bluetooth está disponible y solicita que se
active si está desactivado
    private void VerificarEstadoBT() {

        if(btAdapter==null) {
            Toast.makeText(getBaseContext(), "El dispositivo no soporta bluetooth",
Toast.LENGTH_LONG).show();
        } else {
            if (btAdapter.isEnabled()) {
            } else {
                Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent, 1);
            }
        }
    }
//------------barra deslizante----------------------------------------------------------

    @Override
    public void onProgressChanged(SeekBar barra, int progress, boolean fromUser) {
        //((TextView)findViewById(R.id.txtdatobarra)).setText("Velocidad:"+progress+"
Hz");

        if (barra.equals(barravelocidad) ||  barra.equals(barravelocidadmanas)) {
            progreso_vel = progress;
        }
        if(barra.equals(barrapisos)){
            n_pisos=progress+1;
        }
        if(barra.equals(barravelocidadserv) || barra.equals(barravelocidadservman)){
            progreso_vel_serv=progress*100;
        }

       // dato1=String.valueOf(progress);
        //MyConexionBT.write(dato1);

    }
    @Override
```

```
    public void onStartTrackingTouch(SeekBar barra) {

 //((TextView)findViewById(R.id.txtaccion)).setText("Inicio:"+seekBar.getProgress());

    }
    @Override
    public void onStopTrackingTouch(SeekBar barra) {
       //((TextView)findViewById(R.id.txtaccion)).setText("Se
 detuvo:"+seekBar.getProgress());
       if (barra.equals(barravelocidad) ||  barra.equals(barravelocidadmanas)){
          dato1 = String.valueOf(barra.getProgress());
       //enviar_buffer[4]=dato1;
       MyConexionBT.write(dato1);
          //---------historial--------------------
          java.util.Date fecha = new Date();
          historial=fecha+":Velocidad Variador:"+dato1+"\n"+historial;
          //*****fin para historial*****************************************
           }

       if(barra.equals(barrapisos)){

             //---------historial--------------------
             java.util.Date fecha = new Date();
             historial=fecha+":Nro de pisos:"+n_pisos+"\n"+historial;
             //*****fin para historial*****************************************

       }
       if(barra.equals(barravelocidadserv)|| barra.equals(barravelocidadservman)){
          //---------historial--------------------
          java.util.Date fecha = new Date();
          historial=fecha+":Velocidad Servomotor:"+progreso_vel_serv+"\n"+historial;
          if (progreso_vel_serv==300){
             MyConexionBT.write("B");
          }
          if (progreso_vel_serv==200){
             MyConexionBT.write("E");
          }
          if (progreso_vel_serv==100){
             MyConexionBT.write("K");
          }
          if (progreso_vel_serv==0){
             MyConexionBT.write("k");
          }
       }



    }
```

```
    //---------fin barra deslizante----------------------------------------------------------

    //Crea la clase que permite crear el evento de conexion
    private class ConnectedThread extends Thread
    {
        private final InputStream mmInStream;
        private final OutputStream mmOutStream;

        public ConnectedThread(BluetoothSocket socket)
        {
            InputStream tmpIn = null;
            OutputStream tmpOut = null;
            try
            {
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e) { }
            mmInStream = tmpIn;
            mmOutStream = tmpOut;
        }

        public void run()
        {
            byte[] buffer = new byte[256];
            int bytes;
            // Se mantiene en modo escucha para determinar el ingreso de datos
            while (true) {
                try {
                    bytes = mmInStream.read(buffer);
                    readMessage = new String(buffer, 0, bytes);
                    if (readMessage.equals("X")){
                        datocompleto=datorecibido;
                        //System.out.println(datocompleto);
                        datorecibido = "";
                    }
                    datorecibido = datorecibido + readMessage;//almacena todos los datos
recibidos

                    // Envia los datos obtenidos hacia el evento via handler
                    bluetoothIn.obtainMessage(handlerState, bytes, -1,
readMessage).sendToTarget();


                } catch (IOException e) {
                    break;
                }
            }
        }

