UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

Universitat Politécnica de València (Campus d'Alcoi)

Trabajo Fin de Grado

Diseño para impresión en 3D, control y montaje de un robot SCARA para manipulación

Autor/a: Pedro Rodríguez Rubio
Grado en Ingeniería Eléctrica
Tutor/a: Jaime Masiá Vañó
Curso de elaboración y defensa: 2016/2017

ÍNDICE

Resumen

Gracias a las nuevas tecnologías como la impresión en 3D y herramientas como *Arduino* se generan nuevas maneras de producir prototipos al alcance de cualquier usuario. En este caso, un robot SCARA completamente funcional.

para hacer esto posible se necesitan tres softwares distintos, *SolidWorks* para el diseño de las piezas que lo conforman, *KiCad* para su electrónica y finalmente *LabVIEW* para su programación.

Abstract

Thanks to new technologies such as 3D printing and tools like *Arduino*, new ways of producing prototypes are generated within the reach of any user. In this case, a fully functional SCARA robot.

To do this, you need three different software, *SolidWorks* for the design of the parts that make it up, *KiCad* for your electronics and finally *LabVIEW* for programming.

Palabras clave: SCARA, impresión 3D, servomotor, ABS, Arduino.

1. Justificación y/o Introducción

Con la irrupción de nuevas tecnologías como la impresión 3D y el, cada vez más común, uso de herramientas como *Arduino,* se quiso comprobar si sería posible la realización de un robot **SCARA** completamente funcional.

El desarrollo de este proyecto surge para su posterior utilización en el grupo de robótica (GROMEP) tanto en exhibiciones y demostraciones como en el propio trabajo del grupo controlando servomotores. Tiene una interface de control muy intuitiva, por lo que puede ser controlado por un individuo con unas nociones mínimas de la materia. Se ha creado un programa de demostración que coloca fichas de dominó.

Los apartados de los que consta este proyecto son el diseño de las piezas, la electrónica y la programación del brazo robótico.

Para todo ello se ha tenido que realizar toda la estructura del robot en *SolidWorks,* ya que nos permite generar archivos .stl, que posteriormente convertiremos a .gcode para su impresión.

Para este punto ha sido necesaria la realización de varios prototipos hasta acercarnos al definitivo, probando con distintas impresoras 3D, y comprobando experimentalmente la diferencia de calidad entre unas y otras.

En cuanto a la electrónica *(Arduino),* se sabía de antemano que para poder trabajar con varios servomotores, uno por cada grado de libertad (5), no sería posible sin el apoyo de un sistema adicional de potencia. Esto es necesario ya que el regulador de tensión (NCP1117), de la herramienta *Arduino*, tiene una corriente .de salida $I_o = 800mA$, insuficiente para los servomotores. Debido a esto, se considera necesario realizar una placa electrónica, con el software *KiCad,* con un regulador de tensión adecuado a nuestras necesidades.

Para el control de los servomotores se realiza un programa con el software *LabVIEW* basado en guardar posiciones concretas en memoria para su posterior utilización.

El primer escollo del proyecto es la incapacidad de controlar cinco servomotores con *Arduino Uno* debido a que su microcontrolador *Atmega328p* solo tiene tres salidas PWM, por lo que se tuvo que continuar el trabajo con *Arduino Mega 2560.*

Tras solucionar dicho problema nos encontramos con el consumo de los cinco servomotores, y es que, si bien un regulador 7805 5V 1A, alimentado con una fuente de

alimentación externa de 7V y 2A, no tiene problema para controlar un servomotor por separado, es incapaz de poder suministrar la corriente necesaria para el control de todos los servomotores simultáneamente. Por ello se tuvo que plantear de nuevo tanto la manera de alimentar como el tipo de regulador.

El problema se soluciona utilizando un regulador de tensión, también de 5V, pero con una corriente máxima de salida de 2.2 amperios y alimentado con una batería de 7,4 voltios capaz de suministrar hasta 24 amperios.

A continuación, se encuentra el desarrollo de todo el proceso paso a paso de la realización completa de un robot SCARA de cinco grados de libertad.

2. Proceso o método

2.1. Robot SCARA

Un *SCARA* consiste en un robot articulado, de cuatro grados de libertad, con movimientos semejantes al brazo humano (hombro, codo, muñeca). De ahí que sea muy utilizado en sistemas de ensamblaje.

Son robots de un amplio movimiento en sus ejes X e Y, aunque bien es cierto que están más limitados en el eje Z. Para conseguir movimiento en dicho eje diseñamos un mecanismo de engranajes "piñón-cremallera".

La principal idea de este proyecto es la realización de un *SCARA* con el mínimo coste posible, y aprovechando la gran irrupción de las impresoras 3D en el mundo de la producción consideramos la oportunidad de poder diseñar e imprimir todas las partes posibles del robot en nuestra impresora 3D Dimensión elite.

2.1.1. Diseño de la estructura

Existen varios diseños válidos en la realización de un robot *SCARA*, y puesto que buscamos la máxima funcionalidad posible, nos decantamos por un diseño con "hombro", "brazo", "antebrazo" y una "muñeca" dónde instalamos un mecanismo con piñón y cremallera para poder movernos por el eje Z, ya que sin dicho mecanismo únicamente podríamos desplazarnos por el plano (X e Y). Dicho mecanismo, en el que profundizaremos más adelante, lleva una pinza acoplada a dos servomotores en el extremo, que hace las funciones de "mano" de nuestro sistema.

Nuestro diseño irá acoplado a una estructura metálica, que a su vez dispondrá de una superficie de metacrilato para poder probar nuestro diseño.

A continuación, se mostrará una serie de imágenes con sus correspondientes acotaciones pieza a pieza de nuestro diseño.

Pero antes, decir que la totalidad de las piezas se han impreso con *ABS (Acrilonitrilo butadieno estireno),* ya que es un material muy resistente al impacto (golpes) muy utilizado en automoción y otros usos tanto industriales como domésticos, por lo que soporta a la perfección cualquier tipo de movimiento, por brusco que sea, de nuestro robot SCARA. Es un termopástico amorfo.

Se le llama plástico de ingeniería, debido a que es un plástico cuya elaboración y procesamiento es más complejo que los plásticos comunes, como son las polioleofinas (polietileno, polipropileno).

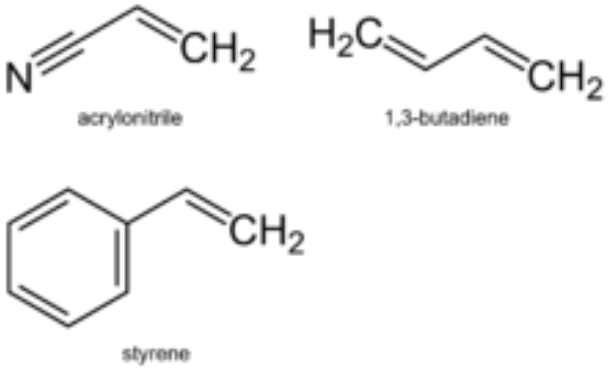


Ilustración 1: componentes del ABS

Los bloques de acrilonitrilo proporcionan rigidez, resistencia a ataques químicos y estabilidad a alta temperatura, así como dureza, propiedades muy apreciadas en ciertas aplicaciones como son equipos pesados o aparatos electrónicos.

Los bloques de butadieno, que es un elastómero, proporcionan tenacidad a cualquier temperatura. Esto es especialmente interesante para ambientes fríos, en los cuales otros plásticos se vuelven quebradizos. El bloque de estireno aporta resistencia mecánica y rigidez.

Esta mezcla de propiedades, llamada sinergia, indica que el producto final contiene mejores propiedades que la suma de ellos. El ABS es un ejemplo claro del diseño de

materiales en ingeniería química, que busca lograr compuestos de materiales ya existentes en oposición a desarrollar materiales completamente nuevos.

El rasgo más importante del ABS es su gran tenacidad, incluso a baja temperatura (sigue siendo tenaz a -40°C). Además, es duro y rígido, tiene una resistencia química aceptable, baja absorción de agua, y por lo tanto buena estabilidad dimensional, alta resistencia a la abrasión, y puede recubrirse con una capa metálica con facilidad.

El ABS se puede, en una de sus variantes, cromar por electrólisis dándole distintos baños de metal a los cuales es receptivo.

Por todo esto, la elección de ABS para la realización del proyecto resulta la más idónea.

A continuación, tabla de propiedades físico-mecánicas.

Alargamiento en la rotura (%)	45
Coeficiente de fricción	0,5
Módulo de tracción (GPa)	2,1-2,4
Resistencia a la tracción (MPa)	41-45
Resistencia al impacto Izod (J/m^{-1})	200-400
Absorción de agua en 24 horas (%)	0.3-0.7
Densidad (g/cm^3)	1,05
Resistencia a la radiación	Aceptable
Resistencia a los ultravioletas	Baja

3. Análisis, desarrollo y Presentación de resultados

3.1. Prototipo inicial

El primer prototipo realizado para el proyecto fue impreso con una impresora 3D *BCN3D+*. Esta es un tipo de impresora de calidad media, por lo que el diseño de las piezas siempre va condicionado a ella, es decir, no se pueden dejar grandes huecos sin una base donde apoyar ya que hay que evitar siempre la utilización de material se soporte. Esto se debe a que como los soportes son con el mismo material que el resto de impresión se corre el riesgo de dañar la pieza a la hora de retirarlos.

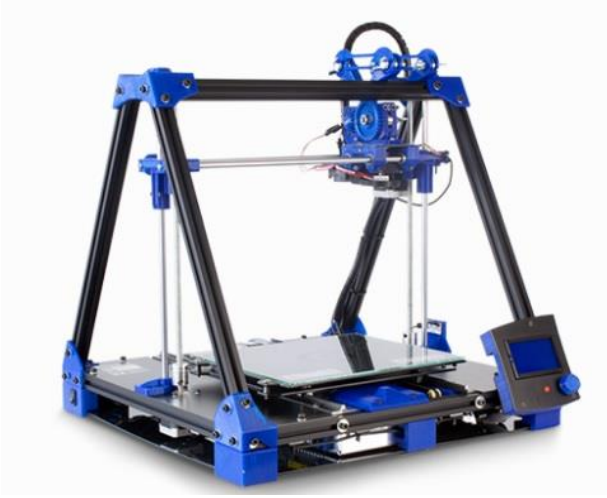


Ilustración2: Impresora 3D BCN3D+

Esto nos llevó a tener que diseñar tanto el Brazo como el Antebrazo con una unión metálica en el centro, como se muestra en la siguiente imagen.

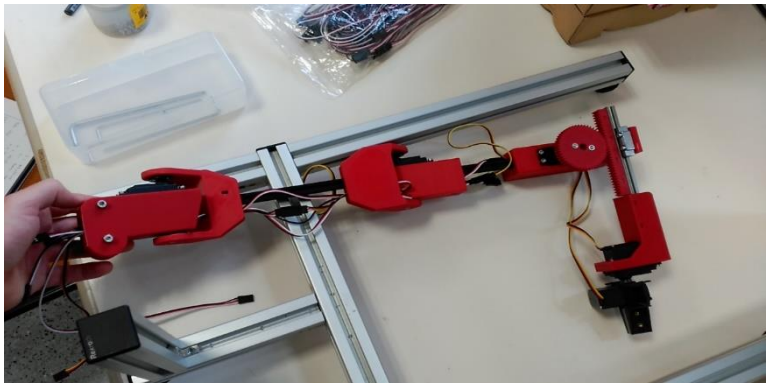


Ilustración 3: primer prototipo ensamblado

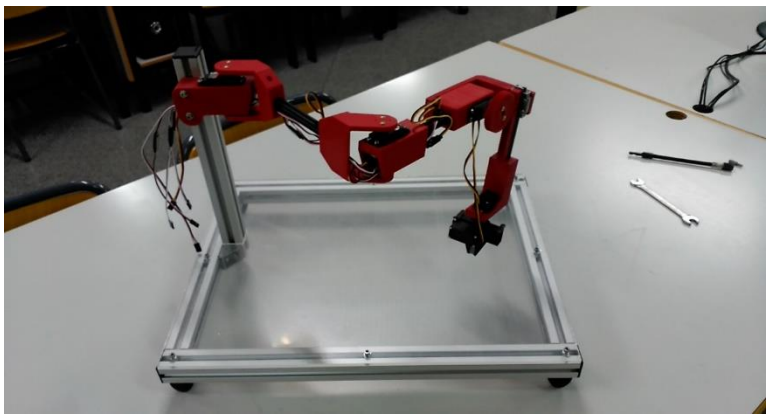


Ilustración 4: primer prototipo en la estructura de apoyo

Al terminar el montaje de dicho prototipo se detecta que sufre una importante flexión en la pieza del hombro, por lo que se debe mejorar el diseño del SCARA, haciéndolo más robusto, por lo que se decide diseñar una nueva pieza Brazo completamente en ABS, pero se diseña como un ensamblaje de dos piezas, teniendo en cuenta que sigue siendo en la impresora 3d BCN3D+, por lo que no se puede realizar un conducto interno para el cableado demasiado grande.

Ilustración 5: Prototipo con Brazo nuevo, seguía presentando flexión por antebrazo

Ilustración 6: Nuevo antebrazo realizado en SolidWorks en un ensamblaje de dos piezas

Como se puede ver en la anterior imagen el robot SCARA sigue presentando flexión, ahora en la parte del antebrazo, por lo que ahora sí, se decide realizar Brazo y Antebrazo en una única pieza para impresora 3D *Dimension Elite*, con lo que se mejoraría, no solo la calidad de la pieza impresa, sino también la flexión.

3.2. Prototipo definitivo

Cabe mencionar, que, aunque comenzamos con la intención de trabajar con Arduino Uno, por motivos que más adelante se explicarán, tuvimos que continuar nuestro trabajo con Arduino Mega2560.

En el primer prototipo tuvimos que adaptar nuestro diseño a la impresora con que íbamos a realizar la impresión, *BCN 3D+*, este diseño implicaba no poder imprimir de una sola pieza aquellas piezas de gran tamaño o geometría difícil para dicha impresora (Brazo y Antebrazo)

¿Qué queremos decir con "geometría difícil"? Para que la pieza pueda ser correctamente impresa, no debe tener ninguna línea al aire, es decir, toda línea de impresión que no tenga un base sobre la que apoyar deberá contener un soporte que la propia impresora coloca.

El problema es que en este tipo de impresora el material de soporte el es mismo que el material utilizado para la pieza, lo que implica una alta probabilidad de estropear la pieza a la hora de retirar dichos soportes.

Es por esto que en el primer prototipo utilizamos barras metálicas en lugar de una pieza 100% impresas. Pero tras el ensamblaje pudimos comprobar que el brazo sufría una alta flexión, por lo que se diseñó una pieza en dos partes para sustituirla en el Brazo. Compramos así, que se mejoraba cuantiosamente el problema de la flexión, por lo que se decide in decidimos ir un paso más allá y realizar todas las piezas en la nueva impresora adquirida 10,06por la EPSA, la *Dimension Elite.*

Ilustración 7: Impresora 3D Dimensión Elite del laboratorio I3L9

La ventaja que aporta esta impresora es que puedes imprimir cualquier pieza sin tener que preocuparte por el tipo de geometría, a diferencia de la BCN 3D+, que debes diseñar condicionado por la impresora.

Esto se debe al sistema de eliminación de soporte SCA-1200, es un sistema basado en poder disolver el material de soporte sin necesidad de tener que eliminarlos mecánicamente, por lo que la pieza no debe sufrir daño alguno.

El soporte soluble le permite fabricar modelos intrincados con salientes e incluso conjuntos móviles, anidados, en un solo trabajo de impresión. El SCA-1200 es un sistema de eliminación de soporte que permite disolver el soporte en una solución acuosa, sin utilizar las manos y de forma desatendida. Basta con colocar el prototipo impreso en 3D en el sistema y el SCA ofrece la temperatura y agitación adecuada para revelar de forma eficiente la pieza o conjunto listo para utilizar.

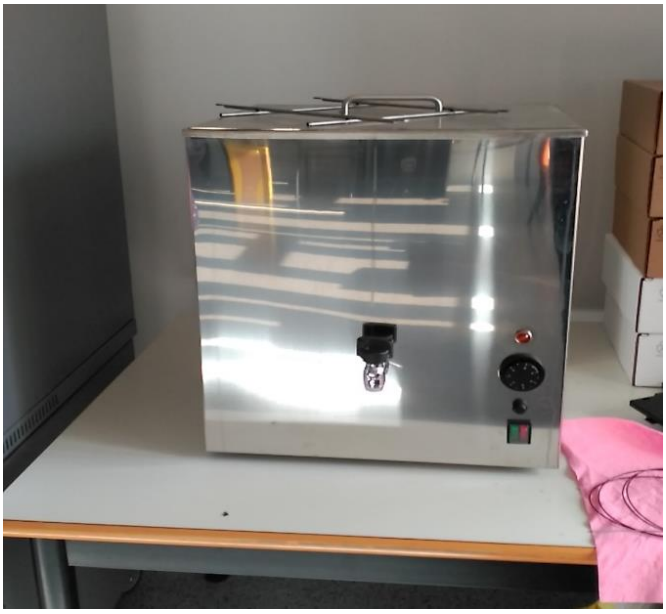


Ilustración 8: Sistema de eliminación de soporte SCA-1200

3.2.1. Hombro

Es la pieza de unión entre nuestro sistema y la estructura metálica, es una pieza diseñada en su totalidad mediante el software CAD *SolidWorks*, como podemos ver en la imagen.

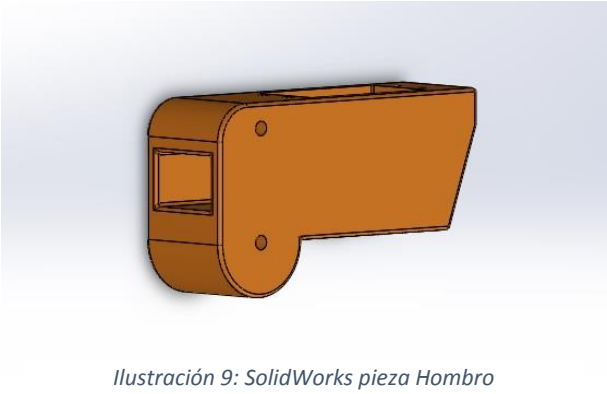


Ilustración 9: SolidWorks pieza Hombro

La pieza, como podemos observar en el corte transversal de la siguiente imagen, tiene un hueco interno para pasar todos los cables, ya que al ser la pieza más cercana a la placa electrónica todos los cables salen por ella.

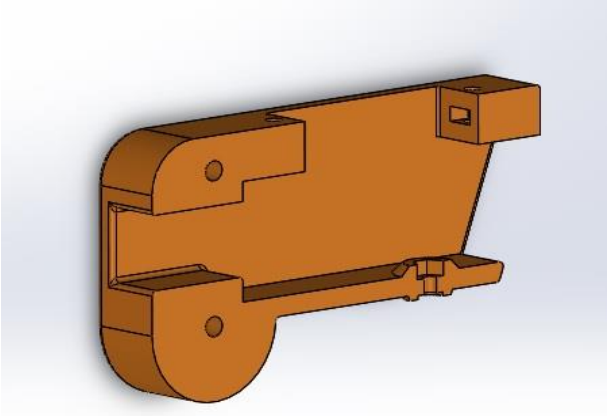


Ilustración 10: SolidWorks pieza Hombro, corte transversal

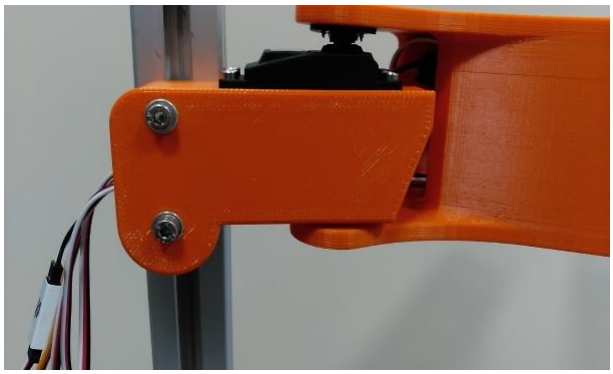


Ilustración 11: Pieza real montada

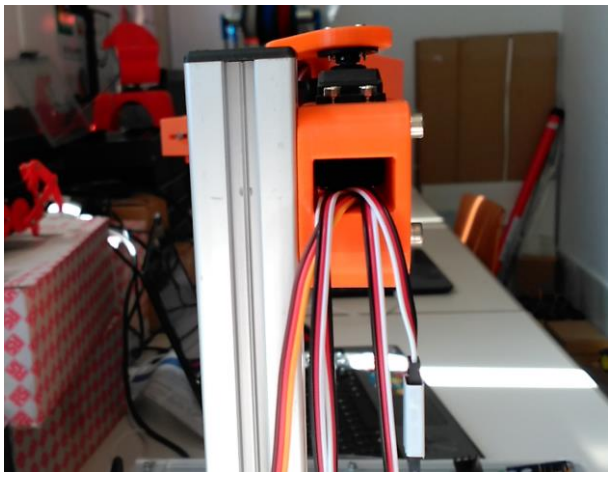


Ilustración 12: Parte trasera del hombro, por donde desemboca todo el cableado

Está unida a la estructura metálica mediante dos tornillos tipo *Allen* con cabeza cónica de métrica 4x50mm. Dichos tornillos han tenido que ser cortados alrededor de 15mm para su perfecta colocación. Ya que no encontramos métrica adecuada a nuestra necesidad. A continuación, imágenes de dichos elementos antes y después de haber sido modificados, junto a un pie de rey para asegurar su métrica.

Ilustración 13: Antes de su modificación

Ilustración 14: Después de su modificación

A dicha pieza se le acopla un servomotor del tipo *Futaba s3003* que servirá de articulación cuando se una con la pieza brazo.

COTAS HOMBRO

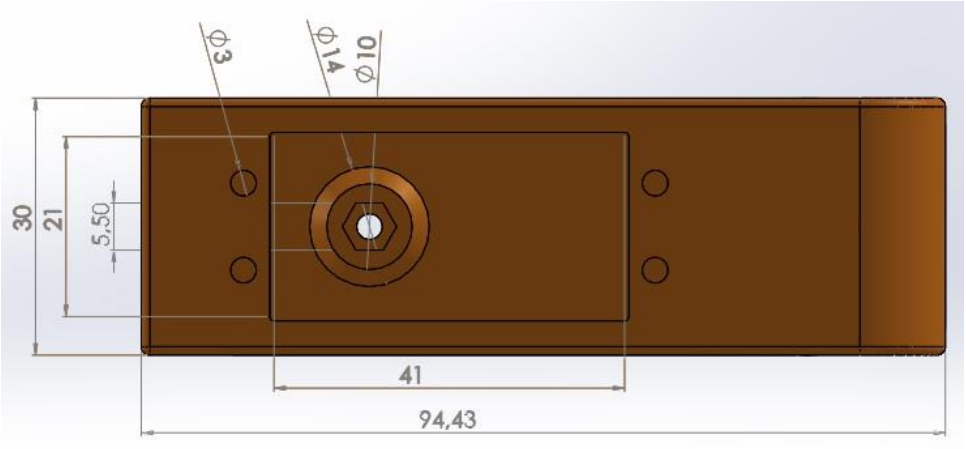


Ilustración 15: Hombro planta

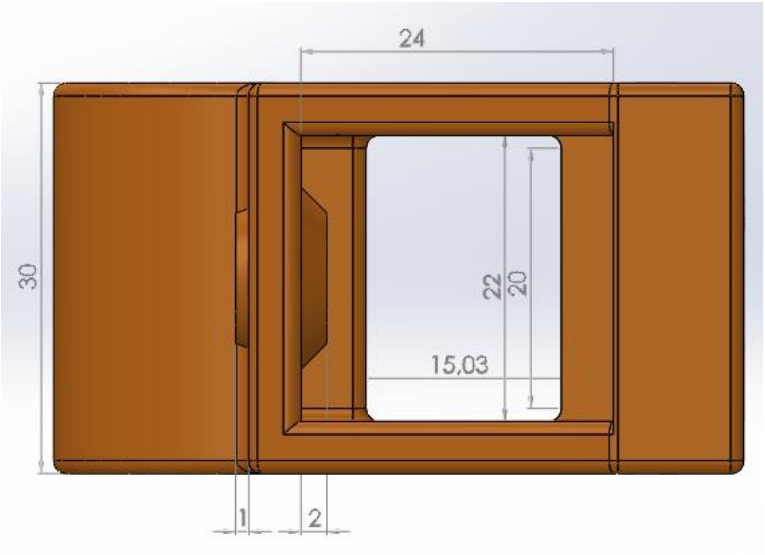


Ilustración 16: Hombro de perfil

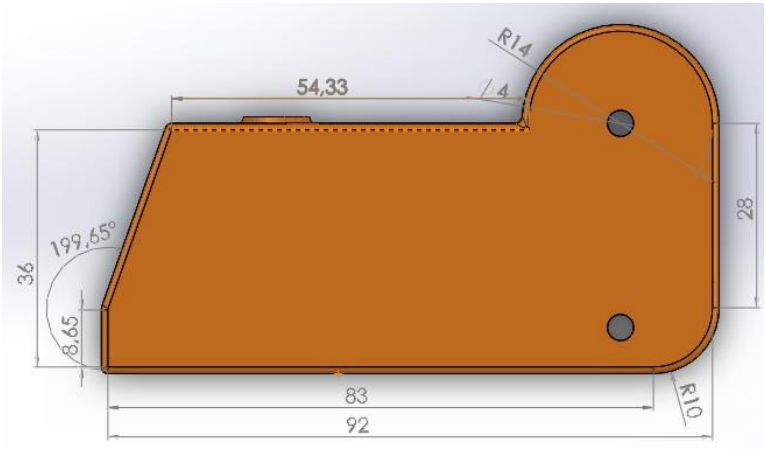


Ilustración 17: Hombro alzado

3.2.2. Brazo

En el prototipo definitivo ya no observamos problemas de flexión, por lo que damos por válido éste último diseño.