        //Envio de trama
```

```
        public void write(String input)
        {
            try {
                mmOutStream.write(input.getBytes());
            }
            catch (IOException e)
            {
                //si no es posible enviar datos se cierra la conexión
                //Toast.makeText(getBaseContext(), "NO SE ESTABLECIÓ CONEXIÓN
BLUETOOTH", Toast.LENGTH_LONG).show();
                finish();
            }
        }

    }
}
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PROTOTIPO DE UN ASCENSOR Y SU CONTROL COMPARANDO TECNOLOGÍAS DIFERENTES DE ACCIONAMIENTO

ANEXO X: CÓDIGO FUENTE OSCILOSCOPIO

```
#include <SPI.h> // libreria de comunicación con el reloj del micro SPI
#include <HardwareSerial.h>//hardware serial para definir dataout,datain, con el fin de
no utilizar
//liberias externas porque el programa funcionaba a una velocidad baja, definimos los
pines
// Para LOLIN I2C PRO, esos son los pines de comunicación, pero usando el Hardware
serial se pueden
//designar otros pines, sin embargo para garantizar el funcionamiento de los pines,se
elijen por defecto

#define SELPIN 13    // chip-select
#define DATAOUT 23   // MOSI
#define DATAIN 19    // MISO
#define SPICLOCK 18  // Clock SCK
//====================variables de interrupciones
TIME=======================//
static volatile uint16_t interruptCounter=0;
int totalInterruptCounter;
hw_timer_t * timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;
//====================fin variables de interrupciones
TIME=======================//
//====================variables de interrupciones
EXTERNA=======================//
const byte interruptPin = 15;//PIN SCL
volatile int interruptCounter_ext = 0;
uint32_t numberOfInterrupts = 0;
portMUX_TYPE mux = portMUX_INITIALIZER_UNLOCKED;
uint32_t frecuencia;
uint32_t velocidad;
uint32_t vel_suma;
uint32_t vel_prom;
uint32_t int_ext_aux=0;
//====================fin variables de interrupciones
EXTERNA=======================//
//====================ENTRADA DE LOS
CANALES=======================//
int readvalue;
long valor_leido_CH1;
long valor_leido_CH2;
long valor_leido_CH3;
//====================FIN ENTRADA DE LOS
CANALES====================//
//====================VARIABLE SERIAL
BLUETOOHT====================//
int ledPin=5;//5--2
String palabra;//palabra serial
```

```
char letra;//letra serial
//====================FIN VARIABLE SERIAL
BLUETOOHT====================//
//====================RESISTENCIAS PARA DIVISORES DE
TENSIÓN====================//
float r1_CH1 = 99200; // 100K 99100 esta RELACION ES MAXIMO 35v
float r2_CH1 = 9870; // 10K 9850
float r1_CH2 = 99100; // 100K
float r2_CH2 = 10070; // 10K
//====================FIN RESISTENCIAS PARA DIVISORES DE
TENSIÓN====================//

//====================VARIABLES PARA
MUESTRAS====================//
int muestras[334];
int muestras2[334];
int muestras3[334];
int T_muestras=334;
int cont_muestras=0;
float cuadra1=0;
float cuadra2=0;
float cuadra3=0;
float V_RMS1=0;
float V_RMS2=0;
float V_RMS3=0;
float offset=2070;//2.5 V del sensor aproximadamente
float sensi=1/104.6;//sensibilidad del sensor
float relacionpar=15.5/14.4;
float par=0;
int contCH2=0;
int contCH3=0;
float R_intensidad=0;
float I_RMS=0,absoluto=0;

//====================FIN VARIABLES PARA
MUESTRAS====================//
//====================VARIABLES PARA
RESULTADO====================//
int prom=0;
float I_RMS_prom=0;
float V_RMS_prom=0;
float Par_prom=0;
int grafON=0;
//====================FIN VARIABLES PARA
RESULTADO====================//
//====================void de interrupciones
TIME========================//
void IRAM_ATTR onTimer() {
  //portENTER_CRITICAL_ISR(&timerMux);
  interruptCounter++;
```

```
  //portEXIT_CRITICAL_ISR(&timerMux);
 }
 //====================fin void de interrupciones
 TIME=========================//
 //====================void de interrupciones
 EXTERNAS=========================//
 void IRAM_ATTR handleInterrupt() {
  // portENTER_CRITICAL_ISR(&mux);
   interruptCounter_ext++;
  // portEXIT_CRITICAL_ISR(&mux);
 }
 //====================fin void de interrupciones
 EXTERNAS=========================//
 void setup(){
   //set pin modes
   pinMode(SELPIN, OUTPUT);
   // disable device to start with
   digitalWrite(SELPIN, HIGH);
   SPI.setClockDivider( SPI_CLOCK_DIV8 ); // slow the SPI bus down
   SPI.setBitOrder(MSBFIRST);
   SPI.setDataMode(SPI_MODE0);    // SPI 0,0 as per MCP330x data sheet
   SPI.begin();
   pinMode(ledPin,OUTPUT);//SALIDA DEL LED DE LA PLACA LOLIN I2C PRO
   pinMode(interruptPin, INPUT_PULLUP);//pin de interrupcion
   attachInterrupt(digitalPinToInterrupt(interruptPin), handleInterrupt,
 RISING);//interrupcion flanco ascendente,falling=descendente
   Serial.begin(9600);
   //====================inicio de
 interrupciones=========================//
   timer = timerBegin(0, 80, true); //80 Mhz..maximo 80 Mhz dividimos 80Mhz/80=1
 Mhz es decir 1000000 veces por segundo
   timerAttachInterrupt(timer, &onTimer, true);
   timerAlarmWrite(timer, 120, true);//interrupcion cada 1000000 =1 seg//40 us por canal
   timerAlarmEnable(timer);
  //====================fin inicio de
 interrupciones=========================//

 }
 int read_adc(int channel){
   int adcvalue = 0;
   int b1 = 0, b2 = 0;
   int sign = 0;

   // command bits for MCP3304
   // 0000 = diff, ch0 = in+, ch1 = in-
   // 0010 = diff, ch2 = in+, ch3 = in-
   // 0100 = diff, ch4 = in+, ch5 = in-

   digitalWrite (SELPIN, LOW); // Select adc
```

```
  // first byte
  // first byte will always be B000010xx where xx are the D2 and D1 channel bits
  byte commandbits = B00001000;
  commandbits |= (channel >> 1);        // high bit of channel

  SPI.transfer(commandbits);       // send out first byte of command bits

  // second byte; Bx0000000; leftmost bit is D0 channel bit
  commandbits = B00000000;
  commandbits |= (channel << 7);       // if D0 is set it will be the leftmost bit now
  b1 = SPI.transfer(commandbits);       // send out second byte of command bits

  // hi byte will have XX0SBBBB
  // set the top 3 don't care bits so it will format nicely
  b1 |= B11100000;
  //Serial.print(b1, BIN); Serial.print(" ");
  sign = b1 & B00010000;
  int hi = b1 & B00001111;

  // read low byte
  b2 = SPI.transfer(b2);             // don't care what we send