Las uniones con hombro y antebrazo son realizadas mediante los servomotores *Futaba s3003* y un mecanismo con rodamiento y arandela de presión, que permite la rotación de la pieza con el menor rozamiento posible

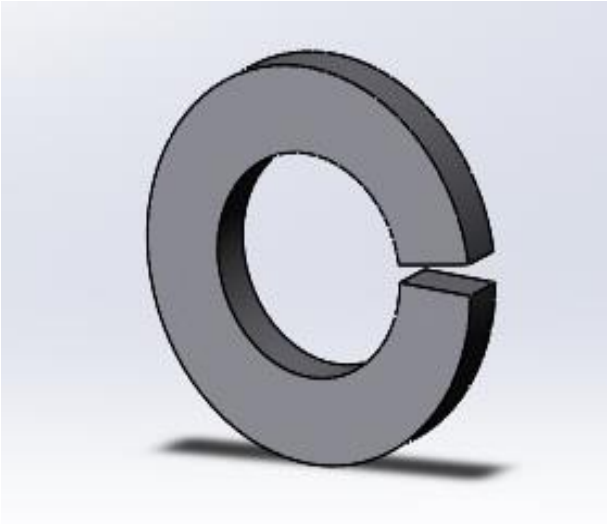


Ilustración 18: Arandela de presión SolidWorks

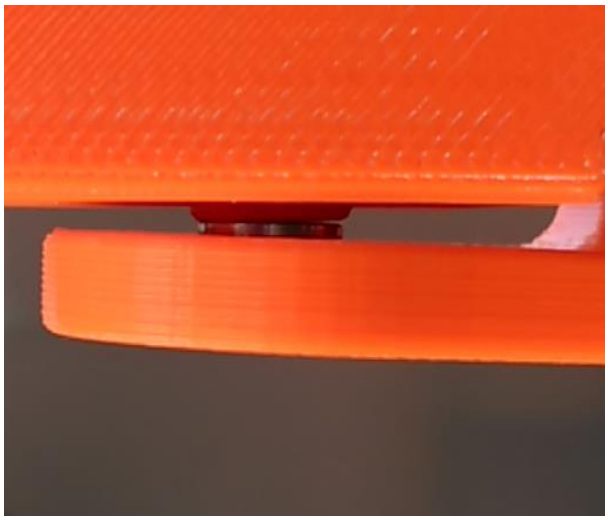


Ilustración 19: Montaje definitivo SolidWorks

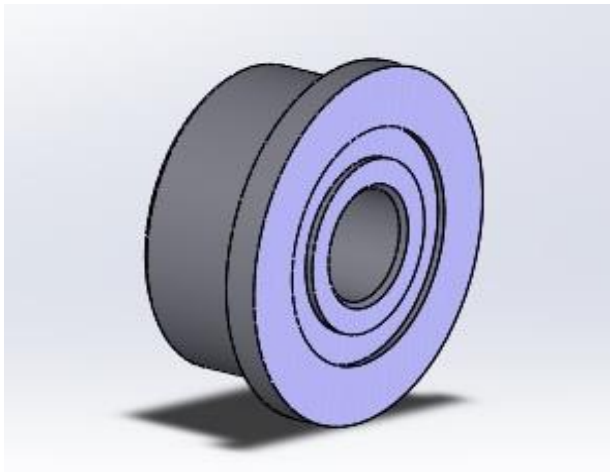


Ilustración 20: Rodamiento SolidWorks

La forma y rigidez de la pieza nos permite mover el prototipo sin que presente ningún tipo de flexión, el diseño presenta un vaciado interior muy considerable, pensando siempre en que tanto los cables, como sobre todo la unión entre estos deben quedar dentro de la pieza, ya que, si quedaran por fuera, en uno de los extremos, interferiría en el

movimiento del brazo, generando tensiones que acabaría por dañarlo. Además, esta impresora nos permite diseñar un buen conducto para alojar el cableado.

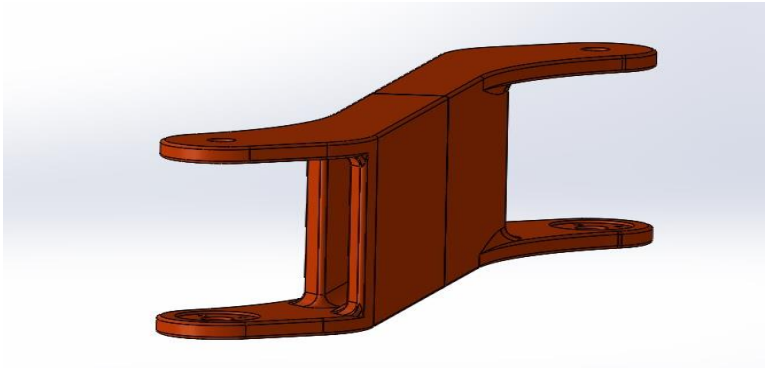


Ilustración 21: Brazo diseñado en SolidWorks

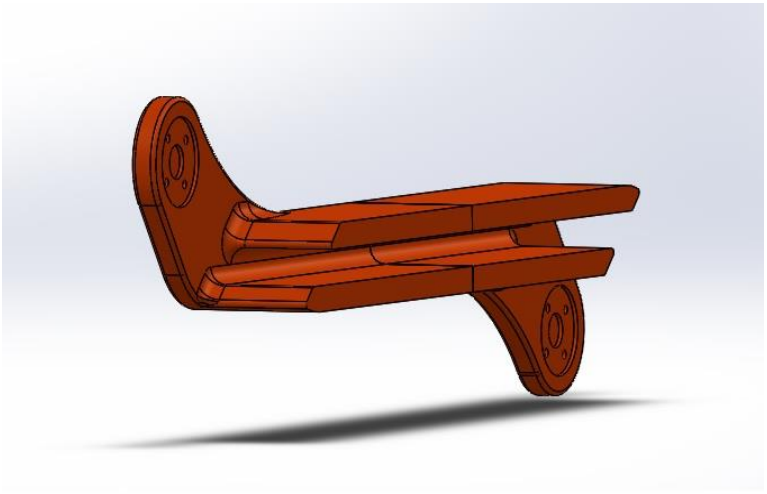


Ilustración 22: Corte transversal en SolidWorks para ver conducto para cableado

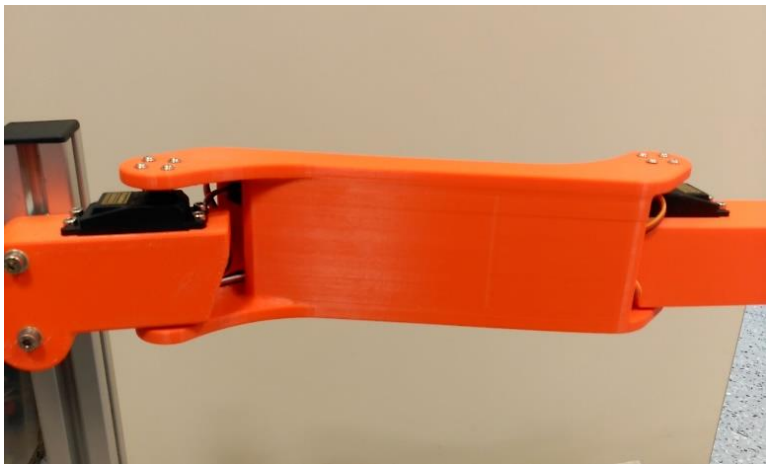


Ilustración23: Pieza perfectamente ensamblada en el prototipo

Ilustración 24: Proceso de montaje, hombro más brazo

COTAS BRAZO

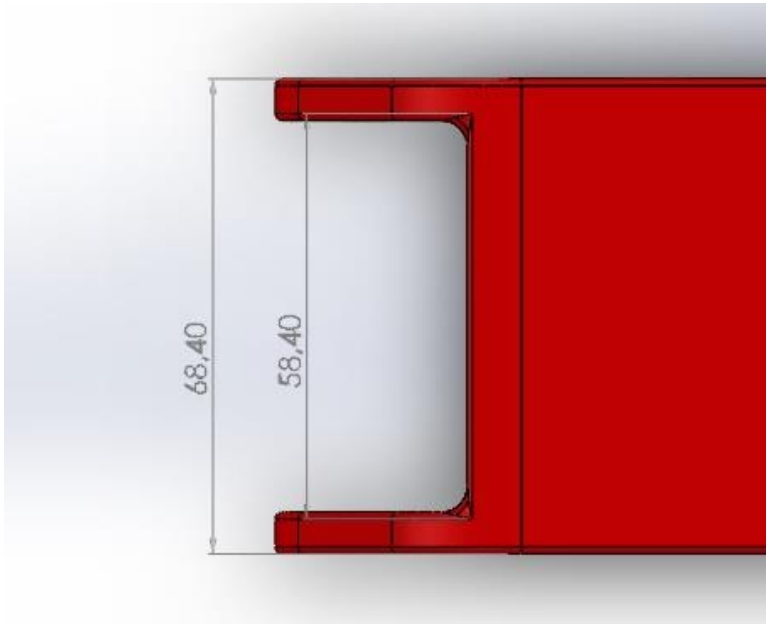


Ilustración 25: Brazo planta

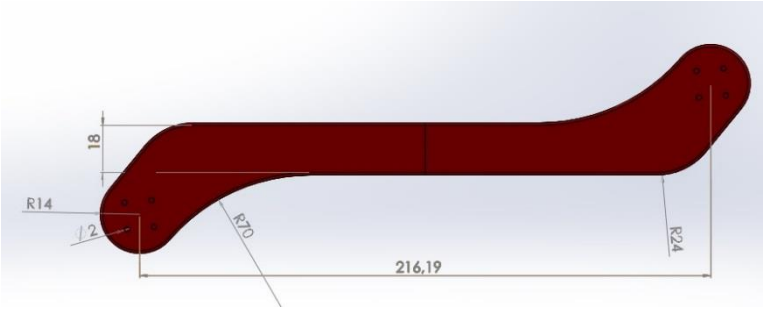


Ilustración 26: Brazo alzado

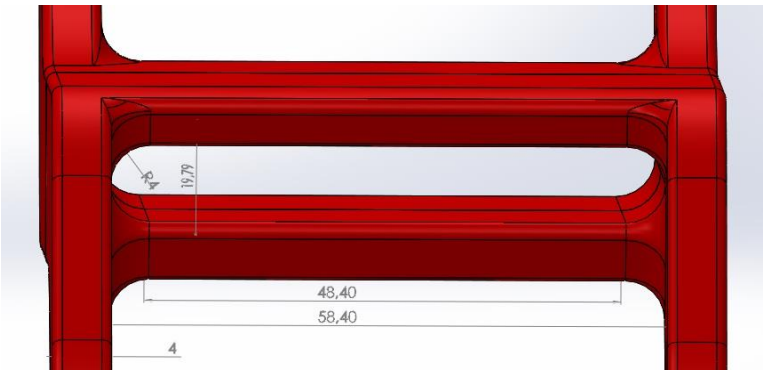


Ilustración 27: Brazo de perfil

3.2.3. Antebrazo

Al igual que la pieza *Brazo*, el antebrazo también ha podido ser realizado en su totalidad con ABS en la impresora 3D *Dimensión Elite*.

De esta forma, gracias al diseño de una pieza única para el antebrazo, hemos conseguido solucionar el problema de la flexión que teníamos.

Además, al igual que sucedió con la pieza Brazo, al imprimir con esta impresora se ha podido diseñar un amplio hueco en su interior para hacer pasar el cableado.

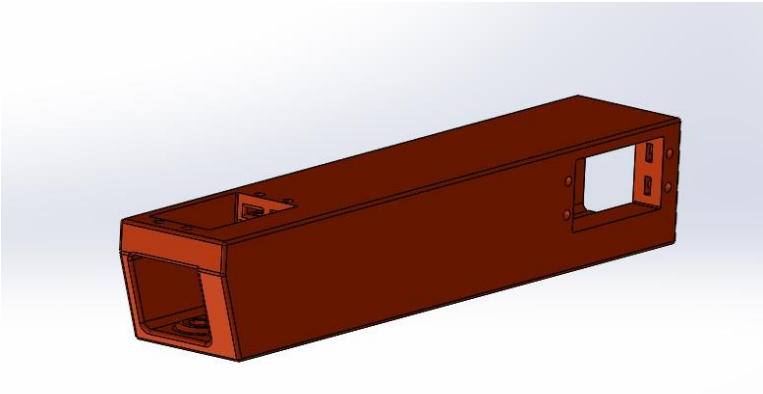


Ilustración 28: Antebrazo en SolidWorks

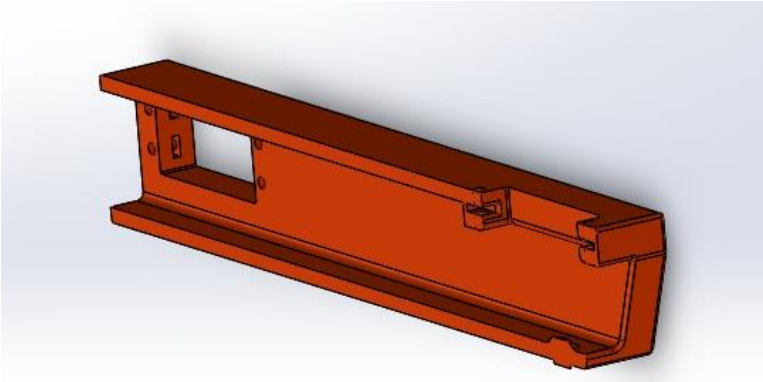


Ilustración 29: Corte transversal en SolidWorks

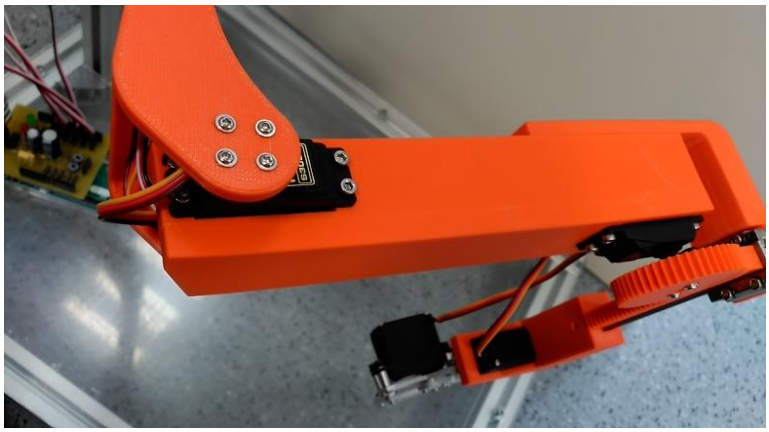


Ilustración 30: Antebrazo ensamblado con carril y engranajes

En el extremo de la pieza (“muñeca”) se encuentra el servomotor encargado de accionar el sistema piñón – cremallera, que es quien nos permite accionar el mecanismo encargado de movernos por el eje Z. De este hablaremos más adelante en detalle

También dispone, en su extremo, de dos agujeros para tornillería de métrica 3, para permitirnos ensamblar el carril corredizo, del que hablaremos en el siguiente punto.

COTAS ANTEBRAZO

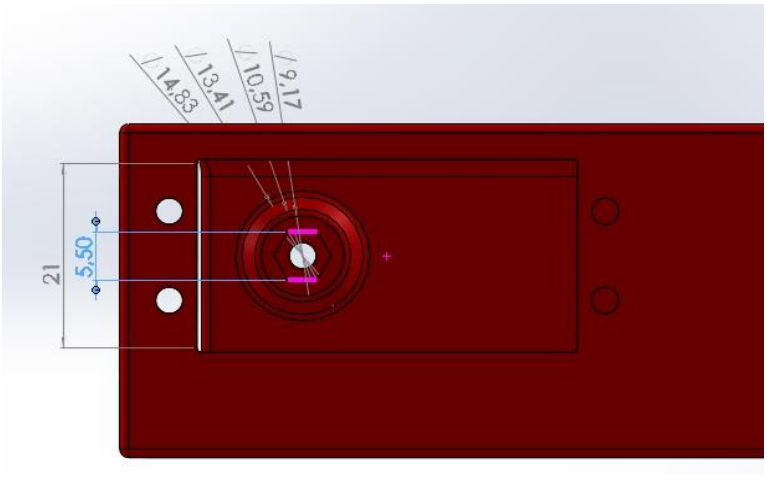


Ilustración 31: Antebrazo planta

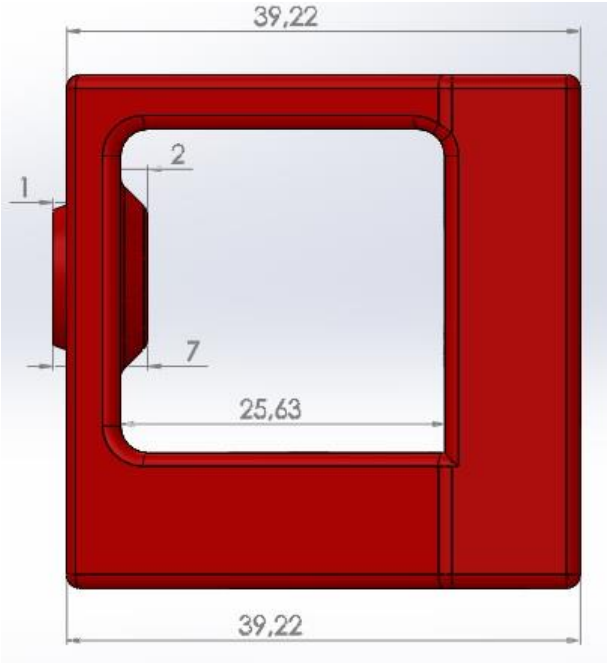


Ilustración 321: Antebrazo de perfil

Ilustración 33: Antebrazo alzado

3.2.4. Carril corredizo

Esta pieza nos permite modificar el diámetro del piñón en el sistema piñón – cremallera.

Por defecto se ha colocado en piñón de 50mm de diámetro, pero el juego del carril corredizo nos permite variar entre 40mm y 60mm. Tema del que se hablara más en profundidad en el apartado Piñón-Cremallera.

Es una pieza de *ABS* en su totalidad que se sujeta, mediante dos ojales, a dos tornillos anclados al Antebrazo, permitiendo que, aflojando los dos tornillos, la pieza se deslice para poder cambiar el tipo de piñón necesario.

A esta pieza irá atornillada la *Mesa deslizante, Igus, para anchura de Rail 13mm.*

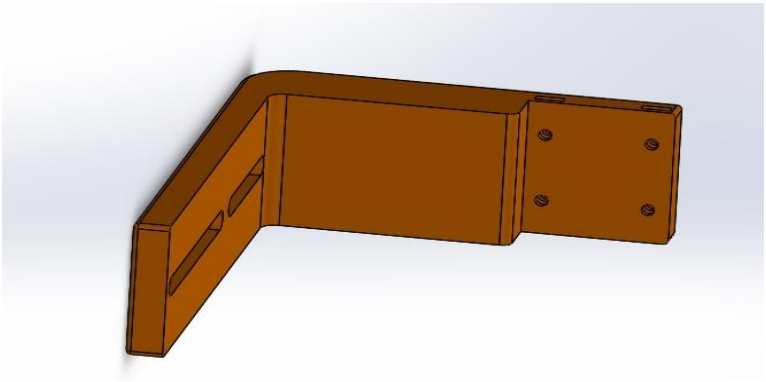


Ilustración 34: Carril corredizo en SolidWorks

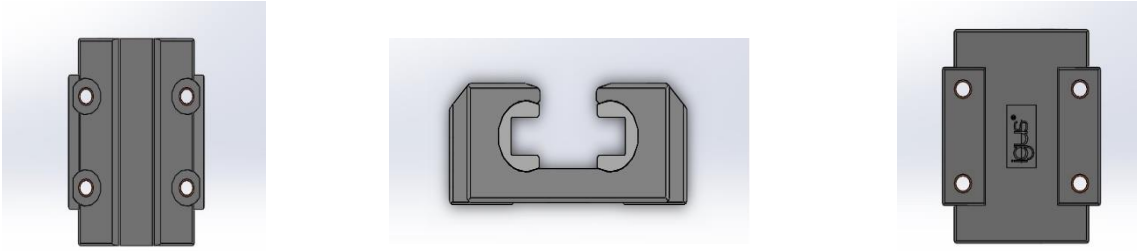


Ilustración 35: Mesa deslizante, Igus, para anchura de Rail 13mm

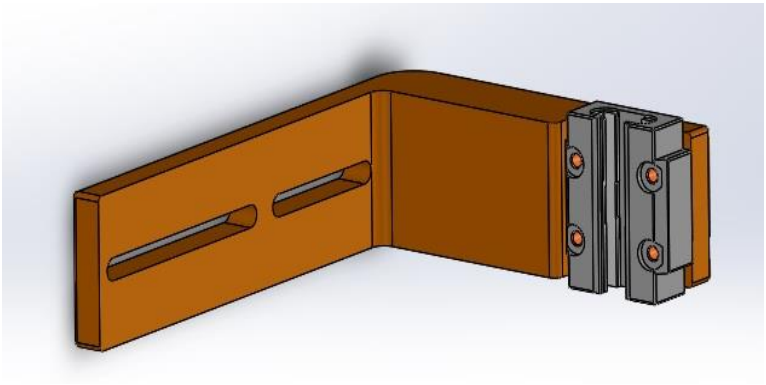


Ilustración 36: Carril corredizo en SolidWorks con mesa deslizante, Igus, para anchura de Rail 13mm