  //Serial.print(b2, BIN); Serial.print("\r\n");
  int lo = b2;
  digitalWrite(SELPIN, HIGH); // turn off device

  int reading = hi * 256 + lo;

  if (sign) {
    reading = (4096 - reading) * -1;
  }

  return (reading);
}

int read_adc1(int channel){
  int adcvalue = 0;
  int b1 = 0, b2 = 0;
  int sign = 0;

  // command bits for MCP3304
  // 0000 = diff, ch0 = in+, ch1 = in-
  // 0010 = diff, ch2 = in+, ch3 = in-
  // 0100 = diff, ch4 = in+, ch5 = in-

  digitalWrite (SELPIN, LOW); // Select adc

  // first byte
  // first byte will always be B000010xx where xx are the D2 and D1 channel bits
  byte commandbits = B00001000;
  commandbits |= (channel >> 1);        // high bit of channel
```

```
  SPI.transfer(commandbits);       // send out first byte of command bits

  // second byte; Bx0000000; leftmost bit is D0 channel bit
  commandbits = B00000000;
  commandbits |= (channel << 7);       // if D0 is set it will be the leftmost bit now
  b1 = SPI.transfer(commandbits);      // send out second byte of command bits

  // hi byte will have XX0SBBBB
  // set the top 3 don't care bits so it will format nicely
  b1 |= B11100000;
  //Serial.print(b1, BIN); Serial.print(" ");
  sign = b1 & B00010000;
  int hi = b1 & B00001111;

  // read low byte
  b2 = SPI.transfer(b2);             // don't care what we send
  //Serial.print(b2, BIN); Serial.print("\r\n");
  int lo = b2;
  digitalWrite(SELPIN, HIGH); // turn off device

  int reading = hi * 256 + lo;

  if (sign) {
    reading = (4096 - reading) * -1;
  }

  return (reading);
}

void loop(){

  valor_leido_CH1 = read_adc(0);// valor del canal 1
  valor_leido_CH2 = read_adc(2);// valor del canal 2
  valor_leido_CH3 = read_adc(4);// valor del canal 3
     if (interruptCounter > 0) {
       portENTER_CRITICAL(&timerMux);
       interruptCounter--;
       portEXIT_CRITICAL(&timerMux);
          if( cont_muestras <= T_muestras){
            attachInterrupt(digitalPinToInterrupt(interruptPin),
handleInterrupt,CHANGE);//ON INTERRUPCION EXTERNA,interrupts() HABILITA
                  muestras[cont_muestras]=valor_leido_CH1;
                  muestras2[cont_muestras]=valor_leido_CH2;
                  muestras3[cont_muestras]=valor_leido_CH3;
                  cont_muestras++;
          }else
          {  detachInterrupt(digitalPinToInterrupt(interruptPin));    //OFF
INTERRUPCION EXTERNA,noInterrupts() DESHABILITA
            numberOfInterrupts = interruptCounter_ext;
```

```
            lectura();
            cont_muestras =0;
            interruptCounter=0;
            interruptCounter_ext=0;
           }
      }
}


  void lectura(){
      if (Serial.read()=='G')
          {grafON=1-grafON;
           }
          //if (Serial.read()=='T')
          //{grafON=0;
          // }
 //-------------para calculos---------------------------------
       for(int i=1;i<cont_muestras;i++){
         if(muestras2[i]>offset){
          // Serial.println(muestras2[i]);
          cuadra2= cuadra2+pow(muestras2[i],2);
          contCH2++;
         }
          if(muestras3[i]>offset){
          // Serial.println(muestras2[i]);
          cuadra3= cuadra3+pow(muestras3[i],2);
          contCH3++;