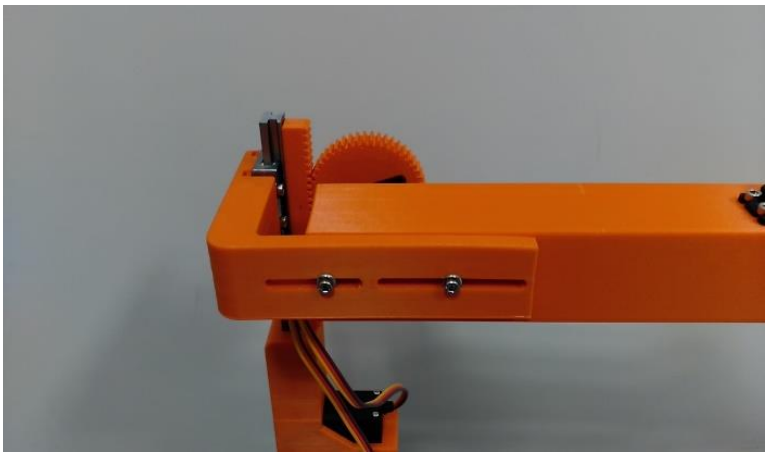


Ilustración 37: Carril deslizante ensamblado junto con sistema de engranajes

COTAS CARRIL CORREDIZO

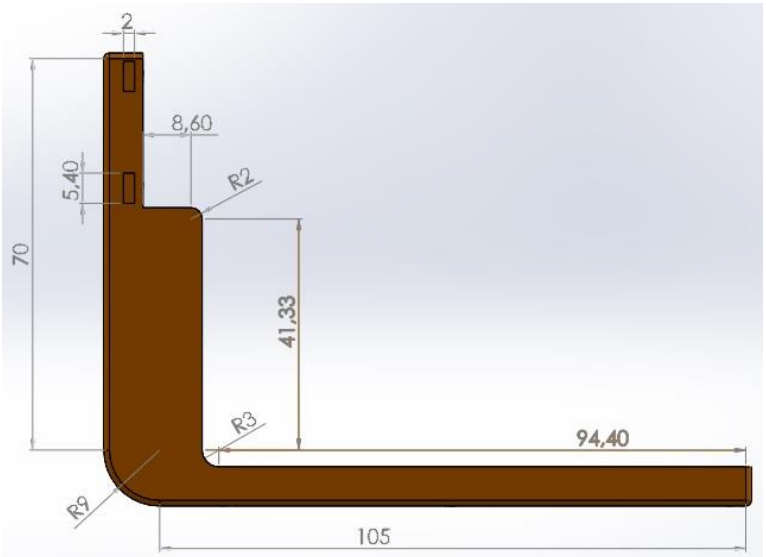


Ilustración 38: Carril planta

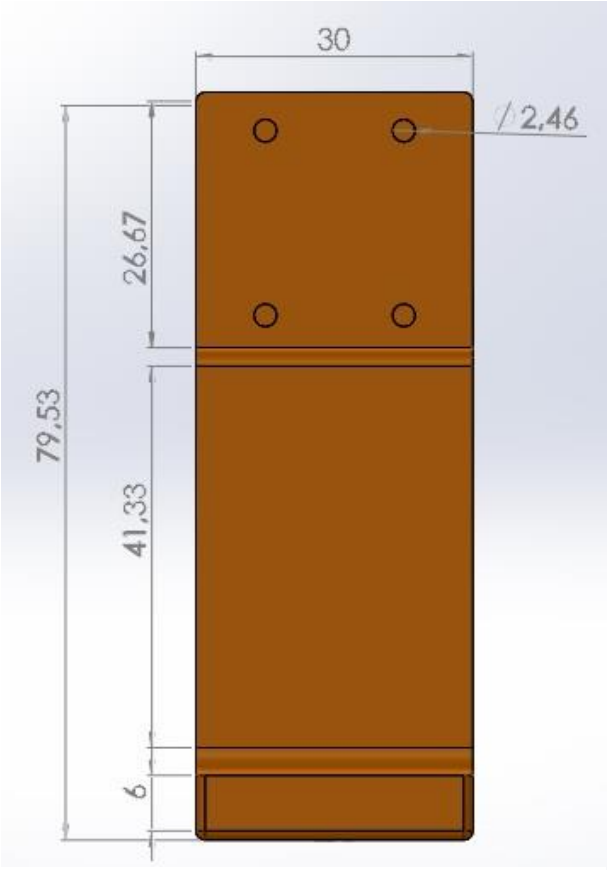


Ilustración 39: Carril de perfil

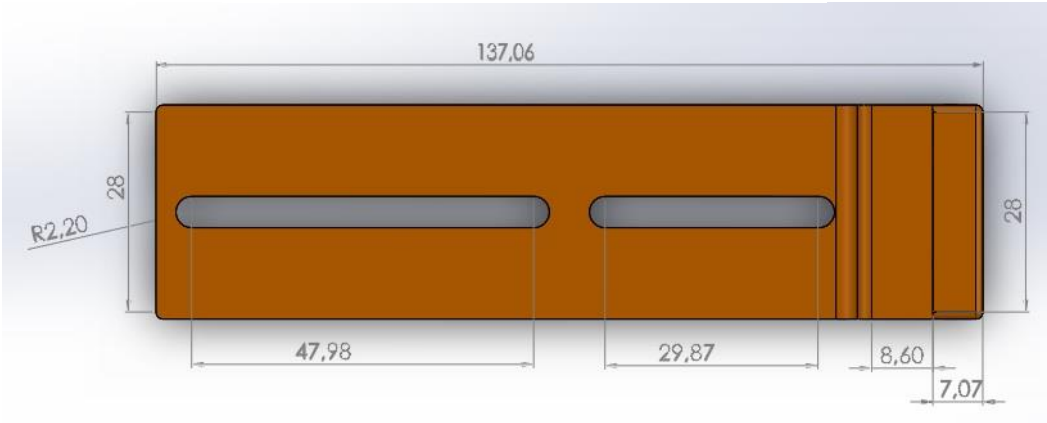


Ilustración 40: Carril alzado

3.2.5. Piñón-cremallera

Es el mecanismo que nos sirve para poder trabajar en el eje Z. Está formado por un piñón, el cual puede ser sustituido por distintos diámetros (40mm≤D≤60mm) según el par motor que necesitemos, y por una cremallera de 10 centímetros atornillada al carril guía de aluminio. A continuación, imágenes de SolidWorks de dicho elemento.

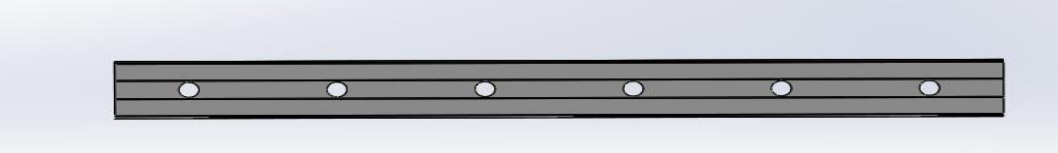


Ilustración 41: Parte delantera de carril guía

Ilustración 42: Parte trasera de carril guía

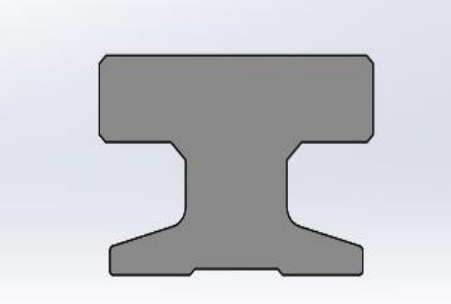
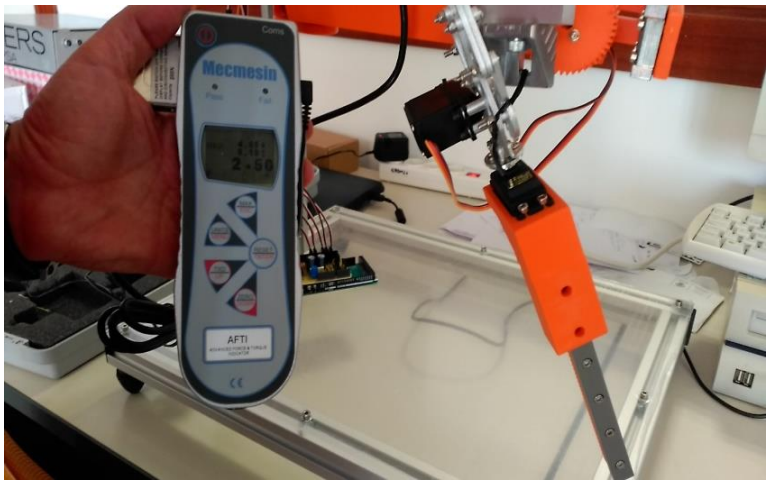


Ilustración 43: Fotografía tomada en el laboratorio I3L4 con dinamómetro

Mediante un dinamómetro pudimos analizar cuanta fuerza ejercían nuestras piezas sobre el servomotor encargado de hacer rotar el piñón de nuestro sistema piñón-cremallera, para saber si nuestro servomotor, de par 3,2 kg*cm (0,32 N*m) (datos del fabricante), iba a ser suficiente, y en caso de que así fuera, comprobar cuanta carga íbamos a poder levantar. En la siguiente imagen podremos observar los newtons de todo el sistema que nuestro piñón debe mover; cremallera, patín, soporte para servomotor de pinza, dos servomotores para pinza y pinza.



En el dinamómetro de la imagen se observa como el sistema ejerce una fuerza total de F=2.5N, por lo que sabiendo que T=F*R (Nm),

$$F = \frac{T}{R} = \frac{0{,}32\,Nm}{0{,}025\,m} = 12{,}5\,N$$

obtenemos la diferencia de newtons para saber el peso total que podremos levantar,

$$F\,(restante) = 12{,}8\,N - 2{,}5\,N = 10{,}3\,N$$

$$1\,Newton = 0{,}1020\,kilogramos\ fuerza$$

$$1{,}0503\,kilogramos$$

Para el cálculo del piñón, suponemos una carrera C=80mm y módulo m=1. Por tanto, sabiendo que

$$C = \Pi R\ ,\quad R = 25{,}4648\,mm\ ,\quad R \approx 25\,mm\ .$$

El diámetro de nuestro piñón será por tanto de 50 milímetros, pero como dijimos anteriormente, podremos sustituirlo por uno de 40 o por uno de 60 milímetros.

Tras calcular el radio deberemos proceder a calcular el número de dientes que deberá tener nuestro piñón.

$$z = \frac{2\mathrm{R}}{m} = \frac{2 * 25{,}4648}{1} \approx 50\,dientes$$

Para comprobar nuestro cálculo, hacemos las operaciones de manera inversa, por tanto,

$$C = \Pi R = \Pi 50 = 78{,}54\,mm$$

aproximadamente los 80 milímetros de carrera que suponíamos.

De esta misma forma hemos calculado las carreras que obtendríamos para 40 y 60 milímetros, al igual que el peso que va a poder levantar nuestro robot SCARA. Los datos para los respectivos diámetros los podemos ver en la siguiente tabla, extraída de una hoja de cálculo.

D(mm)	C(mm)	z(dientes)	F(N)	F(restante)	P(kgf)
40,00	62,83	40,00	16,00	13,50	1,38
50,00	78,54	50,00	12,80	10,30	1,05
60,00	94,25	60,00	10,67	8,17	0,83

T(N*m)= 0,32
m= 1
Fsistema(N)= 2,5

Ilustración 44: Tabla de resultados para todos los diámetros

COTAS DE PIÑÓN Y CREMALLERA

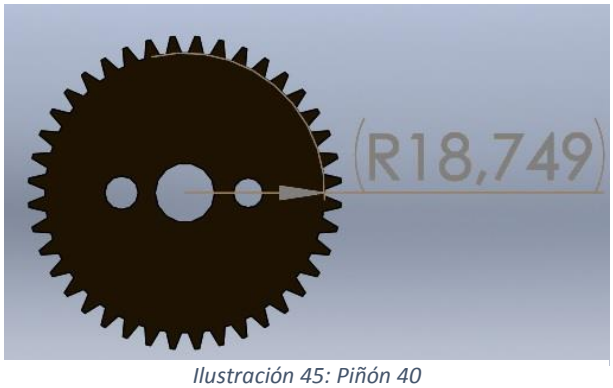


Ilustración 45: Piñón 40

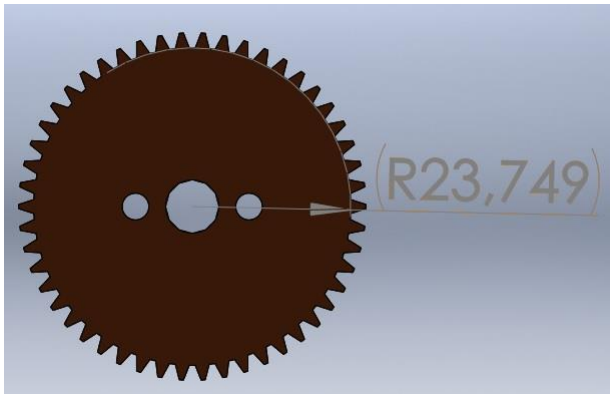


Ilustración 46: Piñón 50

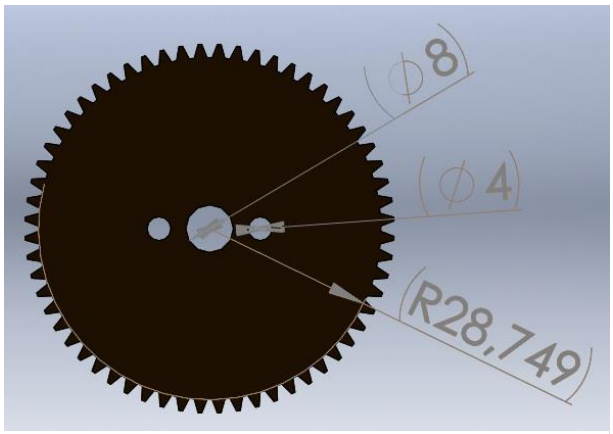


Ilustración 47: Piñón 60

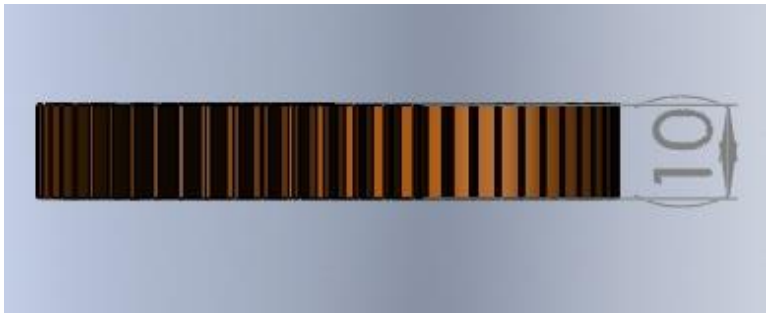


Ilustración 48: Ancho del piñón

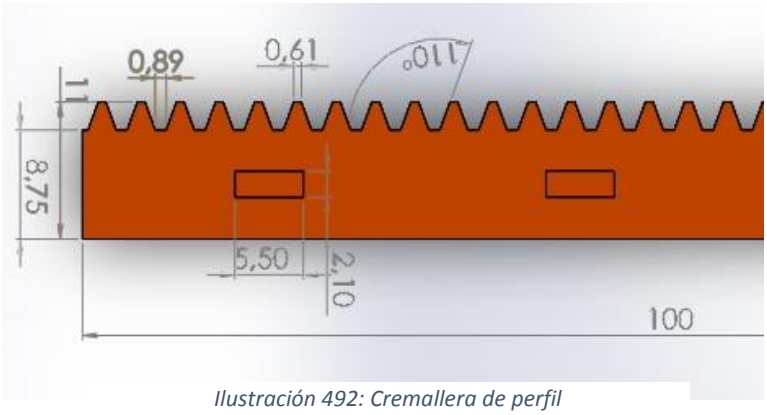


Ilustración 492: Cremallera de perfil

Ilustración 50: Cremallera alzada

3.2.6. Soporte de la Pinza

Esta pieza es el nexo de unión entre la cremallera y la pinza, ya que el carril guía sobre el que va atornillada la cremallera va insertado y sujetado con dos tornillos métrica 3 a esta pieza.

Dicha pieza lleva en su extremo el hueco necesario para el servomotor que moverá, de forma rotatoria, la pinza de nuestro robot SCARA.

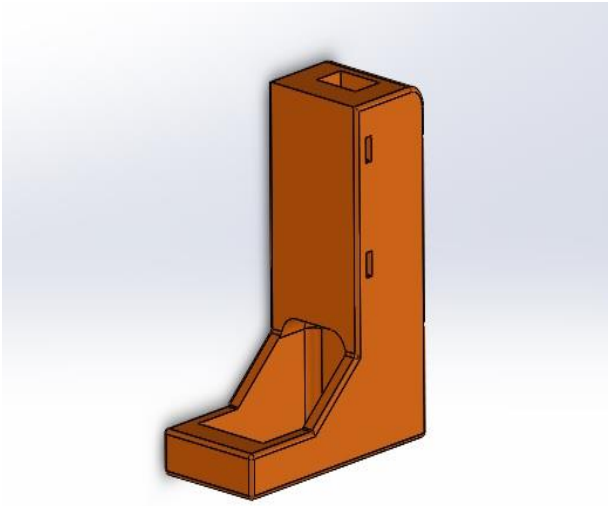


Ilustración 51: Soporte para pinza SolidWorks

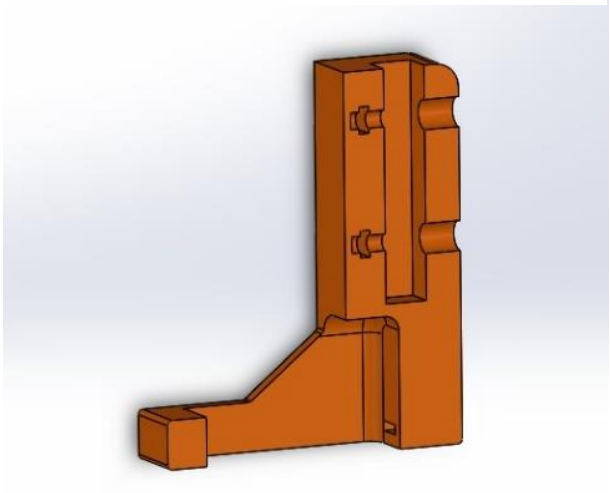


Ilustración 52: Corte transversal en soporte pinza

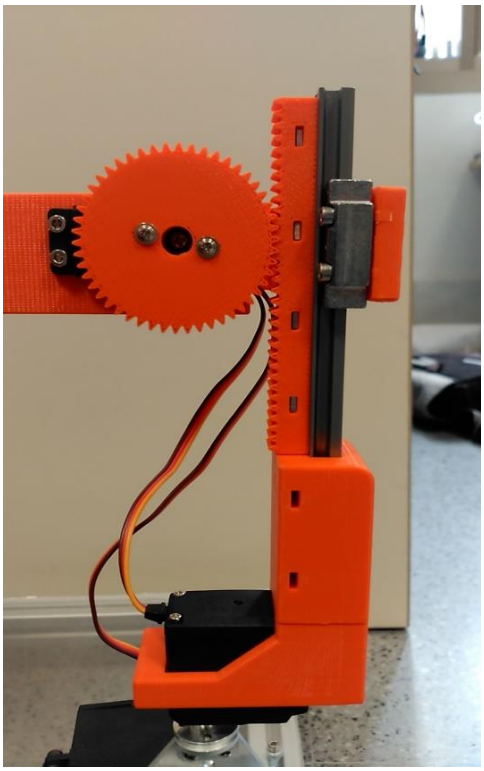


Ilustración 53: Sistema completo de piñón-cremallera y soporte para pinza ya ensamblado

COTAS DE SOPORTE PARA PINZA

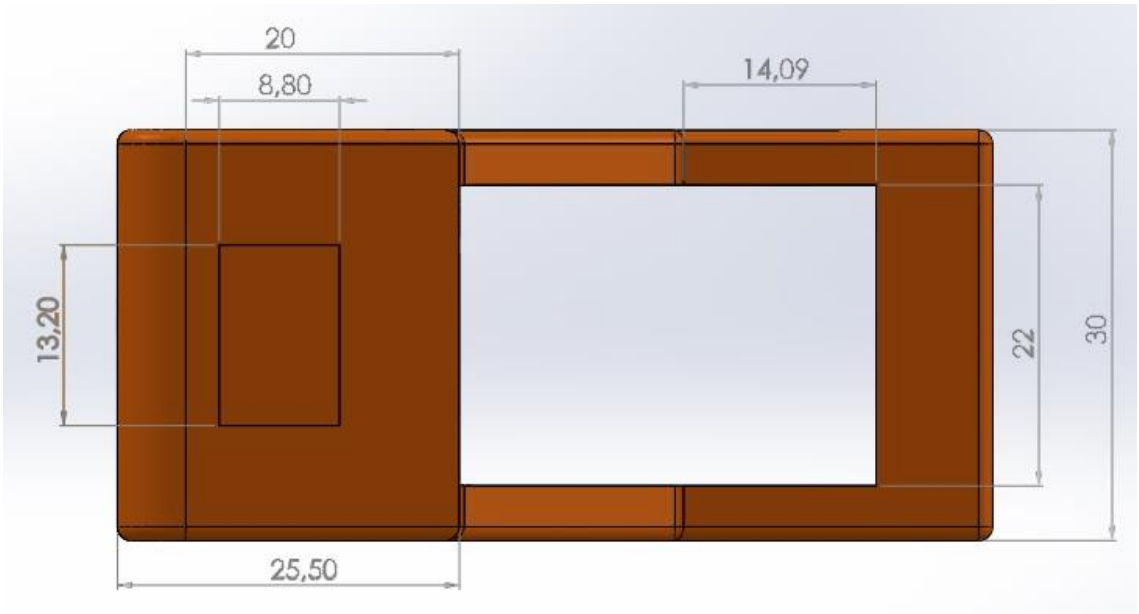


Ilustración 54: Soporte de pinza planta 1

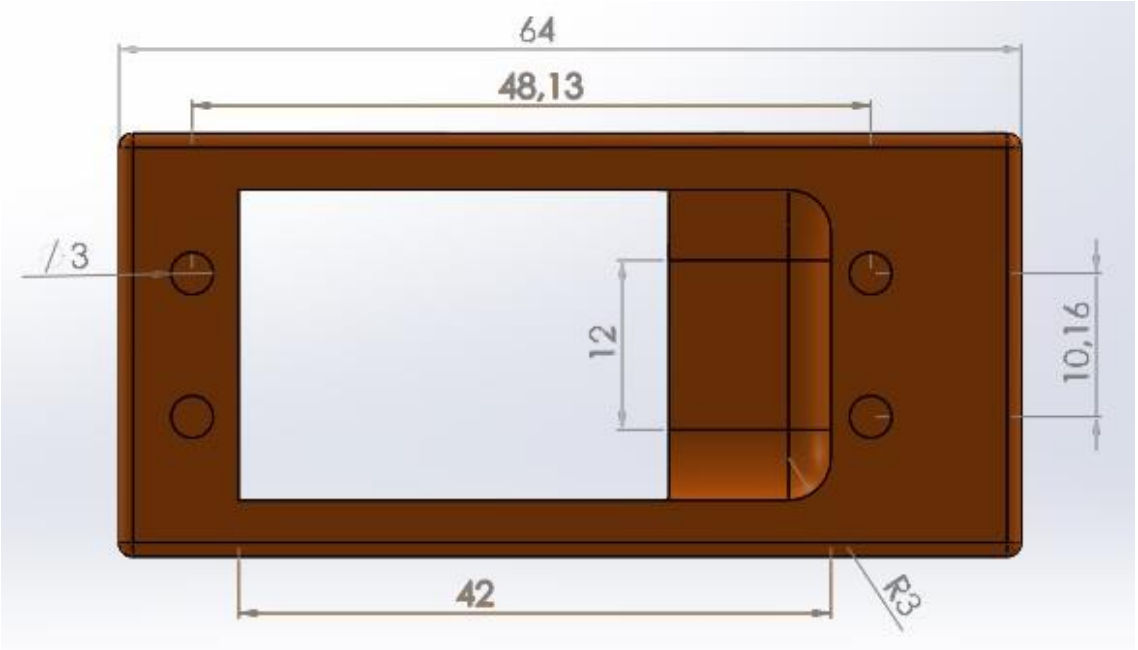


Ilustración 553: Soporte de pinza planta 2

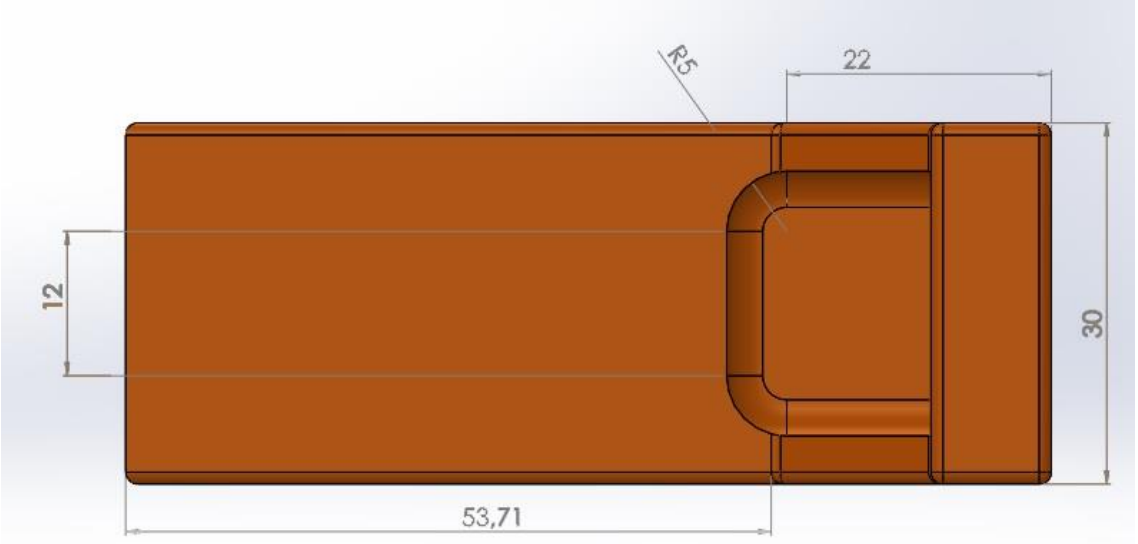


Ilustración 56: Soporte de pinza de perfil 1

Ilustración 57: Soporte de pinza de perfil 2

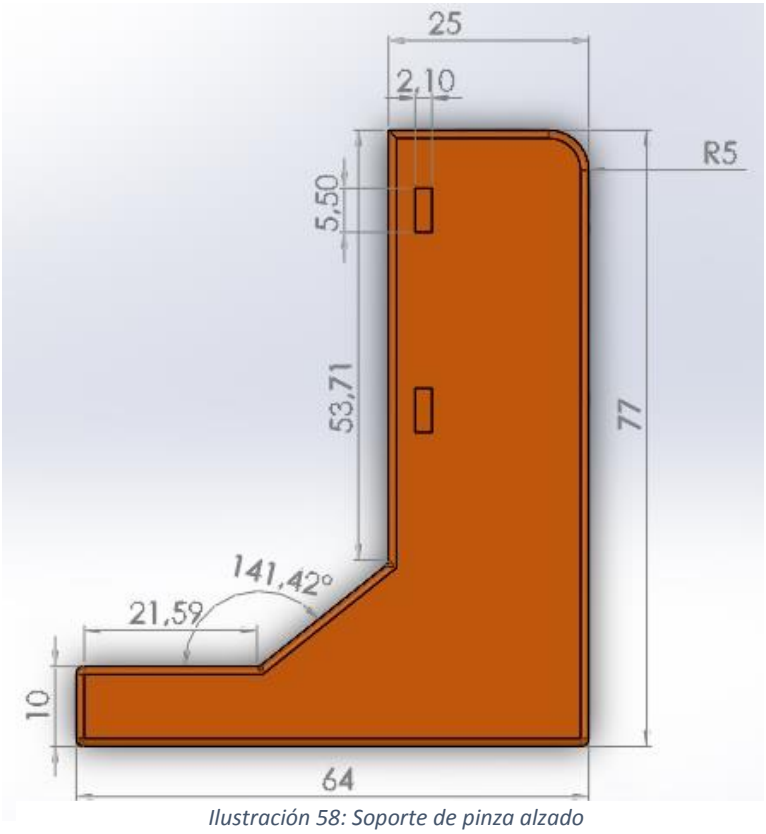


Ilustración 58: Soporte de pinza alzado

3.2.7. Pinza

Junto con los servomotores (lógicamente), la pinza ha sido el único elemento comprado, y no diseñado. Ya que se llegó a la conclusión, por experiencia del tutor en proyectos similares, que la mejor manera de asegurar un buen funcionamiento de dicha pieza, sin riesgos del tipo rotura, era adquirirla.

Nos decantamos por una pinza de *Leantec robotics&electronics*, la pinza es de aluminio en su totalidad, por lo que aseguramos la máxima ligereza, además es exclusiva para los servomotores *Futaba s3003* utilizados en nuestro proyecto.

La pinza tiene una apertura máxima de 50mm entre patas.

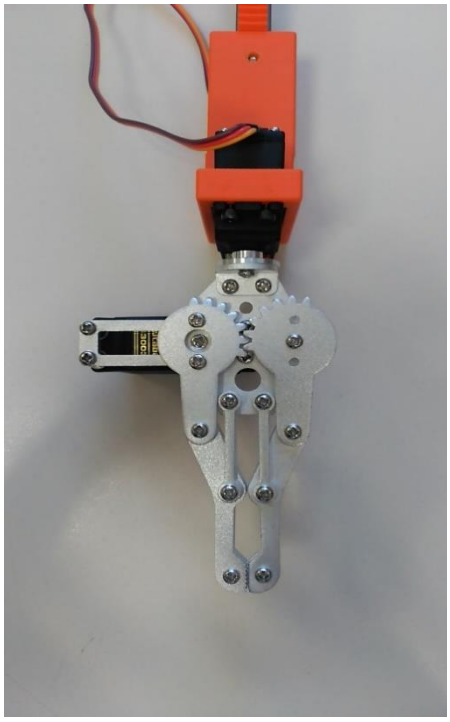


Ilustración 59: Pinza cerrada montaje acabado

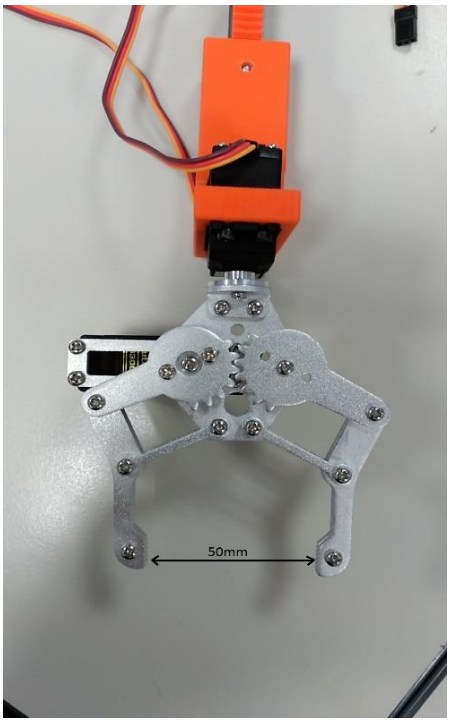


Ilustración 60: Pinza abierta 50mm

3.2.8. Bolsillo para engranajes

Como se ha comentado anteriormente, se han realizado piñones de tres tipos de diámetros, para sacarle el máximo partido posible al par suministrado por el servomotor, así que se ha realizado, también en SolidWorks, un bolsillo, que se atornillará a la estructura metálica con un tornillo tipo Allen de métrica 4 para tener fácil acceso a cualquiera de los tres diámetros de piñón.

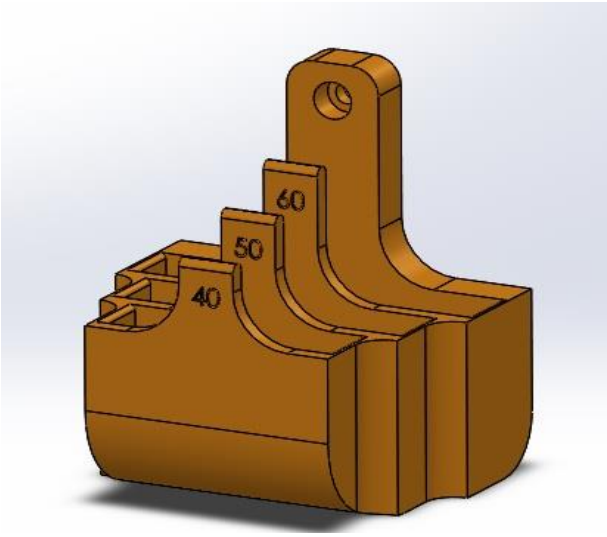


Ilustración 61: Bolsillo para engranajes en SolidWorks

3.2.9. Caja de Arduino

Como se ha comentado al principio del apartado "Prototipo Definitivo", y más tarde se explicará más detalladamente, hemos utilizado Arduino Mega2560 y no Arduino Uno.

A dicho Arduino se le ha realizado una caja dónde se depositará Arduino junto con la placa electrónica para tener todo más protegido frente a posibles golpes o desconexiones accidentales. Dicha caja cuenta con agujeros para la refrigeración de la electrónica.

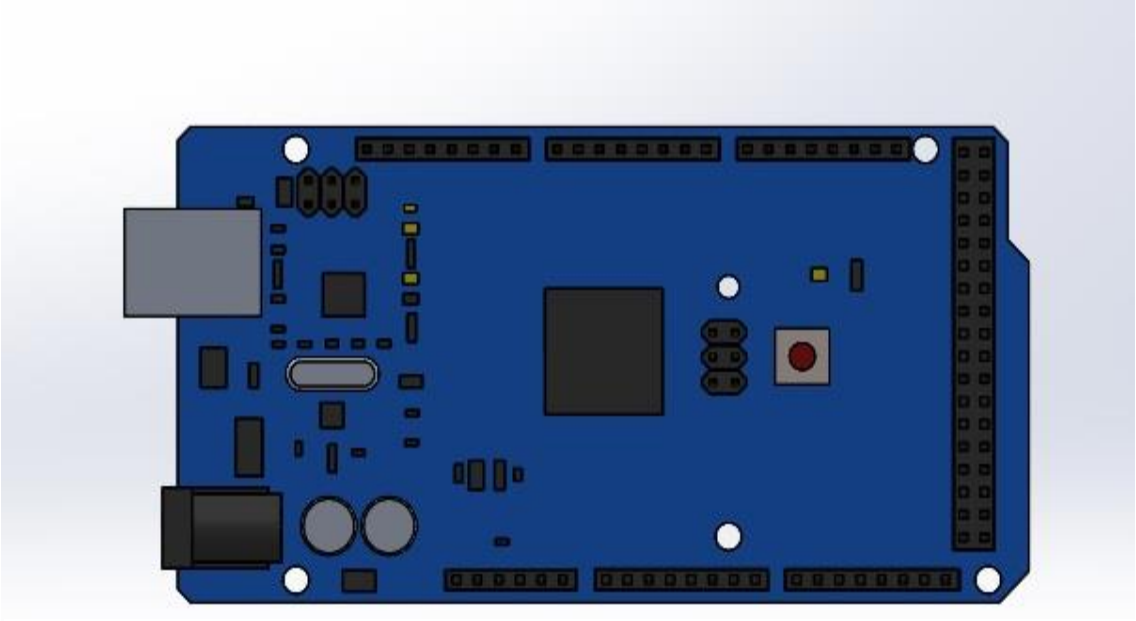


Ilustración 62: Arduino Mega2560 diseñado en SolidWorks.

COTAS DE CAJA PARA ELECTRÓNICA

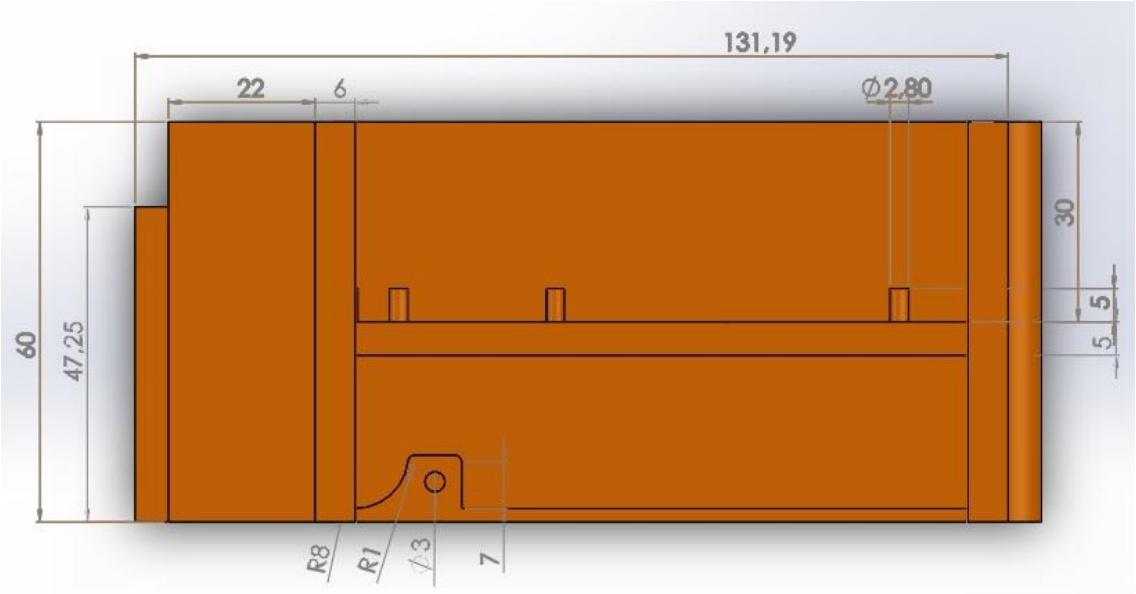


Ilustración 63: Base caja

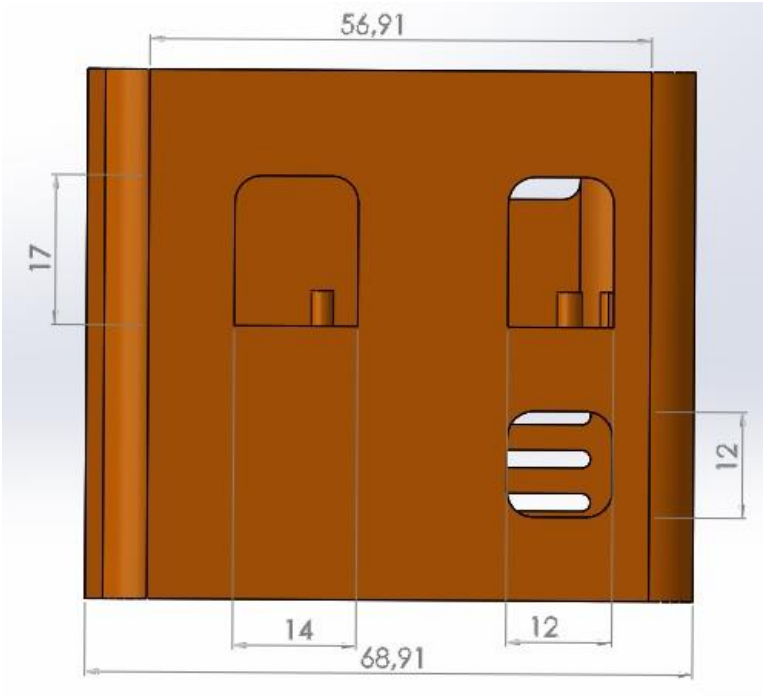


Ilustración 64: Base caja 2

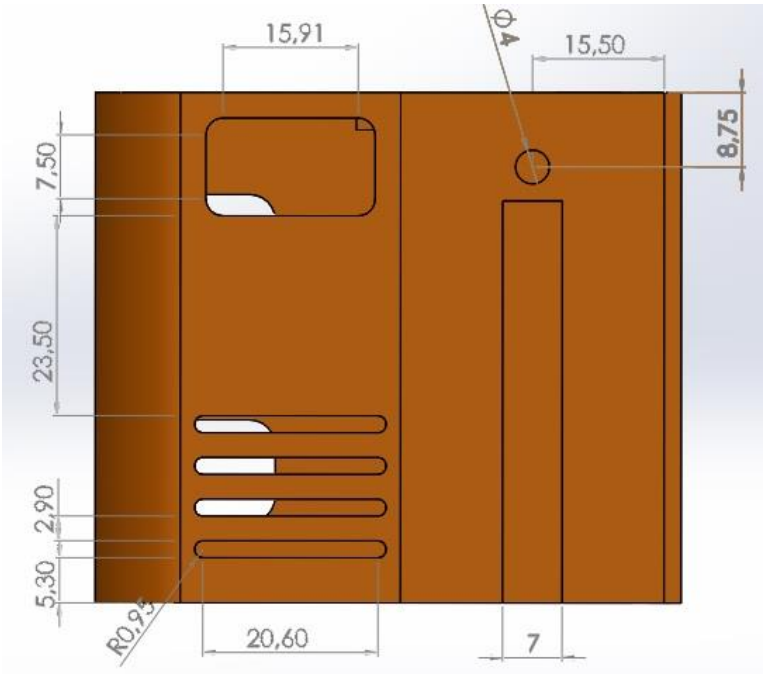


Ilustración 65: Base caja 3

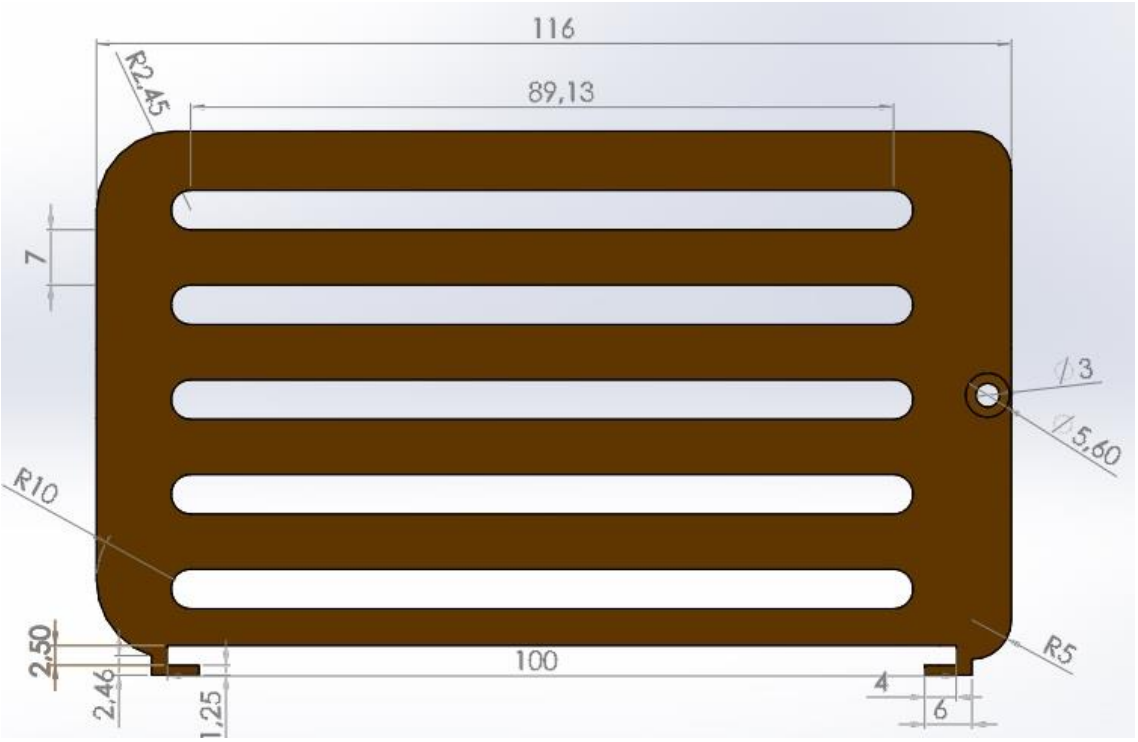


Ilustración 66: Tapadera blanda

Ilustración 67: Tapadera alzado

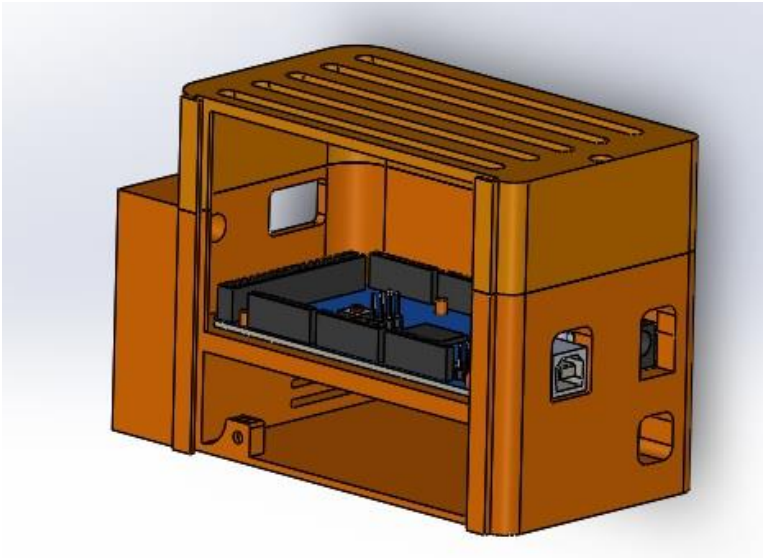


Ilustración 68: Caja abierta

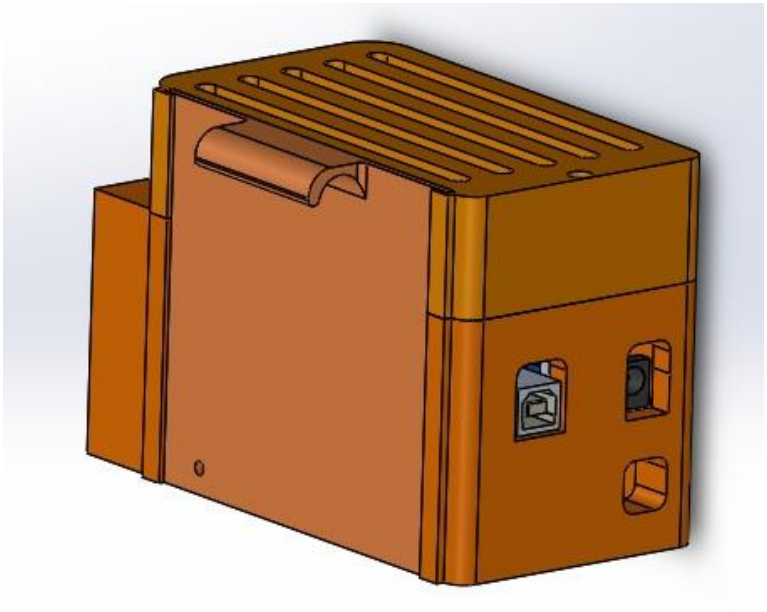


Ilustración 69: Caja cerrada

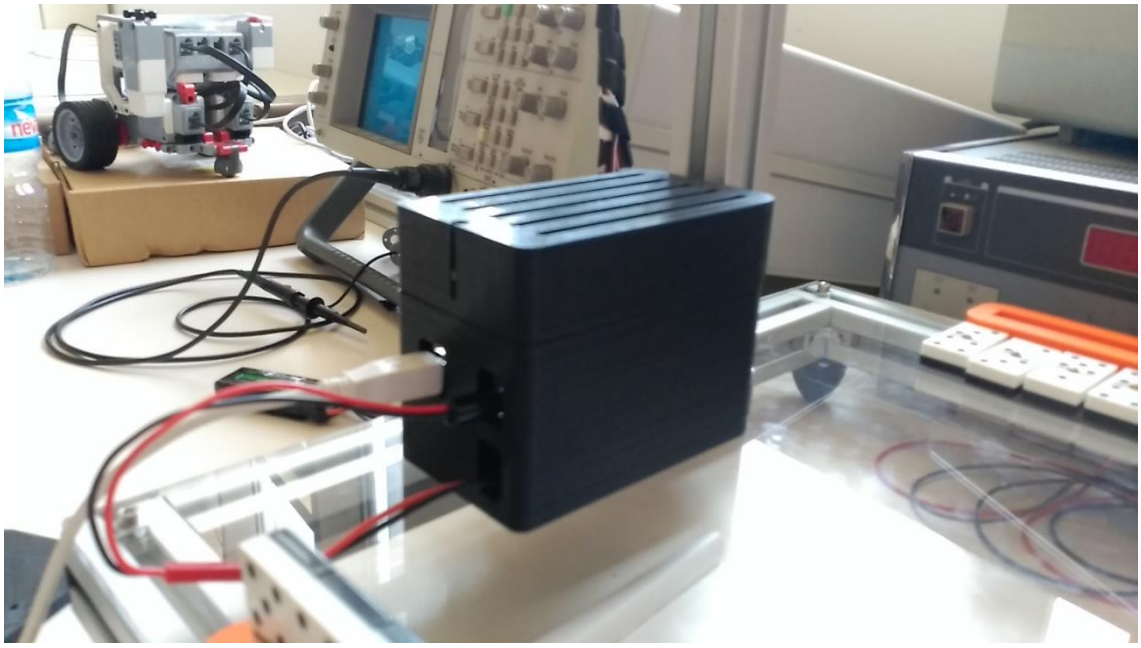


Ilustración 70: Imagen real de caja

3.2.10. Soporte fichas de dominó

Soporte realizado para la demostración del programa con las fichas de dominó.