         }
          cuadra1= cuadra1+pow(muestras[i],2);

       }




      V_RMS1=map(sqrt(cuadra1/334),0,4095,0,5000);
      V_RMS2=map(sqrt(cuadra2/contCH2)-offset,0,4095,0,5000);
      V_RMS3=map(sqrt(cuadra3/contCH3)-offset,0,4095,0,5000);
      I_RMS=V_RMS2*sensi;
      par=V_RMS3*sensi*relacionpar;

frecuencia=(numberOfInterrupts*1000000)/(334*120);//Hz=Int/Muestras*T_muestreo
      velocidad=((frecuencia*60)/1024)/2;
 //-------------fin para calculos---------------------------------


      //-------------RESULTADO---------------------------------

     if(prom<3){
       vel_suma=vel_suma+velocidad;
```

```
V_RMS_prom=V_RMS_prom+V_RMS1;
I_RMS_prom=I_RMS_prom+I_RMS;
Par_prom=Par_prom+par;
prom++;
}else
 {

  Serial.print("I");
    delay(5);
  Serial.print("_");
   if(grafON==0){
     Serial.print("0"); //VOLTAJE
     Serial.print("_");
     Serial.print("0"); //CORRIENTE
     Serial.print("_");
    }else{

      //para enviar grafica ch1
      for(int i=1;i<cont_muestras;i++){
        Serial.print(muestras[i]);//voltaje
        Serial.print("U");//voltaje
      }
     //para enviar grafica ch2
       Serial.print("_");
      for(int i=1;i<cont_muestras;i++){
       Serial.print(muestras2[i]);//voltaje
       Serial.print("D");//voltaje
      }
      Serial.print("_");

     }
     vel_prom=vel_suma/prom;
     if(vel_prom>=1400 && vel_prom<1600){ //se necesesita una ecuación de
ajuste a partir de una determinada velocidad esta es y=0.6623x+473.07
       Serial.print((vel_prom*0.6623)+475.07);
       Serial.print("_");
       }
       else
       {
        Serial.print(vel_prom);
        Serial.print("_");
        }
     Serial.print(V_RMS_prom/(prom*1000),3);//voltaje v
     Serial.print("_");
     Serial.print(I_RMS_prom/prom,3);//voltaje A
     Serial.print("_");
     Serial.println(Par_prom/prom,3);//voltaje Nm
     prom=0;
     vel_suma=0;
     vel_prom=0;
```

```
            V_RMS_prom=0;
            I_RMS_prom=0;
            Par_prom=0;
           // Serial.print("F");
          }


      //-------------FIN RESULTADO---------------------------------
      /*
frecuencia=(numberOfInterrupts*1000000)/(333.33*120);//Hz=Int/Muestras*T_muestr
eo
      Serial.print("Velocidad: ");
      Serial.print((frecuencia*60)/1024); //revisar para presicion
      int_ext_aux=0;
      Serial.println(" rpm");
      Serial.print("Volt
      aje CH1: ");
      Serial.println((maximo_ch1*3.3)/4096);
      Serial.print("Amperaje CH2: ");
      Serial.println((((maximo_ch2*3.3)/4096)-2.53)*(1/0.10416));
      Serial.print("Par CH3: ");
      Serial.println((((maximo_ch3*3.3)/4096)-2.53)*(1/0.10416));
      maximo_ch1=0;
      maximo_ch2=0;
      maximo_ch3=0; */
      //Serial.print("F");
      cuadra1=0;
      cuadra2=0;
      cuadra3=0;
      V_RMS1=0;
      V_RMS2=0;
      V_RMS3=0;
      interruptCounter=0;
      interruptCounter_ext=0;
      contCH2=0;
      contCH3=0;
      delay(1);
      //interrupts();
  }
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PROTOTIPO DE UN ASCENSOR Y SU CONTROL COMPARANDO TECNOLOGÍAS DIFERENTES DE ACCIONAMIENTO

ANEXO XI: CÓDIGO FUENTE AUTÓMATA PROGRAMABLE