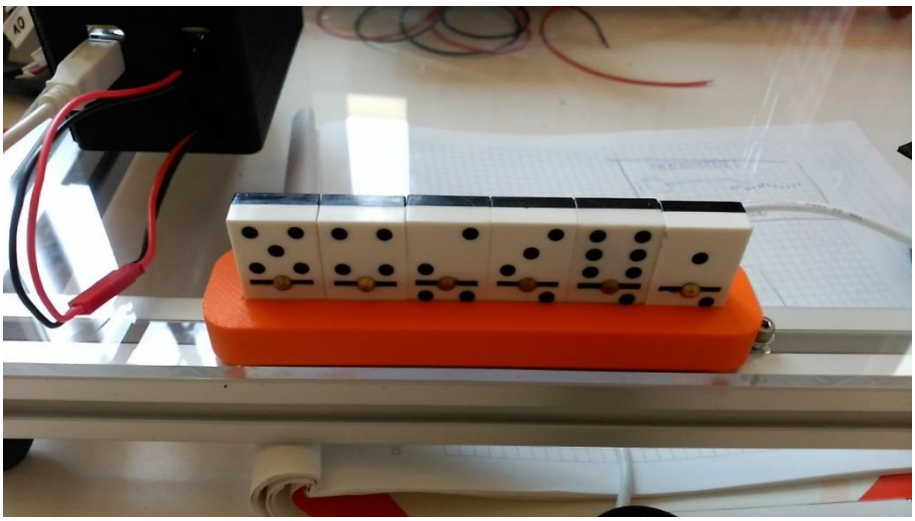


Ilustración 71: Imagen real soporte fichas

COTAS SOPORTE

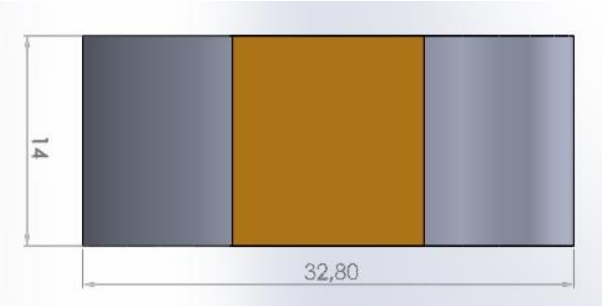


Ilustración 72: Perfil soporte dominó

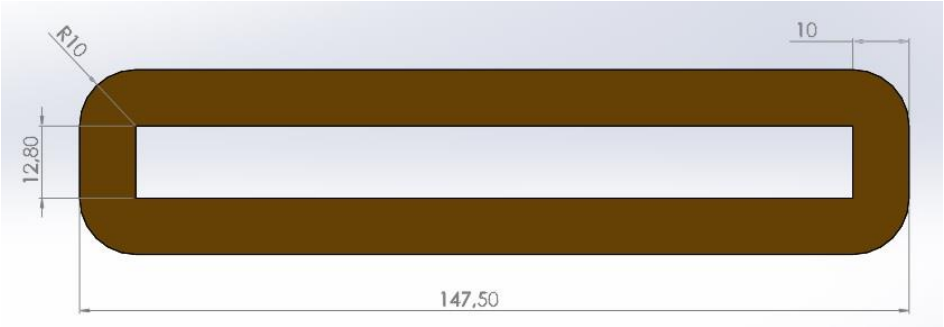


Ilustración 73: Planta soporte dominó

3.3. Análisis de los servomotores

Para el correcto funcionamiento del robot *SCARA* han sido necesarios 5 servomotores *Futaba s3003*, todos iguales.

Se han colocado dos servomotores, uno en el hombro y otro en codo, para su correcta articulación sobre el plano. Otro servomotor en la muñeca, que mediante un piñón de diámetro 50mm mueve una cremallera de 100x5mm.

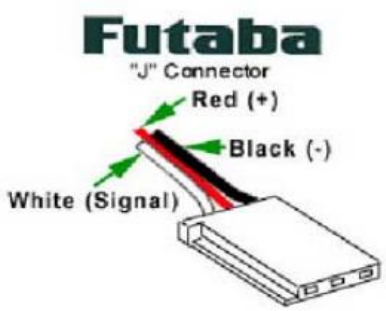
Al final de dicha cremallera encontramos dos servomotores, uno para abrir y cerrar la pinza y otro para conseguir rotar dicha pinza.

Cabe recordar que para tener más variaciones y comodidad a la hora de trabajar con nuestro *SCARA* se ha diseñado un sistema, el cual se ha mostrado anteriormente, para poder cambiar el diámetro de nuestro piñón de 40 a 60 milímetros. Así conseguimos jugar con la relación Par (T) y Radio (r) para conseguir más o menos Fuerza (N).

$$\frac{T\,(N/m)}{r\,(m)} = F\,(N)$$

Los servomotores serán alimentados a 5 voltios, ya que el rango de funcionamiento es entre 4,8V y 6V.

DATASHEET SERVOMOTOR FUTABA s3003



...S3003 FUTABA SERVO...

Detailed Specifications			
Control System:	+Pulse Width Control 1520usec Neutral	Current Drain (4.8V):	7.2mA/idle
Required Pulse:	3-5 Volt Peak to Peak Square Wave	Current Drain (6.0V):	8mA/idle
Operating Voltage:	4.8-6.0 Volts	Direction:	Counter Clockwise/Pulse Traveling 1520-1900usec
Operating Temperature Range:	-20 to +60 Degree C	Motor Type:	3 Pole Ferrite
Operating Speed (4.8V):	0.23sec/60 degrees at no load	Potentiometer Drive:	Indirect Drive
Operating Speed (6.0V):	0.19sec/60 degrees at no load	Bearing Type:	Plastic Bearing
Stall Torque (4.8V):	44 oz/in. (3.2kg.cm)	Gear Type:	All Nylon Gears
Stall Torque (6.0V):	56.8 oz/in. (4.1kg.cm)	Connector Wire Length:	12"
Operating Angle:	45 Deg. one side pulse traveling 400usec	Dimensions:	1.6" x 0.8"x 1.4" (41 x 20 x 36mm)
360 Modifiable:	Yes	Weight:	1.3oz. (37.2g)

3.4. Circuito de potencia

El circuito de potencia ha sido realizado con la idea de que sea una placa universal para cualquier tipo de Arduino, y no solo eso, si no aprovechando todo lo posible los pines de Arduino, además pensando en aplicaciones para Labview, programa con el que se ha programado el robot SCARA, se han implementado elementos en la placa para la realización de programas con dicho software, como LEDES, pulsadores o potenciómetro.

3.4.1. Arduino

Arduino (Genuino a nivel internacional hasta octubre 2016), es una compañía de hardware libre y una comunidad tecnológica que diseña y manufactura placas computadora de desarrollo de hardware y software, compuesta respectivamente por circuitos impresos que integran un microcontrolador y un entorno de desarrollo (IDE), en donde se programa cada placa.

Arduino se enfoca en acercar y facilitar el uso de la electrónica y programación de sistemas embebidos en proyectos multidisciplinarios. Toda la plataforma, tanto para sus componentes de hardware como de software, son liberados con licencia de código abierto que permite libertad de acceso a ellos.

El hardware consiste en una placa de circuito impreso con un microcontrolador, usualmente Atmel AVR, puertos digitales y analógicos de entrada/salida, los cuales

pueden conectarse a placas de expansión (shields), que amplían las características de funcionamiento de la placa Arduino. Asimismo, posee un puerto de conexión USB desde donde se puede alimentar la placa y establecer comunicación con el computador.

Por otro lado, el software consiste en un entorno de desarrollo (IDE) basado en el entorno de Processing y lenguaje de programación basado en Wiring, así como en el cargador de arranque (bootloader) que es ejecutado en la placa. El microcontrolador de la placa se programa mediante un computador, usando una comunicación serial mediante un convertidor de niveles RS-232 a TTL serial.

La primera placa Arduino fue introducida en 2005, ofreciendo un bajo costo y facilidad de uso para novatos y profesionales. Buscaba desarrollar proyectos interactivos con su entorno mediante el uso de actuadores y sensores. A partir de octubre de 2012, se incorporaron nuevos modelos de placas de desarrollo que usan microcontroladores Cortex M3, ARM de 32 bits, que coexisten con los originales modelos que integran microcontroladores AVR de 8 bits. ARM y AVR no son plataformas compatibles en cuanto a su arquitectura y por lo que tampoco lo es su set de instrucciones, pero se pueden programar y compilar bajo el IDE predeterminado de Arduino sin ningún cambio.

Las placas Arduino están disponibles de dos formas: ensambladas o en forma de kits "Hazlo tú mismo" (por sus siglas en inglés "DIY"). Los esquemas de diseño del Hardware están disponibles bajo licencia Libre, con lo que se permite que cualquier persona pueda crear su propia placa Arduino sin necesidad de comprar una prefabricada. Adafruit Industries estimó a mediados del año 2011 que, alrededor de 300,000 placas Arduino habían sido producidas comercialmente y en el año 2013 estimó que alrededor de 700.000 placas oficiales de la empresa Arduino estaban en manos de los usuarios.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software tal como Adobe Flash, Processing, Max/MSP, Pure Data, etc. Una tendencia tecnológica es utilizar Arduino como tarjeta de adquisición de datos desarrollando interfaces en software como JAVA, Visual Basic y LabVIEW . Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo interactivo libre se puede descargar gratuitamente.

Para la realización del proyecto se ha utilizado **Arduino Uno**, el cual te permite, mediante sus **entradas PWM**, el control de nuestros servomotores (más adelante se explicará y mostrarán todas las conexiones utilizadas en Arduino).

Arduino Uno te da la opción de alimentar a 5V, 3.3V o Vin, esta última para poder alimentar desde una fuente externa a la tensión deseada, en nuestro caso, como contamos con un regulador de 5V y es aconsejable alimentarlo con un mínimo de dos voltios por encima como mínimo, lo alimentaremos desde Vin para poder suministrarle un mínimo de 7-7.5V.

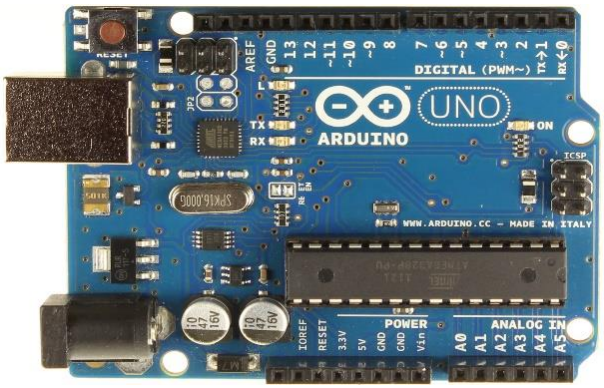


Ilustración 74: Arduino Uno

Arduino Uno es una herramienta sencilla, pero poco potente, por lo que se requiere de un mejor circuito de potencia cuando queremos realizar trabajos más competitivos y completos. Esto se debe al regulador de tensión que trae incorporado Arduino, por lo que en este proyecto se ha realizado una placa electrónica con un regulador 7805 de 5v y 1A, muy fácil de conseguir y de tamaño idóneo para la placa.

Lo primero que debemos comentar antes de continuar es que, aunque la intención fue realizar el proyecto con Arduino Uno nos dimos cuenta que dicho Arduino, aunque la placa diga que cuenta con 6 entradas PWM (3,5,6,9,10,11), realmente solo puedes utilizar simultáneamente 3 de ellas, es decir, que tienes puedes utilizar cualquiera de las 6 pero con un máximo de tres a la vez, en el orden que se quiera, pero sólo 3. Esto es debido a su microcontrolador Atmega328 solo dispone de 3 PWM, por lo que tuvimos que cambiar a Arduino Mega, ya que nosotros disponemos de 5 servomotores, por lo que nos era imposible trabajar con únicamente 3 entradas.

Realmente no nos ha causado ningún tipo de problema puesto que la placa es totalmente válida tanto para uno modelo como para otro, y el regulador de ambos Arduinos también es el mismo, por lo que seguimos necesitando de la placa de potencia para el correcto funcionamiento. Simplemente Arduino Mega nos ocupara un poco más de espacio.

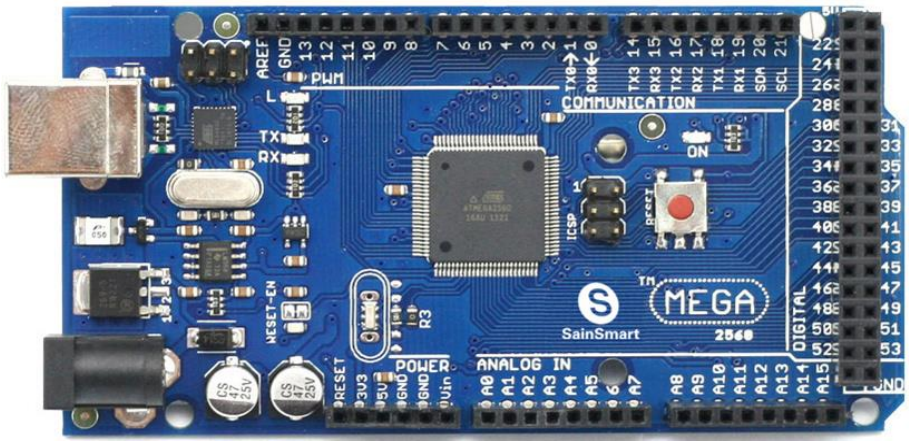


Ilustración 75: Arduino Mega2560

3.4.2. Esquemático

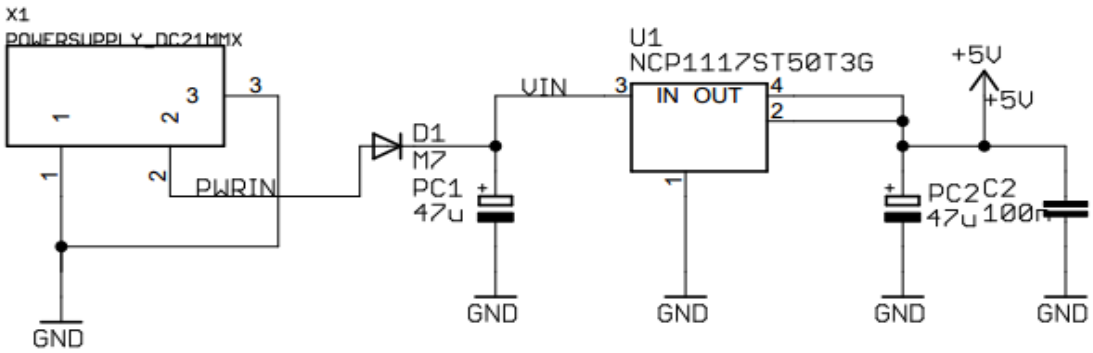


Ilustración 76: Regulador NCP117 de Arduino

Para el diseño de la electrónica se ha utilizado el software *KiCad*. Es un software utilizado para el diseño de circuitos electrónicos ya que nos permite la realización tanto del circuito esquemático como el PCB (Printed Circuit Board) además de una visualización 3D de la placa.

Es un software libre muy flexible y adaptable por lo que resulta una buena elección para la realización de la placa, que posteriormente será hecha en el laboratorio de electrolítica.

La placa será un *Shield* que conectaremos sobre Arduino.

A continuación, se mostrará y explicará todas las partes del esquemático, dónde se ha diseñado todo separado y etiquetado para una perfecta comprensión de la placa.

Se mostrará una imagen del esquemático completo y después parte a parte para una mejor comprensión.

Nota: La imagen completa del esquemático realizado en *KiCad* se encuentra en ANEXO I. (página 76).

3.4.2.1. Shield para Arduino Uno

Aquí podemos ver el esquemático del shield para Arduino Uno con todas sus patillas etiquetadas para evitar la mayor cantidad posible de conexiones mediante líneas, así el diseño gana en claridad y comprensión.

Seguido del shield para Arduino Uno se mostrará una tabla realizada para explicar en que se han utilizado todos y cada uno de los pines de la placa.

Nota: Dicho shield se ha obtenido de una biblioteca para KiCad del docente Juan Ramón Rufino.

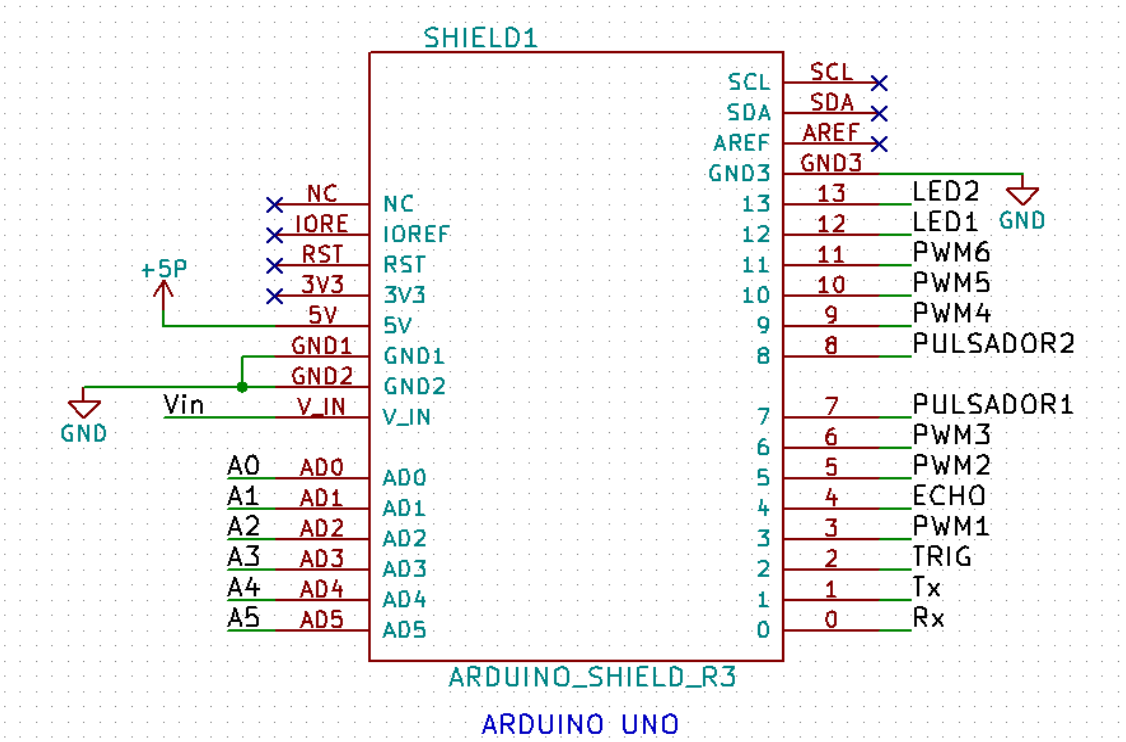


Ilustración 77: Patillaje de Arduino Uno utilizado para el proyecto

POWER	
5V	Punto de alimentación a 5V para aplicaciones independientes del regulador
GND	Masa
GND	Masa
Vin	Punto de alimentación de entrada para alimentar nuestro regulador 7805
ANALOG IN	
A0	Entrada analógica 0, conectada a potenciómetro para regular la entrada de 5V
A1	Entrada analógica 1, con conexión de 3 pines para su aprovechamiento
A2	Entrada analógica 2, con conexión de 3 pines para su aprovechamiento
A3	Entrada analógica 3, con conexión de 3 pines para su aprovechamiento
A4	Entrada analógica 4, con conexión de 3 pines para su aprovechamiento
A5	Entrada analógica 5, con conexión de 3 pines para su aprovechamiento
DIGITAL (PWM~)	
0->Rx	Recepción de datos para conexión serie
1<-Tx	Transmisión de datos para conexión serie
2	Entrada digital 2, en este caso al TRIG del ultrasonido
3~	Entrada PWM-3, con conexión de 3 pines para servomotor *Futaba s3003*
4	Entrada digital 4, en este caso al ECHO del ultrasonido
5~	Entrada PWM-5, con conexión de 3 pines para servomotor *Futaba s3003*
6~	Entrada PWM-6, con conexión de 3 pines para servomotor *Futaba s3003*
7	Entrada digital 7, en este caso a pulsador
8	Entrada digital 8, en este caso a pulsador
9~	Entrada PWM-9, con conexión de 3 pines para servomotor *Futaba s3003*
10~	Entrada PWM-10, con conexión de 3 pines para servomotor *Futaba s3003*
11~	Entrada PWM-11, con conexión de 3 pines para servomotor *Futaba s3003*
12	Entrada digital 12, en este caso a LED
13	Entrada digital 13, en este caso a LED
GND	Masa

Ilustración 78: Tabla con todas las patillas utilizadas en Arduino

3.4.2.2. Regulador de tensión

Cabe mencionar, que el primer diseño y prototipo fue realizado con un regulador 7805 de 5V y 1A creyéndolo suficiente para alimentar los cinco servomotores, pero nos encontramos con el problema de que no, no era suficiente un amperio ya que comprobamos con una fuente de alimentación que el consumo de nuestro sistema estaba muy próximo a un amperio, por lo que decidimos utilizar un regulador, también de la familia 7805 pero esta vez de 2,2 amperios.

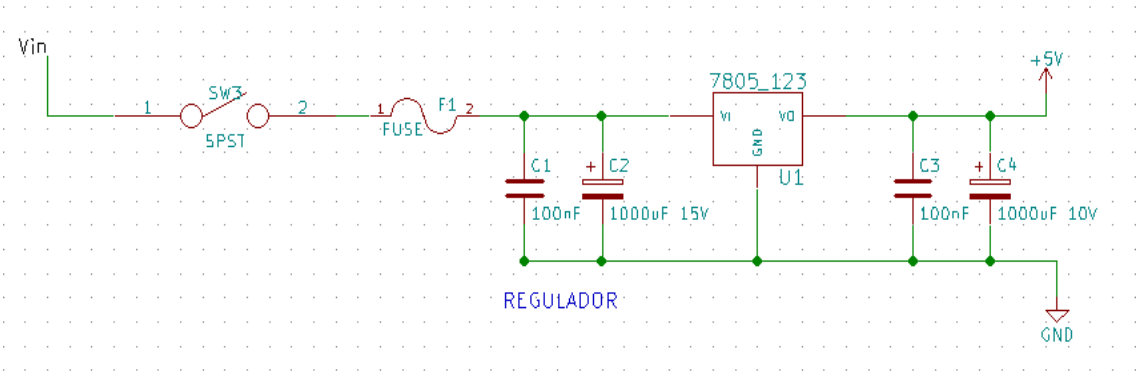


Ilustración 79: Bloque esquemático del regulador de tensión

Como no solo el regulador nos dio problemas, optando por utilizar el de 2,2 amperios, sino que también la alimentación de entrada, al estar suministrando una tensión un poco inferior a la recomendada, tuvimos el problema, observado en el osciloscopio, que la tensión de salida del regulador no eran 5 voltios constantes, si no que era una señal que sufría picos, por lo que optamos por alimentar mediante una batería de tipo LiPo de 7,4V. Las baterías LiPo se caracterizan por ser ligeras y por poder almacenar una gran cantidad de energía.

Dicha batería es la ideal para alimentar el sistema ya que en la entrada del regulador debes tener, como mínimo un par de voltios por encima de lo que vas a suministrar, es decir, un mínimo de 7 voltios (o próximo a 7V). Además, esta batería tiene una tasa descarga 30C y una capacidad de 800mAh, por lo que puede suministrar 24A, sabiendo que iremos sobrados para alimentar nuestros cinco servomotores.

Siendo Vin una batería LiPo, el siguiente elemento que encontramos en un interruptor para poder encender y apagar dicha alimentación, tras este se ha colocado un fusible a modo de elemento de seguridad.

Ilustración 80: Batería LiPo 7,4V

Llegado al regulador 7805 de 5V y 2.2A se ha instalado junto con unos condensadores electrolíticos tanto a la entrada como a la salida y junto a estos, en paralelo, uno cerámico. La función de dichos electrolíticos es la de filtrar el rizado de tensión proveniente de Vin, ya que dicho tipo de condensadores son de mucha capacidad, el inconveniente que tienen es que no funcionan bien a muy altas frecuencias, por lo que se añaden los cerámicos con baja ESR (resistencia serie equivalente) en paralelo.

Los condensadores electrolíticos puestos en el esquemático son de 1000µF y 35V a la entrada y 10V a la salida.

3.4.2.3. Entradas analógicas

La diferencia entre una entrada analógica y una digital es que la digital solo puedo entender dos niveles de señal, LOW o valores cercanos a 0 V y HIGH o valores cercanos a 5 V, mientras que las entradas analógicas se caracterizan por leer valores de tensión de 0 a 5 Voltios con una resolución de 1024 (10 bits). Por eso hemos conectado un potenciómetro a una entrada analógica, para así poder variar la tensión de entrada directa desde Arduino, ya que las entradas analógicas, desde A0 hasta A5 están conectadas a los 5V de Arduino y no al regulador.

En el circuito se han colocado conexiones de 3 pines (Ax, GND, 5V) para ser usadas en la aplicación que se desee.

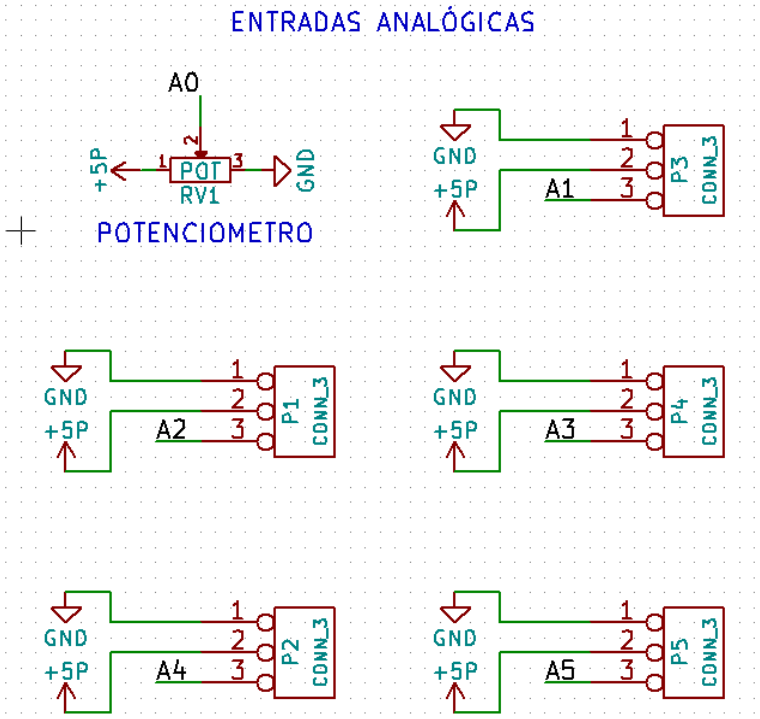


Ilustración 81: Entradas analógicas de Arduino

3.4.2.4. Entradas digitales

Como se ha comentado en el punto anterior, las entradas digitales (1-13) solo pueden entender dos niveles de señal, LOW o valores cercanos a 0 V y HIGH o valores cercanos a 5 V. Esto nos puede servir cuando únicamente nos interese conocer dos estados (pulsadores, LEDES, etc) o leer señales de pulsos digitales. Hemos colocado seis entradas de tres pines para la colocación de nuestros cinco servomotores, sobrando una para futuras ampliaciones.

Una entrada de cuatro pines pensada para colocar un ultrasonido Arduino (sensor de distancia), muy útil para aplicaciones de Arduino.

Otra entrada, también de cuatro pines, conectadas a la comunicación serie de Arduino (Rx y Tx) pensada, entre otras cosas, para conexión con módulo Bluethoot

Dos pulsadores, pensados también, para futuras aplicaciones de esta placa, ya que lo que buscamos es hacerla lo más universal posible, intentado que se adapte a cualquier aplicación realizada con Arduino.

Por último, hemos colocado dos Ledes, cuya funcionalidad es simplemente informativa, es decir, saber visualmente si nuestra placa está funcionando o no.

Todas estas aplicaciones, como anteriormente se mencionó, también servirán para realizar pequeñas aplicaciones con Labview que te permitan encender y apagar ledes, utilizar los pulsadores, etc.

Cabe mencionar el criterio utilizado para la elección de las resistencias, tanto para los ledes como para los pulsadores.

1. **Ledes:** Los ledes estándar, como es este caso, tienen una corriente de alimentación de 20mA, por lo que hay que colocar una resistencia para no superar dicha corriente. Este tipo de ledes tienen una corriente de alimentación de 2.2~2,6V, como podemos observar en la siguiente imagen.

Led de señal estándar

Standard signal leds

Cápsula tintada difusa.
Diffused tinted lens.

REFERENCIA Y COLOR PART NUMBER AND COLOR	CÁPSULA (DIÁMETRO) PACKAGE	LUMINOSIDAD LUMINOUS INTENSITY	LONGITUD DE ONDA WAVELENGTH	ÁNGULO VIEWING ANGLE	CORRIENTE DE ALIMENTACIÓN FORWARD CURRENT	TENSIÓN DE ALIMENTACIÓN FORWARD VOLTAGE
BL-B5141	3 mm	10 mcd	700 nm	40º	20 mA	2,2~2,6 VDC
BL-B2141	3 mm	40 mcd	568 nm	35º	20 mA	2,2~2,6 VDC
BL-B3141	3 mm	30 mcd	585 nm	35º	20 mA	2,2~2,6 VDC
BL-B5134	5 mm	12 mcd	700 nm	35º	20 mA	2,2~2,6 VDC
BL-B2134	5 mm	80 mcd	568 nm	35º	20 mA	2,2~2,6 VDC
BL-B3134	5 mm	70 mcd	585 nm	35º	20 mA	2,2~2,6 VDC

En la siguiente imagen, vemos la curva característica de los ledes estándar y sabiendo que tenemos un led verde y otro rojo hemos optado por coger un valor medio de la tensión para 15mA, por lo que como V(rojo)=1,5Vy V(verde)=3,2Vtendremos un valor medio de V=2,35V.

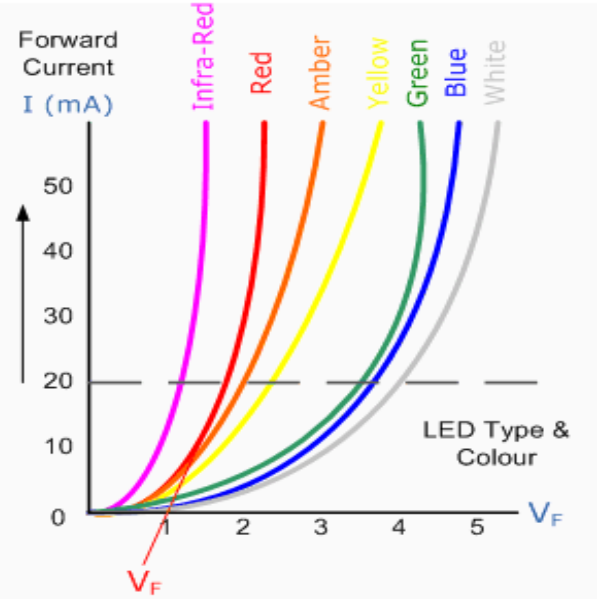


Ilustración 82: Curva característica de Ledes estandar

Ahora podemos calcular que resistencia se va a necesitar para nuestros ledes.

$$5\text{V}-2{,}6\,V=0{,}017\,AR(\Omega)$$

Dónde R=141.176 (Ω), lógicamente este valor no se encuentra comercialmente, el más próximo es una R de 180 ohmios, pero decidimos utilizar una más grande (330 ohmios), tanto por disponibilidad del material como para poder utilizar cualquier tipo de led.

Comprobamos la intensidad obtenida para esta resistencia;

$$I=\frac{5\text{V}-2{,}6\,V}{330\,\Omega}=7{,}27\,mA$$

2. **Pulsadores:** Los pulsadores, como se dijo anteriormente pueden ser utilizados, entre otras aplicaciones para realizar aplicaciones con Labview, y necesariamente necesitan de una resistencia, porque de lo contrario, como vamos a ver en el siguiente dibujo, cuando el pulsador se encuentre abierto quedará un cable al aire en el integrado, y como la entrada de dicho integrado es de alta impedancia el cable al aire hará de "antena" provocando que cualquier ruido electromagnético generado en este cable nos cause pulsos no deseados.

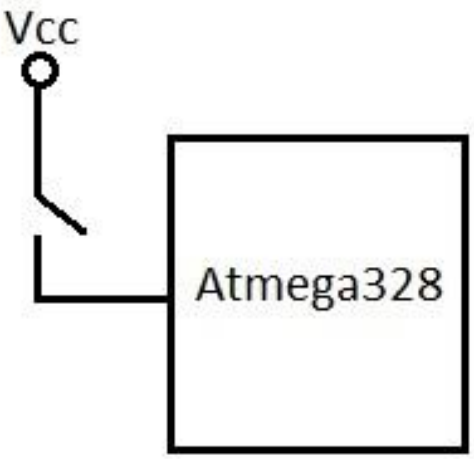
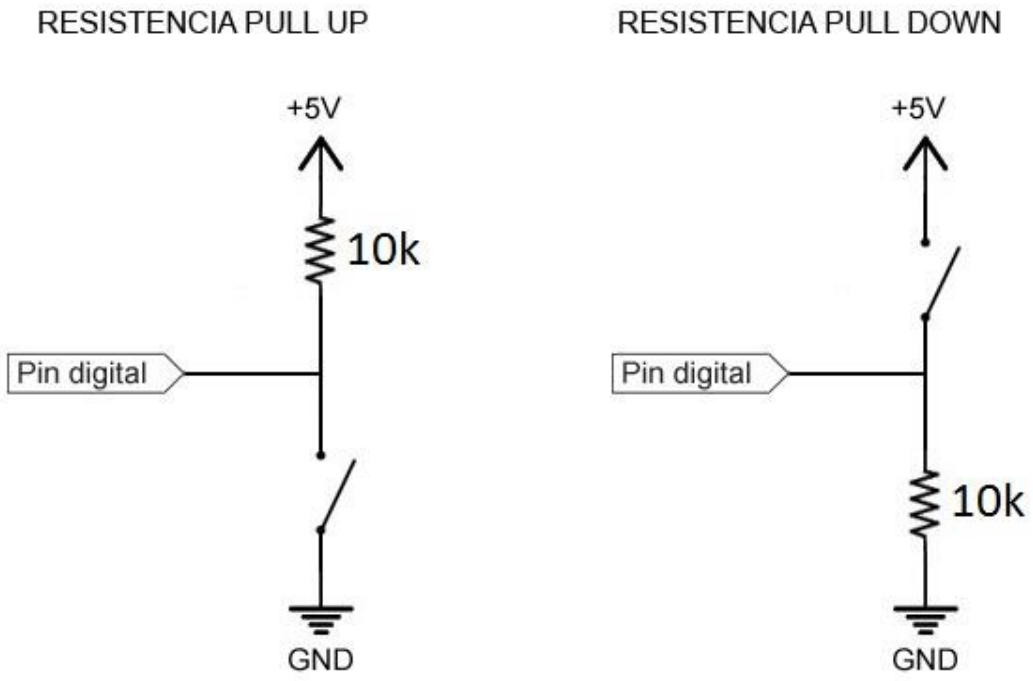


Ilustración 83: Pulsador sin resistencia

Para que esto no ocurra podemos utilizar dos tipos distintos de configuración, **Pull Down** o **Pull Up**.



En la configuración Pull Down, al haber una resistencia conectada a masa, cuando el pulsador se encuentre abierto, el microcontrolador quedará conectado a tierra, por lo tanto, siempre tendremos un 0 (0V). Podría darse el error de pensar que podemos conectar ese "hilo" directo a tierra sin pasar por una resistencia, pero al cerrar el pulsador lo que quedaría sería un cortocircuito.

Mientras en la configuración Pull Up obtenemos justo lo contrario, cuando el pulsador se encuentra abierto obtenemos el máximo valor.

El valor de la resistencia oscilará entre 1k y unos 100k ohmios aproximadamente, pero, ahora bien, si escogemos un valor pequeño, la corriente que circularía sería muy alta, por lo que se calentaría en exceso (desperdiciando energía), mucho peor si bajamos de 1k.

Si nos vamos a valores excesivamente altos pasaría lo contrario, la impedancia de entrada al micro sería tan grande que podría no leer bien los valores de tensión de entrada. Por lo que se ha elegido una resistencia de $R=10\text{k}\,\Omega$.

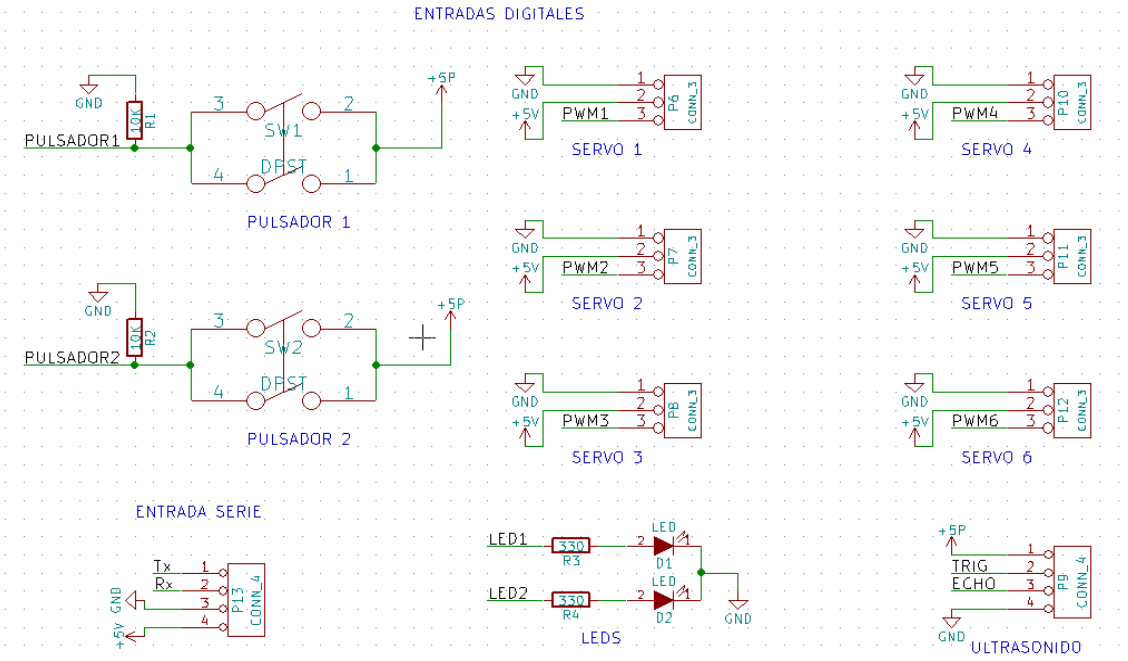


Ilustración 84: Entradas digitales de Arduino Uno

Hemos colocado seis entradas de tres pines para la colocación de nuestros cinco servomotores, sobrando una para futuras ampliaciones.

Una entrada de cuatro pines pensada para colocar un ultrasónico Arduino (sensor de distancia), muy útil para aplicaciones de Arduino.

Otra entrada, también de cuatro pines, conectadas a la comunicación serie de Arduino (Rx y Tx) pensada, entre otras cosas, para conexión con módulo Bluethoot

Dos pulsadores, pensados también, para futuras aplicaciones de esta placa, ya que lo que buscamos es hacerla lo más universal posible, intentado que se adapte a cualquier aplicación realizada con Arduino Uno.

Por último, hemos colocado dos Ledes, cuya funcionalidad es simplemente informativa, es decir, saber visualmente si nuestra placa está funcionando o no.

3.4.3. Netlist

NetList de nuestra placa, como se puede observar en la imagen, con la biblioteca suministrada por el grupo de robótica GROMEP para facilitar la realización de dicha placa en la universidad, ya que todos los componentes contienen las medidas adecuadas para ello.

```
 1       C1 -            100nF : jr_lib_pcb_g:M.C2
 2       C2 -       1000uF 15V : jr_lib_pcb_g:M.C2V10
 3       C3 -            100nF : jr_lib_pcb_g:M.C2
 4       C4 -       1000uF 10V : jr_lib_pcb_g:M.C2V10
 5       D1 -              LED : jr_lib_pcb_g:M.LEDV
 6       D2 -              LED : jr_lib_pcb_g:M.LEDV
 7       F1 -             FUSE : jr_lib_pcb_g:M.R4
 8       P1 -           CONN_3 : jr_lib_pcb_g:M.JUMPER3
 9       P2 -           CONN_3 : jr_lib_pcb_g:M.JUMPER3
10       P3 -           CONN_3 : jr_lib_pcb_g:M.JUMPER3
11       P4 -           CONN_3 : jr_lib_pcb_g:M.JUMPER3
12       P5 -           CONN_3 : jr_lib_pcb_g:M.JUMPER3
13       P6 -           CONN_3 : jr_lib_pcb_g:M.JUMPER3
14       P7 -           CONN_3 : jr_lib_pcb_g:M.JUMPER3
15       P8 -           CONN_3 : jr_lib_pcb_g:M.JUMPER3
16       P9 -           CONN_4 : jr_lib_pcb_g:M.JUMPER4
17      P10 -           CONN_3 : jr_lib_pcb_g:M.JUMPER3
18      P11 -           CONN_3 : jr_lib_pcb_g:M.JUMPER3
19      P12 -           CONN_3 : jr_lib_pcb_g:M.JUMPER3
20      P13 -           CONN_4 : jr_lib_pcb_g:M.JUMPER4
21       R1 -              10K : jr_lib_pcb_g:M.R4
22       R2 -              10K : jr_lib_pcb_g:M.R4
23       R3 -              330 : jr_lib_pcb_g:M.R4
24       R4 -              330 : jr_lib_pcb_g:M.R4
25      RV1 -              POT : jr_lib_pcb_g:M.POT-1
26   SHIELD1 - ARDUINO_SHIELD_R3 : jr_lib_pcb_g:ARDUINO_SHIELD_UNO_JR
27      SW1 -             DPST : jr_lib_pcb_g:M.SW_PUSH_SMALL-2
28      SW2 -             DPST : jr_lib_pcb_g:M.SW_PUSH_SMALL-2
29      SW3 -             SPST : jr_lib_pcb_g:M.JUMPER2
30       U1 -         7805_123 : jr_lib_pcb_g:M.TO220_VERT
```

3.4.4. PCB (Prited Circuit Board)

A continuación, se muestra como ha quedado el circuito tras colocar todos los elementos en PCB, intentando colocar todos los elementos de la manera más lógica a la hora de su futura utilización.

Las medidas para las pistas son de 40 milésimas de pulgada (1,016mm) para conexiones de Vcc y GND, pudiendo usar también 30 milésimas (0,762mm) si es necesario y de 20 milésimas de pulgada (0,508mm) para el resto de conexiones. Las vías se han realizado de 0.08 pulgadas (2,03mm) y un taladro de 0,8 milímetros.

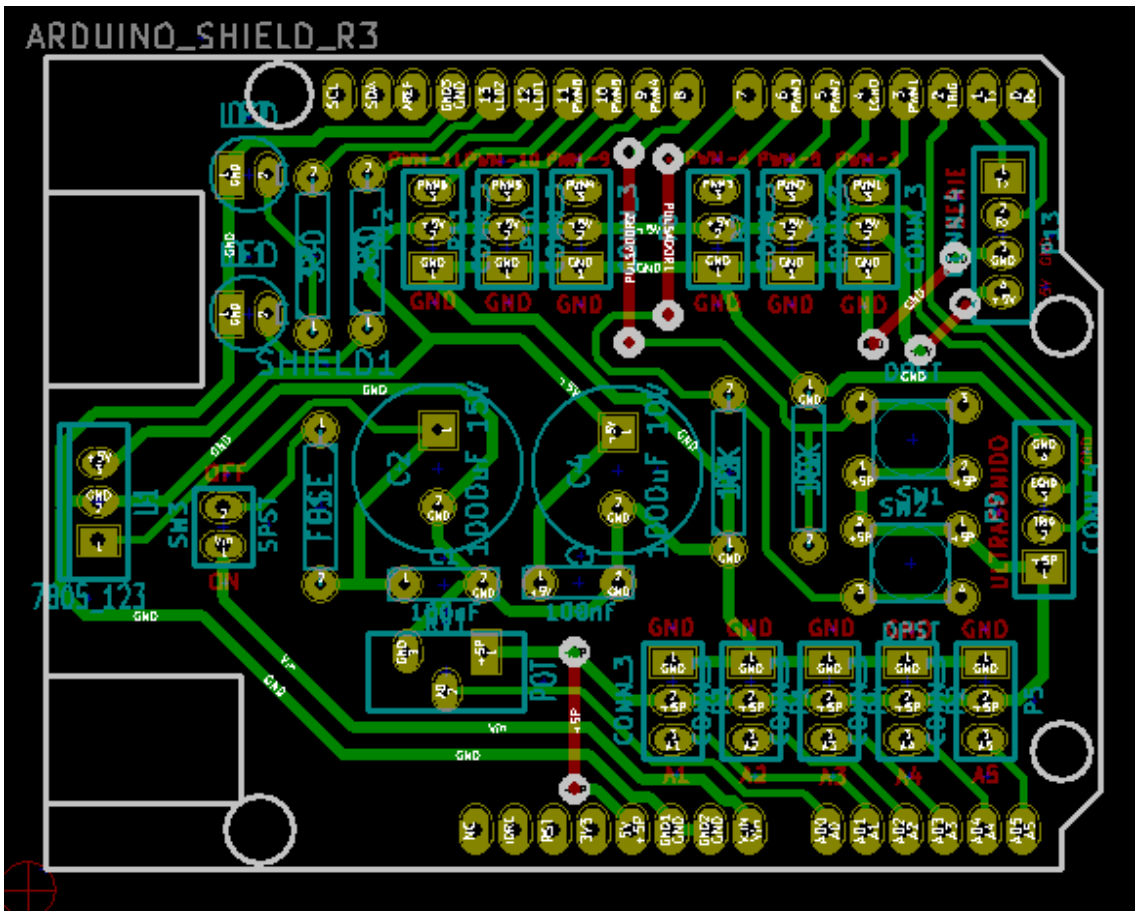


Ilustración 85: Imagen completa del circuito impreso, PCB (Printed Circuit Board)

Las entradas digitales han sido colocadas junto a dichos pines, además de ponerlos lo más cerca posible unos de otros para que nos resulte cómodo conectar los elementos, en este caso los servomotores, además se ha escrito en la capa superior de la placa el número de entrada PWM que es, para facilitar el trabajo.

De la misma manera hemos colocado, en la parte opuesta de la placa, las entradas analógicas, y también de la misma manera, todas ellas numeradas.

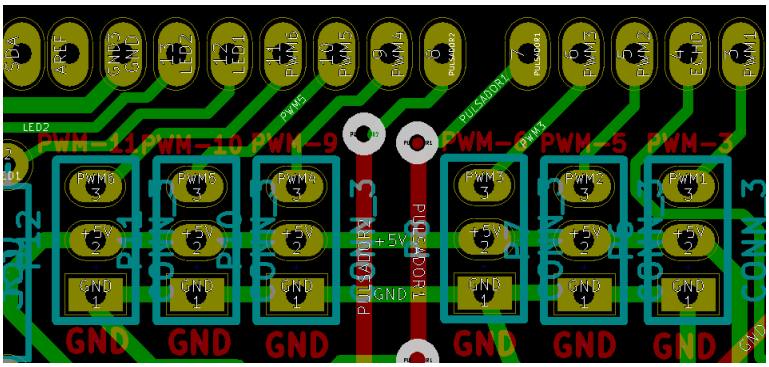


Ilustración 86: Entradas digitales para la conexión de nuestros cinco servomotores

De la misma manera hemos colocado, en la parte opuesta de la placa, las entradas analógicas, y también de la misma manera, todas ellas numeradas.