```
PROGRAM PLC_PRG
VAR
	admin:STRING;
	password:STRING;
	ingresar:BOOL;
	menu:BOOL:=FALSE;
	giroup_man,girodwn_man,marcha_man,paro_man,FWD_man,REV_man:BOOL
;
	X1IM,X2IM:BOOL;
   X1PM,X2PM:BOOL;
	velIM:INT:=0;
	velPM:INT:=0;
	velIMasc:INT:=0;
	velPMasc:INT:=0;
   cantidad_pisos:INT:=4;
	ON:BOOL;
   OFF:BOOL;
	INICIAR: INT;
	piso_envio1,piso_envio2:INT:=0;
	caso:INT:=0;
	p_baja,p_primera,p_segunda,p_tercera,p_cuarta: BOOL;
	ok1:INT:=0;
	ok2:INT:=0;
	accion_envio1,accion_envio2: INT;
	alarma_envio1: INT;
   alarma_envio2: INT;
	alarma:STRING;
END_VAR

(*VERIFICA USUARIO*)
IF ingresar
THEN
	IF admin='admin' AND password='admin'
	THEN
	menu:=TRUE;
	ELSE
	admin:='incorrecto';
	password:='incorrecto';
   menu:=FALSE;
	END_IF
END_IF;

(*------------------------MODO MANUAL----------------------*)
IF giroup_man
THEN
iniciar:=3;
IMFWDREV:=TRUE;(*SALIDA GIRO IM*)
```

```
END_IF;
IF girodwn_man
THEN
iniciar:=3;
IMFWDREV:=FALSE;(*SALIDA GIRO IM*)
END_IF;
IF marcha_man
THEN
iniciar:=3;
IMMARCHAPARO:=TRUE;(*SALIDA MARCHA&PARO IM*)
END_IF;
IF paro_man
THEN
iniciar:=3;
IMMARCHAPARO:=FALSE;(*SALIDA MARCHA&PARO IM*)
END_IF;
(*VELOCIDAD INDUCCION*)
IF NOT X1IM AND NOT X2IM
THEN
velIM:=0;
IMX1:=FALSE;
IMX2:=FALSE;
END_IF;
IF X1IM AND NOT X2IM
THEN
velIM:=500;
IMX1:=TRUE;
IMX2:=FALSE;
END_IF
IF NOT X1IM AND X2IM
THEN
velIM:=1000;
IMX1:=FALSE;
IMX2:=TRUE;
END_IF;
IF X1IM AND X2IM
THEN
velIM:=1500;
IMX1:=TRUE;
IMX2:=TRUE;
END_IF;
IF FWD_man
THEN
iniciar:=3;
PMFWD:=TRUE;(*SALIDA GIRO PM*)
END_IF;
IF REV_man
THEN
iniciar:=3;
PMREV:=TRUE;(*SALIDA GIRO PM*)
```

```
END_IF;
(*VELOCIDAD SERVOMOTOR*)
IF NOT X1PM AND NOT X2PM
THEN
velPM:=0;
PMX1:=FALSE;
PMX2:=FALSE;
END_IF;
IF X1PM AND NOT X2PM
THEN
velPM:=500;
PMX1:=TRUE;
PMX2:=FALSE;
END_IF
IF NOT X1PM AND X2PM
THEN
velPM:=1000;
PMX1:=FALSE;
PMX2:=TRUE;
END_IF;
IF X1PM AND X2PM
THEN
velPM:=1500;
iniciar:=3;
PMX1:=TRUE;
PMX2:=TRUE;
END_IF;
(*-------------------------FIN MODO MANUAL---------------------*)
(*-------------------POSICION DEL ELEVADOR---------------------*)
IF IMind0
THEN
piso_envio1:=0;
END_IF;
IF IMind1
THEN
piso_envio1:=1;
END_IF;
IF IMind2
THEN
piso_envio1:=2;
END_IF;
IF IMind3
THEN
piso_envio1:=3;
END_IF;
IF IMind4
THEN
piso_envio1:=4;
END_IF;
IF PMind0
```

```
THEN
piso_envio2:=0;
END_IF;
IF PMind1
THEN
piso_envio2:=1;
END_IF;
IF PMind2
THEN
piso_envio2:=2;
END_IF;
IF PMind3
THEN
piso_envio2:=3;
END_IF;
IF PMind4
THEN
piso_envio2:=4;
END_IF;
(*-----------------FIN POSICION DEL ELEVADOR-------------------*)
(*------------------------ASCENSOR-----------------------------*)
IF p_baja
THEN
caso:=1;
END_IF;
IF p_primera
THEN
caso:=2;
END_IF;
IF p_segunda
THEN
caso:=3;
END_IF;
IF p_tercera
THEN
caso:=4;
END_IF;
IF p_cuarta
THEN
caso:=5;
END_IF;