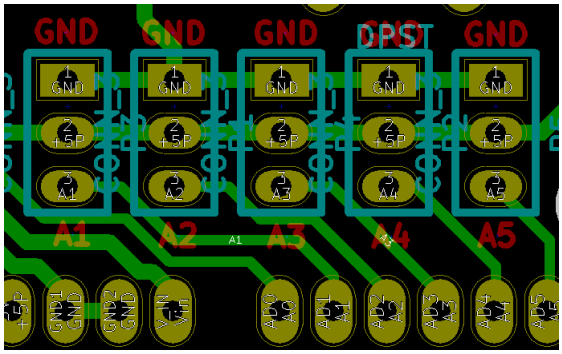


Ilustración 87: Entradas analógicas, al igual que las digitales, de tres pines

La colocación de la huella para el ultrasonido ha sido colocada estratégicamente, ya que estamos condicionados por dicho elemento, y es que debe estar siempre encarado al extremo de la placa para su correcto funcionamiento, aunque para este proyecto no se va a necesitar dicho elemento, se ha preferido colocar ya que se ha pensado en la universalidad de la placa para cualquier tipo de proyecto futuro.

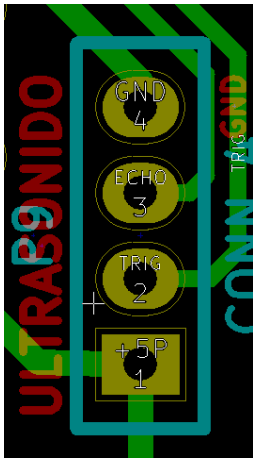


Ilustración 88: uella para ultrasonido

Al igual que con el ultrasonido, en este proyecto no se han utilizado las entradas para la comunicación serie, Rx (recepción de datos) y Tx (transmisión de datos), pero por el mismo motivo se ha colocado una entrada de cuatro pines para posible comunicación con, por ejemplo, bluethooth.

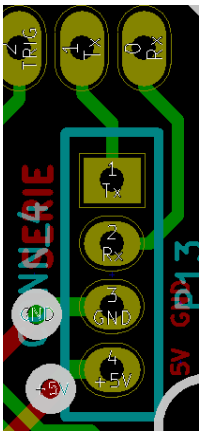


Ilustración 99: Huella de cuatro pines para comunicación serie

3.4.5. Visualización 3D de PCB

A continuación, se muestran imágenes en 3D de la capa A y la capa B de la placa electrónica extraída de KiCad.

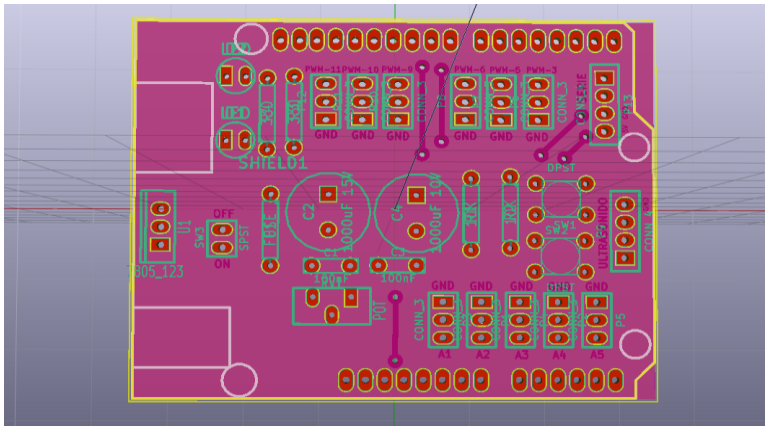


Ilustración 90: Capa superior

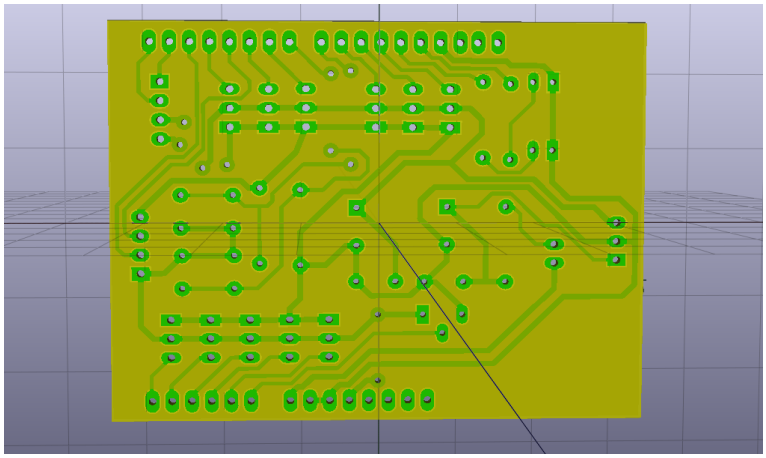


Ilustración 914: Capa inferior

3.4.6. Imágenes de la placa

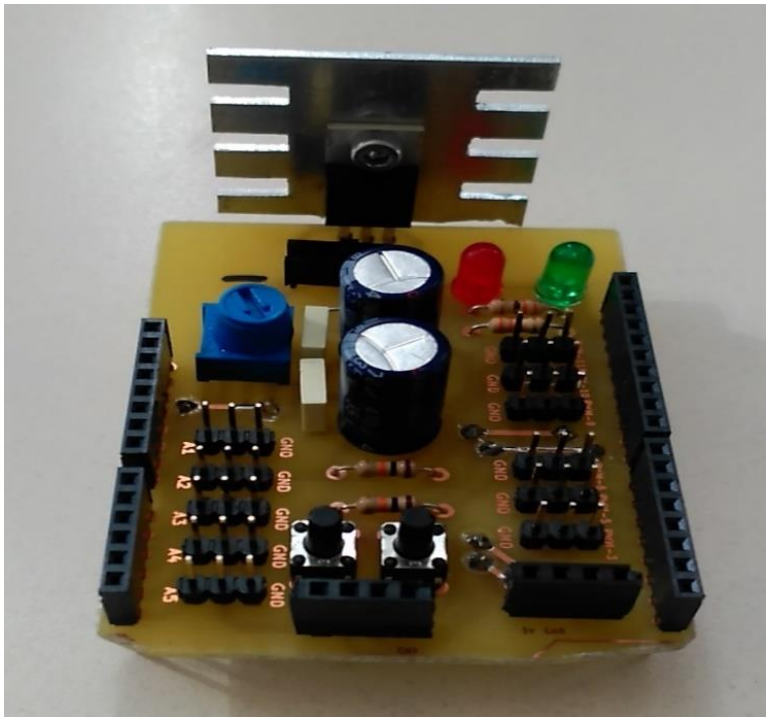


Ilustración 92: Placa terminada

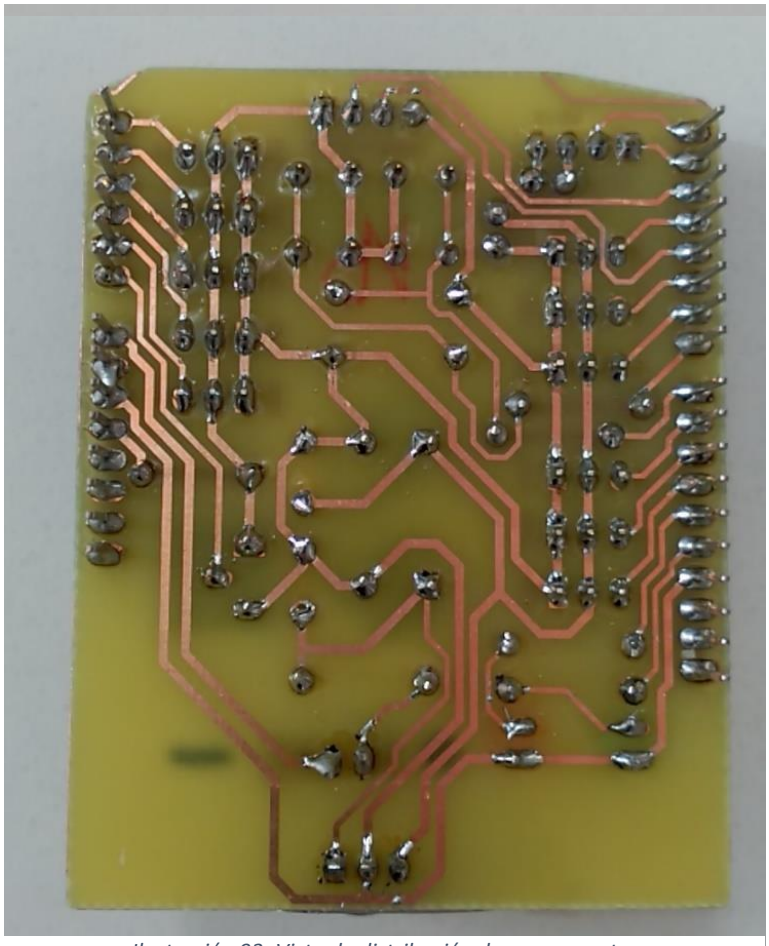


Ilustración 93: Vista de distribución de componentes

4. LABVIEW

4.1. Sobre LABVIEW (extraído de Wikipedia)

LabVIEW (acrónimo de Laboratory Virtual Instrument Engineering Workbench) es una plataforma y entorno de desarrollo para diseñar sistemas, con un lenguaje de programación visual gráfico. Recomendado para sistemas hardware y software de pruebas, control y diseño, simulado o real y embebido, pues acelera la productividad. El lenguaje que usa se llama lenguaje G, donde la G simboliza que es lenguaje Gráfico.

Este programa fue creado por National Instrumets (1976) para funcionar sobre máquinas MAC, salió al mercado por primera vez en 1986. Ahora está disponible para las plataformas Windows, UNIX, MAC y GNU/Linux. La penúltima versión es la 2013, con la increíble demostración de poderse usar simultáneamente para el diseño del firmware de un instrumento RF de última generación, a la programación de alto nivel del mismo instrumento, todo ello con código abierto. Y posteriormente la versión 2014 disponible en versión demo para estudiantes y profesional, la versión demo se puede descargar directamente de la página National Instruments.

Los programas desarrollados con LabVIEW se llaman Instrumentos Virtuales, o VIs, y su origen provenía del control de instrumentos, aunque hoy en día se ha expandido ampliamente no sólo al control de todo tipo de electrónica (Instrumentación electrónica) sino también a su programación embebida, comunicaciones, matemáticas, etc. Un lema tradicional de LabVIEW es: *"La potencia está en el Software"*, que con la aparición de los sistemas multinúcleo se ha hecho aún más potente. Entre sus objetivos están el reducir el tiempo de desarrollo de aplicaciones de todo tipo (no sólo en ámbitos de Pruebas, Control y Diseño) y el permitir la entrada a la informática a profesionales de cualquier otro campo. LabVIEW consigue combinarse con todo tipo de software y hardware, tanto del propio fabricante -tarjetas de adquisición de datos, PAC, Visión, instrumentos y otro Hardware- como de otros fabricantes.

Su principal característica es la facilidad de uso, válido para programadores profesionales como para personas con pocos conocimientos en programación pueden hacer programas relativamente complejos, imposibles para ellos de hacer con lenguajes tradicionales. También es muy rápido hacer programas con LabVIEW y cualquier programador, por experimentado que sea, puede beneficiarse de él. Los programas en LabView son

llamados instrumentos virtuales (VIs) Para los amantes de lo complejo, con LabVIEW pueden crearse programas de miles de VIs (equivalente a millones de páginas de código texto) para aplicaciones complejas, programas de automatizaciones de decenas de miles de puntos de entradas/salidas, proyectos para combinar nuevos VIs con VIs ya creados, etc. Incluso existen buenas prácticas de programación para optimizar el rendimiento y la calidad de la programación. El labView 7.0 introduce un nuevo tipo de subVI llamado VIs Expreso (Express VIS). Estos son VIs interactivos que tienen una configuración de caja de diálogo que permite al usuario personalizar la funcionalidad del VI Expreso. El VIs estándar son VIs modulares y personalizables mediante cableado y funciones que son elementos fundamentales de operación de LabView.

Como se ha dicho es una herramienta gráfica de programación, esto significa que los programas no se escriben, sino que se dibujan, facilitando su comprensión. Al tener ya pre-diseñados una gran cantidad de bloques, se le facilita al usuario la creación del proyecto, con lo cual en vez de estar una gran cantidad de tiempo en programar un dispositivo/bloque, se le permite invertir mucho menos tiempo y dedicarse un poco más en la interfaz gráfica y la interacción con el usuario final. Cada VI consta de dos partes diferenciadas:

- *Panel Frontal*: El *Panel Frontal* es la interfaz con el usuario, la utilizamos para interactuar con el usuario cuando el programa se está ejecutando. Los usuarios podrán observar los datos del programa actualizados en tiempo real (como van fluyendo los datos, un ejemplo sería una calculadora, donde tú le pones las entradas, y te pone el resultado en la salida). En esta interfaz se definen los *controles* (los usamos como entradas, pueden ser botones, marcadores etc..) e *indicadores* (los usamos como salidas, pueden ser gráficas …).
- *Diagrama de Bloques*: es el programa propiamente dicho, donde se define su funcionalidad, aquí se colocan íconos que realizan una determinada función y se interconectan (el código que controla el programa --. Suele haber una tercera parte *cono/conector* que son los medios utilizados para conectar un VI con otros Vis.

En el panel frontal, encontraremos todo tipos de controles o indicadores, donde cada uno de estos elementos tiene asignado en el diagrama de bloques una terminal, es decir el usuario podrá diseñar un proyecto en el panel frontal con controles e indicadores, donde

estos elementos serán las entradas y salidas que interactuarán con la terminal del VI. Podemos observar en el diagrama de bloques, todos los valores de los controles e idicadores, como van fluyendo entre ellos cuando se está ejecutando un programa VI.

4.2. Programa

Se ha realizado un código de programación en LabVIEW para controlar el robot SCARA que consiste en la programación de posiciones a las que se les llamará cuando se necesiten. Es decir, previamente se guardarán todas las posiciones del SCARA que creamos que nos van a ser útiles para nuestra tarea, dichas posiciones se harán moviendo cada servomotor por separado hasta llegar a la posición deseada. Cuando tengamos el brazo colocado en la posición que queremos simplemente tendremos que asignarle un número y un comentario que nos ayudará a recordar que posición es cada número.

Cuando hayamos guardado todas las posiciones que deseemos simplemente tenemos que guardarlas en un archivo para poder recurrir a ellas tras haber apagado el programa. Posteriormente, se ha creado un programa con el que puedes llamar a todas las posiciones que quieras, incluso asignar una pausa entre una y otra si lo deseas, para poder ejecutarlo y que reproduzca las posiciones que le hemos introducido.

También se ha creído conveniente asignar un botón del programa llamado “Inicio” con el que mandar al robot SCARA a una posición inicial previamente definida.

El control de los servomotores se hará moviendo una cantidad deseada de grados cada uno de ellos, aquí se ha definido un valor de 10º por defecto cada vez que arrancamos el programa. El número de grados que de desee mover cada servomotor se podrá cambiar en cualquier momento desde el programa.

A continuación, se mostrará el programa realizado con LabVIEW, que consta de varios subprogramas.

4.2.1. COM

Lo primero que debemos hacer es crear un programa para la detección automática del puerto COM, puesto que sin esto deberíamos de seleccionar el puerto a mano, por lo que

nos hace la tarea mucho más cómoda. Dicha función existe en LabVIEW 2015 pero no en su versión 2016, por lo que tuvimos que rescatarla.

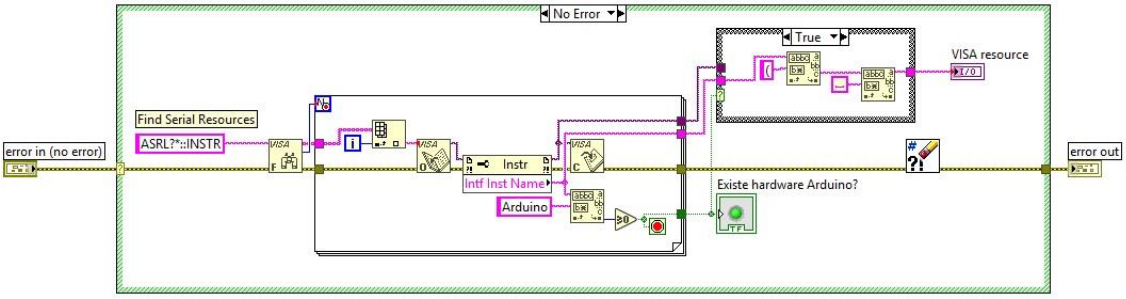


Ilustración 94: Diagrama de bloques del programa "COM"

4.2.2. Array unidades

Nos permite pasar cada uno de nuestros cinco servomotores (0-4) de grados (º) a microsegundos (µs). Esto es necesario porque la función VI de LINX con la que controlamos los servomotores trabaja en µs.

Ilustración 95: Función de LINUX para el control de servomotores

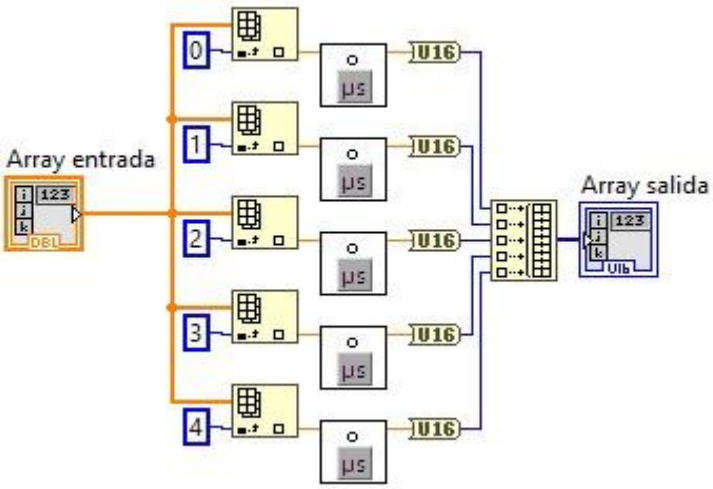


Ilustración 96: Diagrama de bloques del programa "Array unidades"

4.2.3. Buscar punto

Como explicamos en la introducción sobre el programa, a la hora de guardar una posición de nuestros lo necesitamos hacer con un número, por lo que esta función nos dirá si el número que deseamos utilizar para guardar nuestras coordenadas ya existe.

Dicho esto, esta función sirve para mostrar de manera gráfica al usuario las coordenadas ya guardadas en el programa

Si el número que queremos asignar resulta que ya existe, el programa nos lanzará directamente las coordenadas de dicho punto, avisándonos de que ese punto ya se encuentra asignado.

De igual manera, nos sirve para recordar qué coordenadas habíamos asignado para un punto en concreto de nuestro brazo.

A continuación, vemos una imagen del diagrama de bloques realizado para dicha función.

Vemos que se inicia con una Array de único punto que va a un Cluster donde está dicho punto más todas las partes del brazo y un posible comentario.

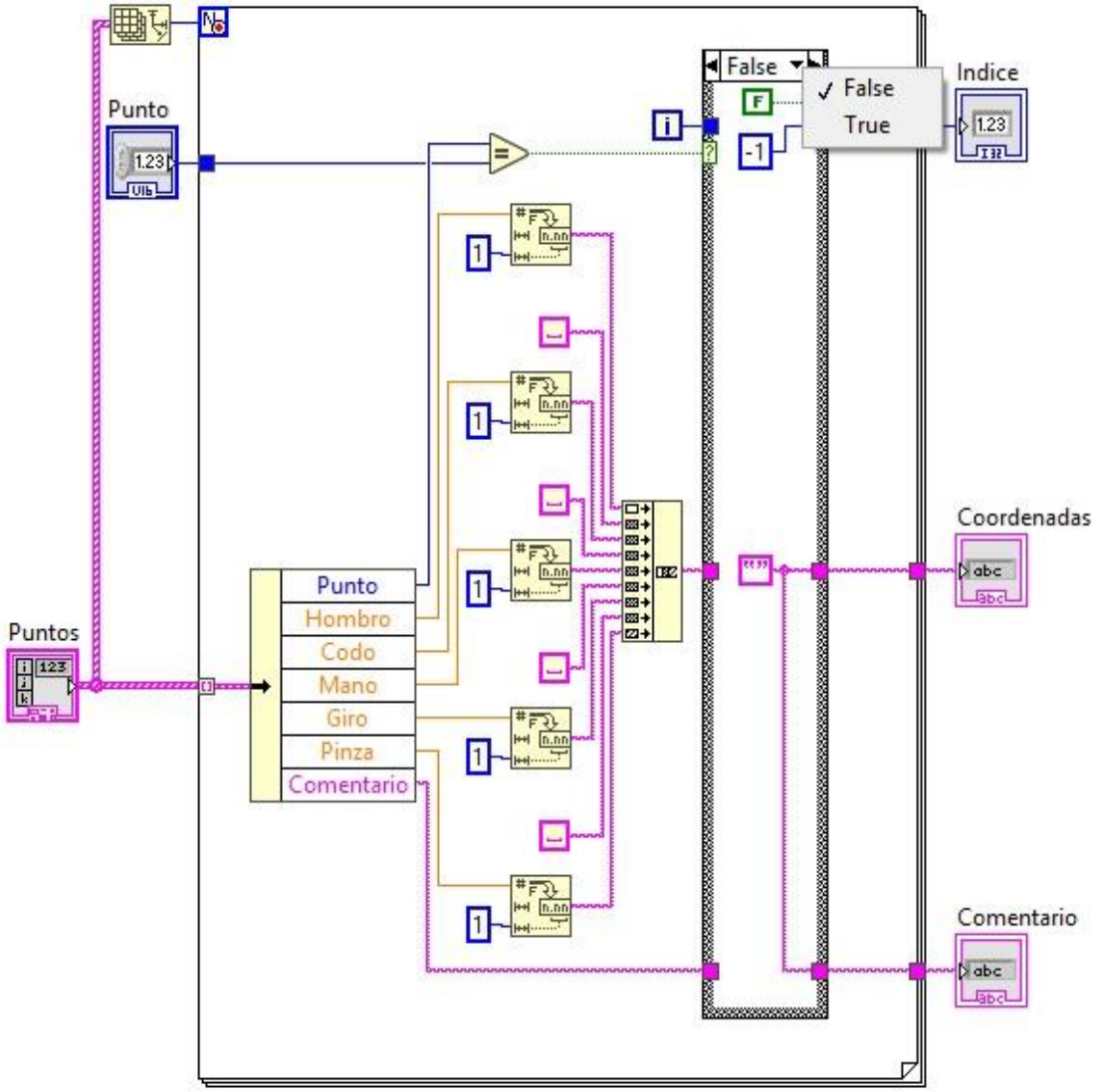


Ilustración 97: Diagrama de bloques del programa "Buscar Punto"

4.2.4. Guardar

El programa guarda las posiciones en memoria para poder llamarlo posteriormente. Guarda cada punto con una precisión de tres decimales, aunque como vimos en "Buscar punto", los muestra únicamente con uno.

También nos permite crear fichero, editar y abrir. Además, nos permite hacerlo en modo "sólo lectura" ayudando a proteger nuestro archivo.