IF ON
THEN
INICIAR:=1;
IMX1:=FALSE;
IMX2:=FALSE;
PMX1:=FALSE;
PMX2:=FALSE;
```

```
PMFWD:=FALSE;
PMREV:=FALSE;
END_IF;
IF INICIAR=1
THEN
CASE caso OF
5:
        IF IMind3 OR IMind2 OR IMind1 OR IMind0
        THEN
        IMFWDREV:=FALSE;(*giro arriba*)
        accion_envio1:=1;
        ok1:=1;
        END_IF;
        IF NOT IMind4 AND ok1=1
        THEN
        IMMARCHAPARO:=TRUE;(*prende induccion*)
        ELSE
        IMMARCHAPARO:=FALSE;(*apaga induccion*)
        accion_envio1:=0;
        ok1:=0;
        END_IF;

        IF PMind3 OR PMind2 OR PMind1 OR PMind0
        THEN
        accion_envio2:=1;
        ok2:=1;
        END_IF;
        IF NOT PMind4 AND ok2=1
        THEN
        PMFWD:=TRUE;(*giro arriba*)
        PMREV:=FALSE;(*giro abajo*)
        ELSE
                IF NOT PMind4 AND ok2=2
                THEN
           PMFWD:=FALSE;(*giro arriba*)
           PMREV:=TRUE;(*giro abajo*)
                ELSE
           PMFWD:=FALSE;(*giro arriba*)
           PMREV:=FALSE;(*giro abajo*)
           accion_envio2:=0;
           ok2:=0;
                END_IF;
        END_IF;
4:
        IF IMind4
        THEN
        IMFWDREV:=TRUE;(*giro abajo*)
        accion_envio1:=2;
        ok1:=1;
        END_IF;
```

```
        IF IMind2 OR IMind1 OR IMind0
        THEN
        IMFWDREV:=FALSE;(*giro arriba*)
        accion_envio1:=1;
        ok1:=1;
        END_IF;
        IF NOT IMind3 AND ok1=1
        THEN
        IMMARCHAPARO:=TRUE;
        ELSE
        IMMARCHAPARO:=FALSE;
        accion_envio1:=0;
        ok1:=0;
        END_IF;

        IF PMind4
        THEN
        accion_envio2:=2;
        ok2:=2;
        END_IF;
        IF PMind2 OR PMind1 OR PMind0
        THEN
        accion_envio2:=1;
        ok2:=1;
        END_IF;
        IF NOT PMind3 AND ok2=1
        THEN
        PMFWD:=TRUE;(*giro arriba*)
        PMREV:=FALSE;(*giro abajo*)
        ELSE
                IF NOT PMind3 AND ok2=2
                THEN
           PMFWD:=FALSE;(*giro arriba*)
           PMREV:=TRUE;(*giro abajo*)
                ELSE
           PMFWD:=FALSE;(*giro arriba*)
           PMREV:=FALSE;(*giro abajo*)
           accion_envio2:=0;
           ok2:=0;
                END_IF;
        END_IF;
3:
        IF IMind4 OR IMind3
        THEN
        IMFWDREV:=TRUE;(*giro abajo*)
        accion_envio1:=2;
        ok1:=1;
        END_IF;
        IF IMind1 OR IMind0
        THEN
```

```
        IMFWDREV:=FALSE;(*giro arriba*)
        accion_envio1:=1;
        ok1:=1;
        END_IF;
        IF NOT IMind2 AND ok1=1
        THEN
        IMMARCHAPARO:=TRUE;
        ELSE
        IMMARCHAPARO:=FALSE;
        accion_envio1:=0;
        ok1:=0;
        END_IF;

        IF PMind4 OR PMind3
        THEN
        accion_envio2:=2;
        ok2:=2;
        END_IF;
        IF PMind1 OR PMind0
        THEN
        accion_envio2:=1;
        ok2:=1;
        END_IF;
        IF NOT PMind2 AND ok2=1
        THEN
        PMFWD:=TRUE;(*giro arriba*)
        PMREV:=FALSE;(*giro abajo*)
        ELSE
                IF NOT PMind2 AND ok2=2
                THEN
           PMFWD:=FALSE;(*giro arriba*)
           PMREV:=TRUE;(*giro abajo*)
                ELSE
           PMFWD:=FALSE;(*giro arriba*)
           PMREV:=FALSE;(*giro abajo*)
           accion_envio2:=0;
           ok2:=0;
                END_IF;
        END_IF;
2:

        IF IMind4 OR IMind3 OR IMind2
        THEN
        IMFWDREV:=TRUE;(*giro abajo*)
        accion_envio1:=2;
        ok1:=1;
        END_IF;
        IF IMind0
        THEN
        IMFWDREV:=FALSE;(*giro arriba*)
```

```
accion_envio1:=1;
ok1:=1;
END_IF;
IF NOT IMind1 AND ok1=1
THEN
IMMARCHAPARO:=TRUE;
ELSE
IMMARCHAPARO:=FALSE;
accion_envio1:=0;
ok1:=0;
END_IF;

IF PMind4 OR PMind3 OR PMind2
THEN
accion_envio2:=2;
ok2:=2;
END_IF;
IF PMind0
THEN
accion_envio2:=1;
ok2:=1;
END_IF;
IF NOT PMind1 AND ok2=1
THEN
PMFWD:=TRUE;(*giro arriba*)
PMREV:=FALSE;(*giro abajo*)
ELSE
        IF NOT PMind1 AND ok2=2
        THEN
   PMFWD:=FALSE;(*giro arriba*)
   PMREV:=TRUE;(*giro abajo*)
        ELSE
   PMFWD:=FALSE;(*giro arriba*)
   PMREV:=FALSE;(*giro abajo*)
   accion_envio2:=0;
   ok2:=0;
        END_IF;
END_IF;
1:
IF IMind4 OR IMind3 OR IMind2 OR IMind1
THEN
IMFWDREV:=TRUE;(*giro abajo*)
accion_envio1:=2;
ok1:=1;
END_IF;
IF NOT IMind0 AND ok1=1
THEN
IMMARCHAPARO:=TRUE;
ELSE
IMMARCHAPARO:=FALSE;
```

```
        accion_envio1:=0;
        ok1:=0;
        END_IF;
        IF PMind4 OR PMind3 OR PMind2 OR PMind1
        THEN
        accion_envio2:=2;
        ok2:=2;
        END_IF;
        IF NOT PMind0 AND ok2=1
        THEN
        PMFWD:=TRUE;(*giro arriba*)
        PMREV:=FALSE;(*giro abajo*)
        ELSE
                IF NOT PMind0 AND ok2=2
                THEN
           PMFWD:=FALSE;(*giro arriba*)
           PMREV:=TRUE;(*giro abajo*)
                ELSE
           PMFWD:=FALSE;(*giro arriba*)
           PMREV:=FALSE;(*giro abajo*)
           accion_envio2:=0;
           ok2:=0;
                END_IF;
        END_IF;
END_CASE;
END_IF;
IF OFF
THEN
INICIAR:=2;
END_IF;
IF INICIAR=2
THEN
velIMasc:=0;
velPMasc:=0;
IMX1:=FALSE;
IMX2:=FALSE;
PMX1:=FALSE;
PMX2:=FALSE;
PMFWD:=FALSE;
PMREV:=FALSE;
END_IF;
(*------------------FINALES DE CARRERA----------------------*)
IF IMfcup OR IMfcdw
THEN
alarma_envio1:=1;
alarma:='FALLO SENSOR FINAL DE CARRERA INDUCCION';
velIMasc:=0;
iniciar:=2;
accion_envio1:=0;
IMMARCHAPARO:=FALSE;(*apaga induccion*)
```

```
ELSE
alarma_envio1:=0;
END_IF;
IF PMfcup OR PMfcdw
THEN
alarma:='FALLO SENSOR FINAL DE CARRERA SERVOMOTOR';
alarma_envio2:=1;
velPMasc:=0;
iniciar:=2;
accion_envio1:=0;
PMFWD:=FALSE;(*giro arriba*)
PMREV:=FALSE;(*giro abajo*)
ELSE
alarma_envio2:=0;
END_IF;
(*----------------FIN FINALES DE CARRERA----------------------*)
```