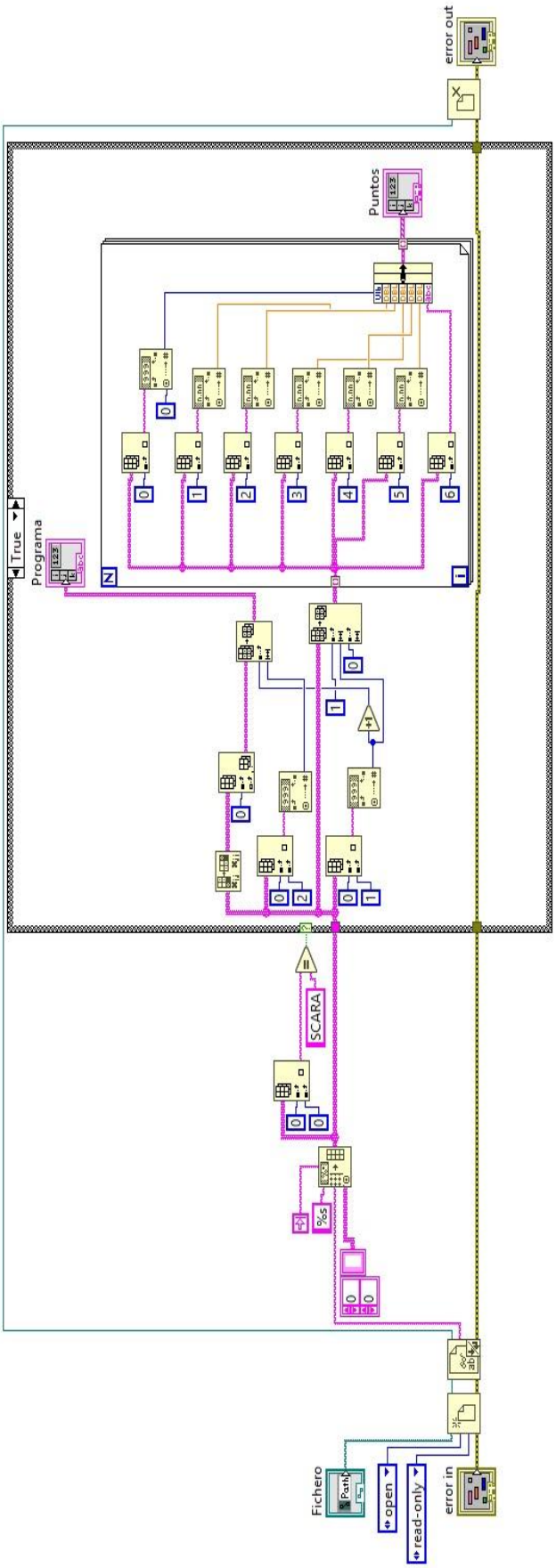


Ilustración 98: Diagrama de bloque del programa "Guardar"

4.2.5. Leer

Este programa nos permite leer lo anteriormente guardado, es decir, nos permite abrir documento, dando acceso a la carpeta dónde guardemos nuestros proyectos, dando también la opción de editar dichos archivos.

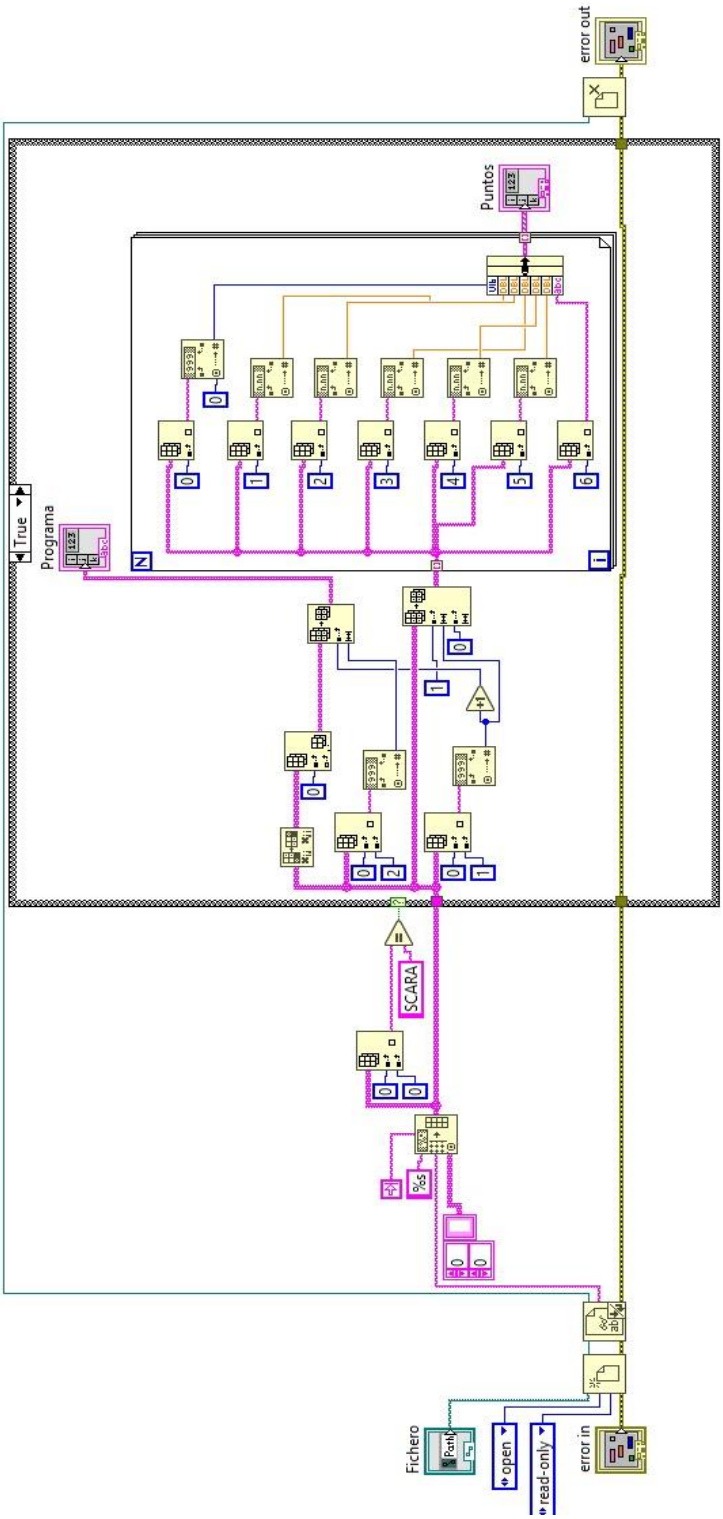


Ilustración 99: Diagrama de bloque del programa "Leer"

4.2.6. Obtiene punto

El programa "Obtiene punto" consiste en poder constatar si un punto ya existe, por lo que nos ayuda a no sobrescribir. También nos será muy útil a la hora de poder recordar cual es la posición de puntos anteriormente guardados.

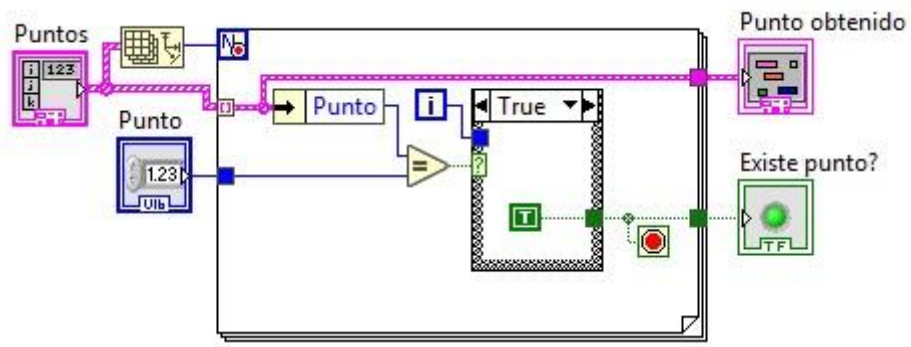


Ilustración 100: Diagrama de bloques del programa "Obtiene punto"

4.2.7. Programa Array-Cadena

Este programa transforma el vector de posiciones que tenemos en memoria en una cadena de caracteres con el fin de que posteriormente sea guardado en disco. Ya que solo podemos guardar en forma de caracteres.

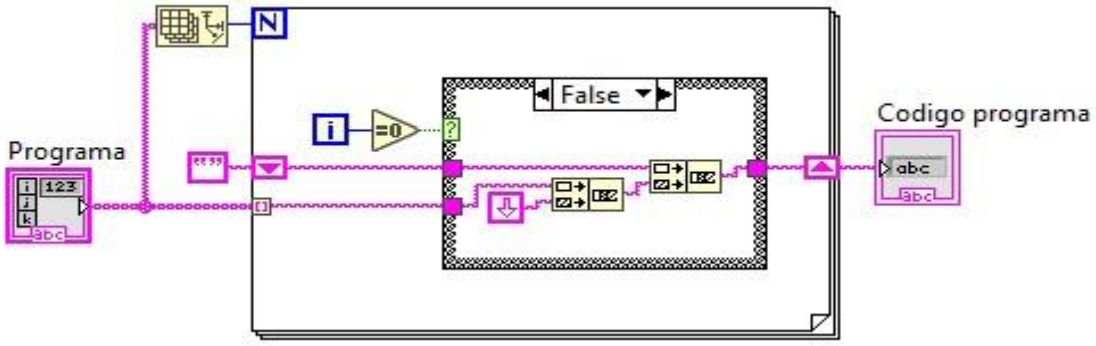


Ilustración 101: Diagrama de bloques del programa "Array-cadena"

4.2.8. Programa Cadena-Array

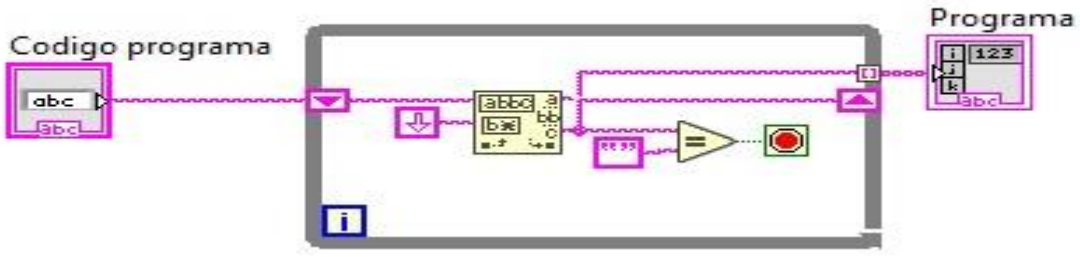


Ilustración 102: Diagrama de bloques del programa "Cadena-Array"

4.2.9. Programa final

El panel frontal diseñado con LabVIEW para control de robot SCARA está formado por dos partes, una interface para controlar el robot y un panel para ejecutar programas de movimiento guardados con el otro panel.

El panel "Interface con el robot" está compuesto por las siguientes aplicaciones;

En la parte superior encontramos todas las opciones posibles para que el usuario pueda trabajar con un proyecto, es decir, nuevo, abrir, guardar y guardar como. Además de la función "Inicio", que llevará el robot a la posición predeterminada de partida.

Ilustración 103: Interface para control del usuario

Después encontramos un panel de comandos que nos servirá para guardar momentáneamente todas las posiciones que deseemos, a estas posiciones deberemos asignarle un número y si también, si lo necesitamos, un pequeño texto orientativo para mayor facilidad a la hora de recordar. Todas estas posiciones podrán ser llamadas y utilizadas individualmente. Cuando cerremos el programa todas las posiciones y programas guardadas se perderán, siempre y cuando no hayan sido guardadas en un fichero.

Ilustración 104: Controles para guardar y mover individualmente coordenadas

La última sección de esta interface nos permite mover cada uno de los servomotores del robot SCARA tantos grados como queramos. Además, nos muestra las actuales coordenadas, que serás las que el programa guarde.

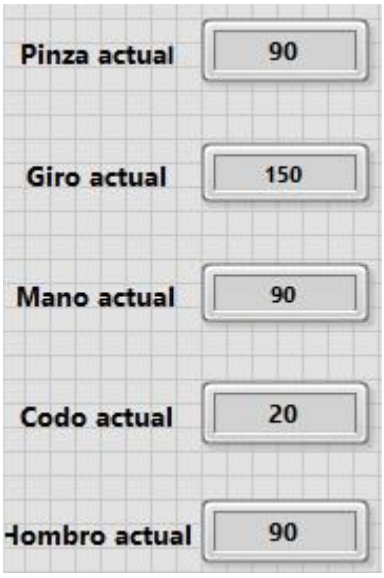


Ilustración 105: Coordenadas

Ilustración 106: Movimiento de servos más cantidad de grados a mover

La segunda parte de nuestro panel frontal nos permite ejecutar en un programa una secuencia de movimientos guardados anteriormente.

Como se ha comentado, a cada posición que se desee guardar se le deberá asignar un número. Este número será utilizado para llamar a cada posición.

Ilustración 107: Comando para solucionar el número que queramos

También podremos poner una pausa, en milisegundos, entre movimiento y movimiento. Además de haber añadido un botón para ir directo a la posición Inicio y por supuesto, uno para parar la comunicación.

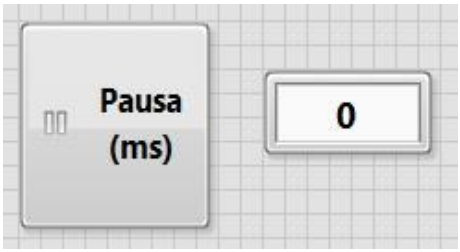


Ilustración 108: Comando para general una pausa en nuestro programa

A continuación, se copiará un ejemplo del archivo que genera el programa cuando guardas unas determinadas coordenadas junto con una secuencia de movimiento.

Ilustración 109: Programa ejemplo a ejecutar

```
SCARA   8       16
1       24,000  25,000  70,000  80,000  100,000 P1 arriba pinza abierta
2       24,000  25,000  10,000  80,000  100,000 P1 posición abajo pinza abierta
3       24,000  25,000  10,000  80,000  160,000 P1 posición abajo pinza cerrada
4       24,000  25,000  120,000 80,000  160,000 P1 posición arriba pinza cerrada
5       129,000 85,000  120,000 80,000  160,000 P1 posición final arriba pinza cerrada
6       129,000 85,000  0,000   80,000  160,000 P1 posición final abajo pinza cerrada
7       129,000 85,000  0,000   80,000  100,000 P1 posición final abajo pinza abierta
8       129,000 85,000  80,000  80,000  100,000 P1 posición final arriba pinza abierta
MOVER 1
PAUSA 1000
MOVER 2
PAUSA 1000
MOVER 3
PAUSA 1000
MOVER 4
PAUSA 1000
MOVER 5
PAUSA 1000
MOVER 6
PAUSA 1000
MOVER 7
PAUSA 1000
MOVER 8
INICIO
```

Ilustración 110: Formato del archivo a ejecutar

En primer lugar, observamos en nombre con el que guarda el programa "SCARA", el número de coordenadas "8" y el número movimientos, incluidas las pausas, "16".

A continuación, podemos ver el número de cada posición con sus respectivas coordenadas, tras esto, las 16 líneas de código que ejecutaremos, que como podemos observar, para llamar a la posición 1, por ejemplo, simplemente tenemos que seleccionar la posición 1 y apretar el botón "Mover".

Ilustración 111: Interface completa para control del programa

Nota: La imagen del diagrama de bloques sin desglosar se encuentra en Anexo I (página 77).

5. Conclusión

El proyecto ha resultado todo un reto, debido a que se han necesitado conocimientos en distintos ámbitos como el diseño en 3D, la mecánica para el piñón-cremallera, electrónica para la realización de la placa de potencia y programación para su correspondiente control. Además de softwares como *Solidworks*, *KiCad* y *LabVIEW*.
Como se comentó, se debieron realizar distintos prototipos hasta conseguir el definitivo, tanto en diseño como en electrónica. Por lo que el proyecto ha servido para conseguir una gran experiencia práctica.

El principal escollo y conclusión que saco de este proyecto es el problema a la hora de alimentar un buen número de servomotores, cinco en este caso, ya que cuando nos disponemos a controlar todos los servos a la vez, la intensidad se convierte en un problema.

En definitiva, hemos comprobado que es posible realizar un robot SCARA, que además es capaz de levantar piezas de hasta un kilogramo de peso, con una impresora 3D y controlado con un Arduino.

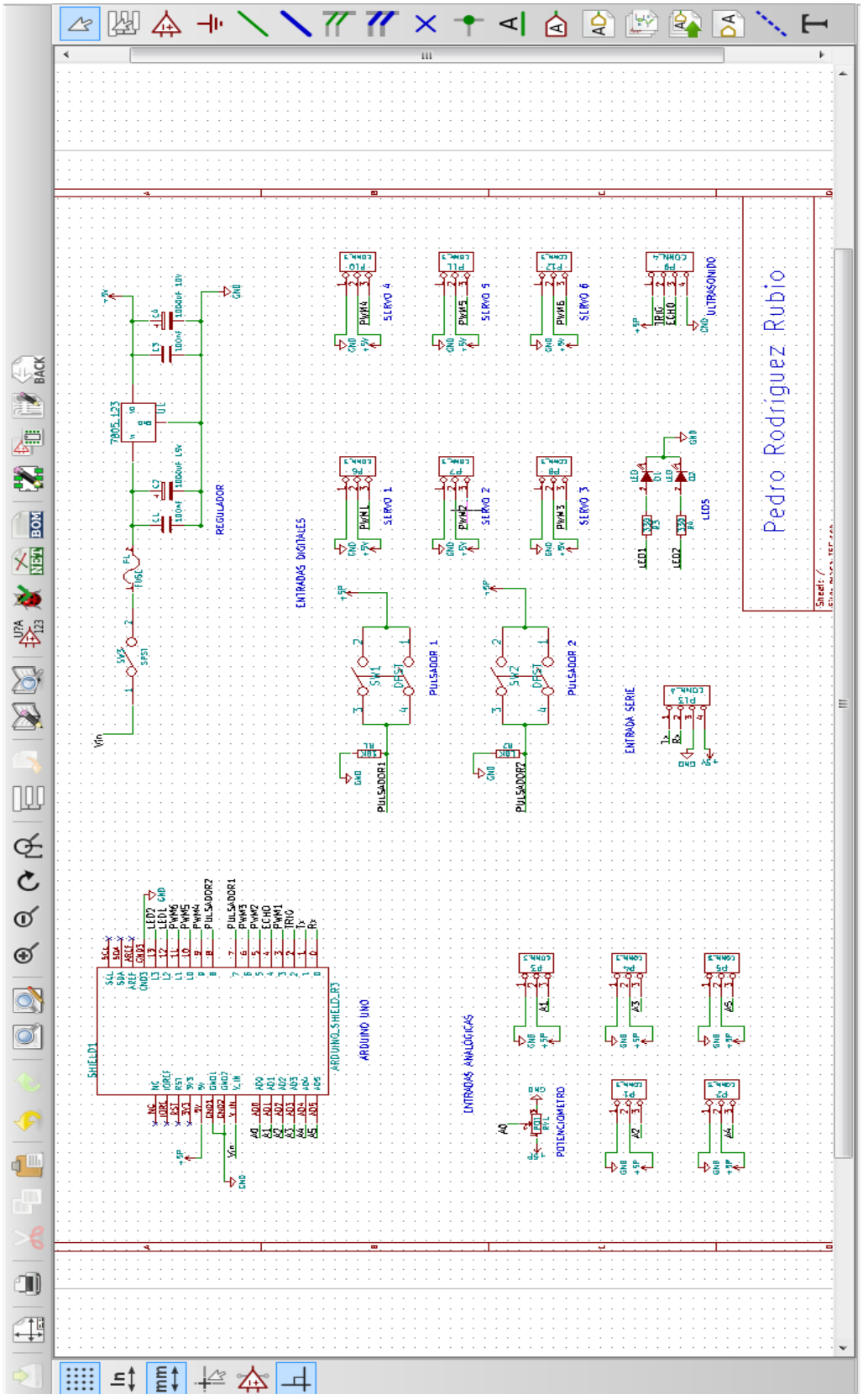
6. Referencias

6.1. Webgrafía

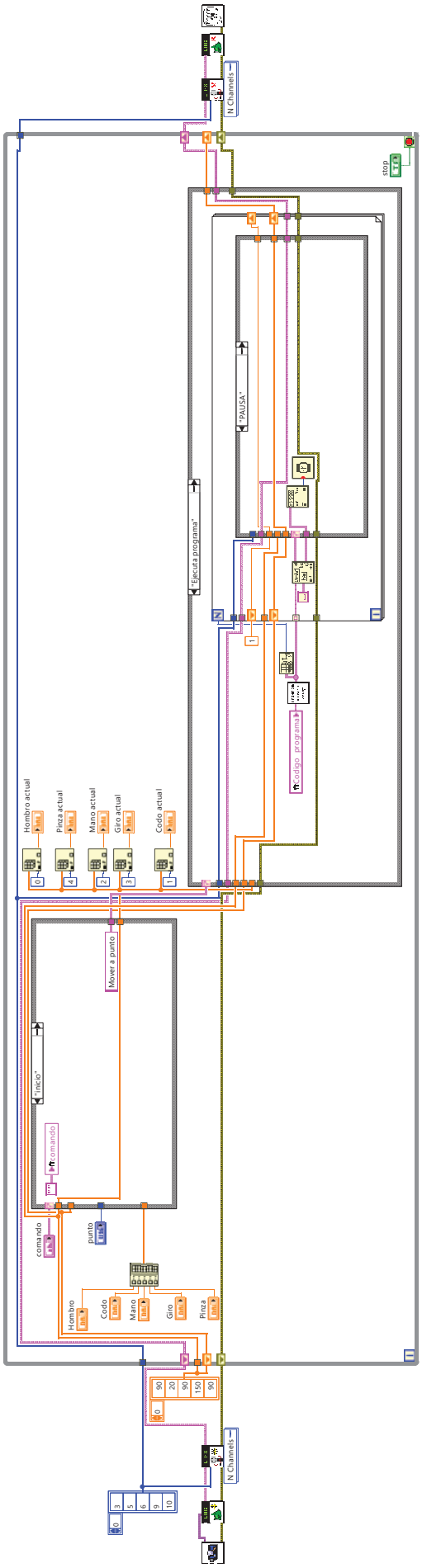
- *Acrilonitrilo butadieno estireno (ABS) extraído de:* https://es.wikipedia.org/wiki/Acrilonitrilo_butadieno_estireno
- Introducción sobre Arduino extraída de: https://es.wikipedia.org/wiki/Arduino
- Introducción sobre LabVIEW extraída de: https://es.wikipedia.org/wiki/LabVIEW

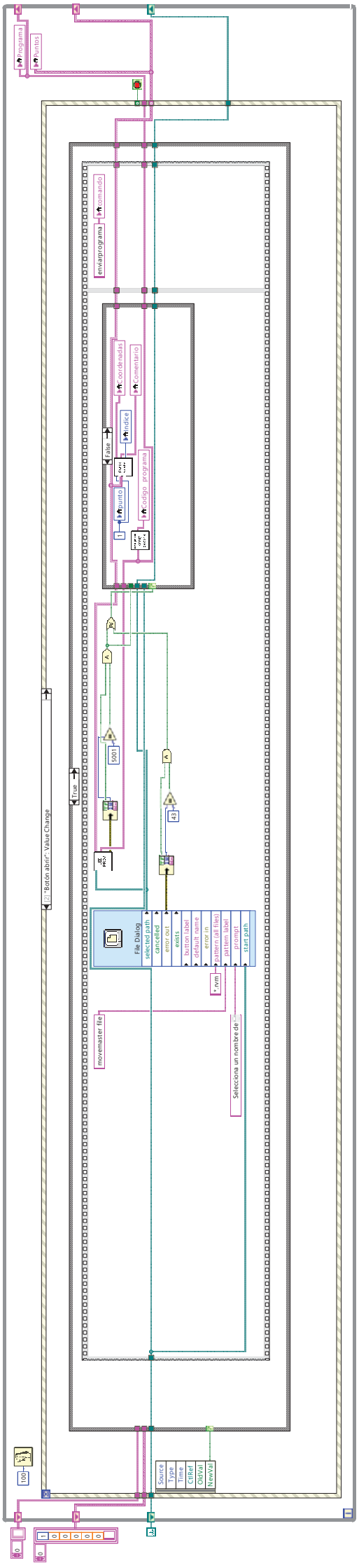
7. Anexos

7.1. ANEXO I



7.2. ANEXO II


N Channels
stop
"PAUSA"
"Ejecuta programa"
Codigo programa
Hombro actual
Pinza actual
Mano actual
Giro actual
Codo actual
"Inicio"
Mover a punto
comando
punto
Hombro
Codo
Mano
Giro
Pinza
90
20
90
150
90
N Channels
3
5
6
9
10


Programa
puntos
enviarprograma
comando
coordenadas
comentario
false
indice
punto
codigo programa
"Boton abrir" Value Change
True
5001
40
File Dialog
selected path
cancelled
error out
exists
button label
default name
error in
pattern (all files)
pattern label
prompt
start path
*.rvm
movemaster file
Selecciona un nombre de
100
Source
Type
Time
CtlRef
OldVal
NewVal

7.3 ANEXO III

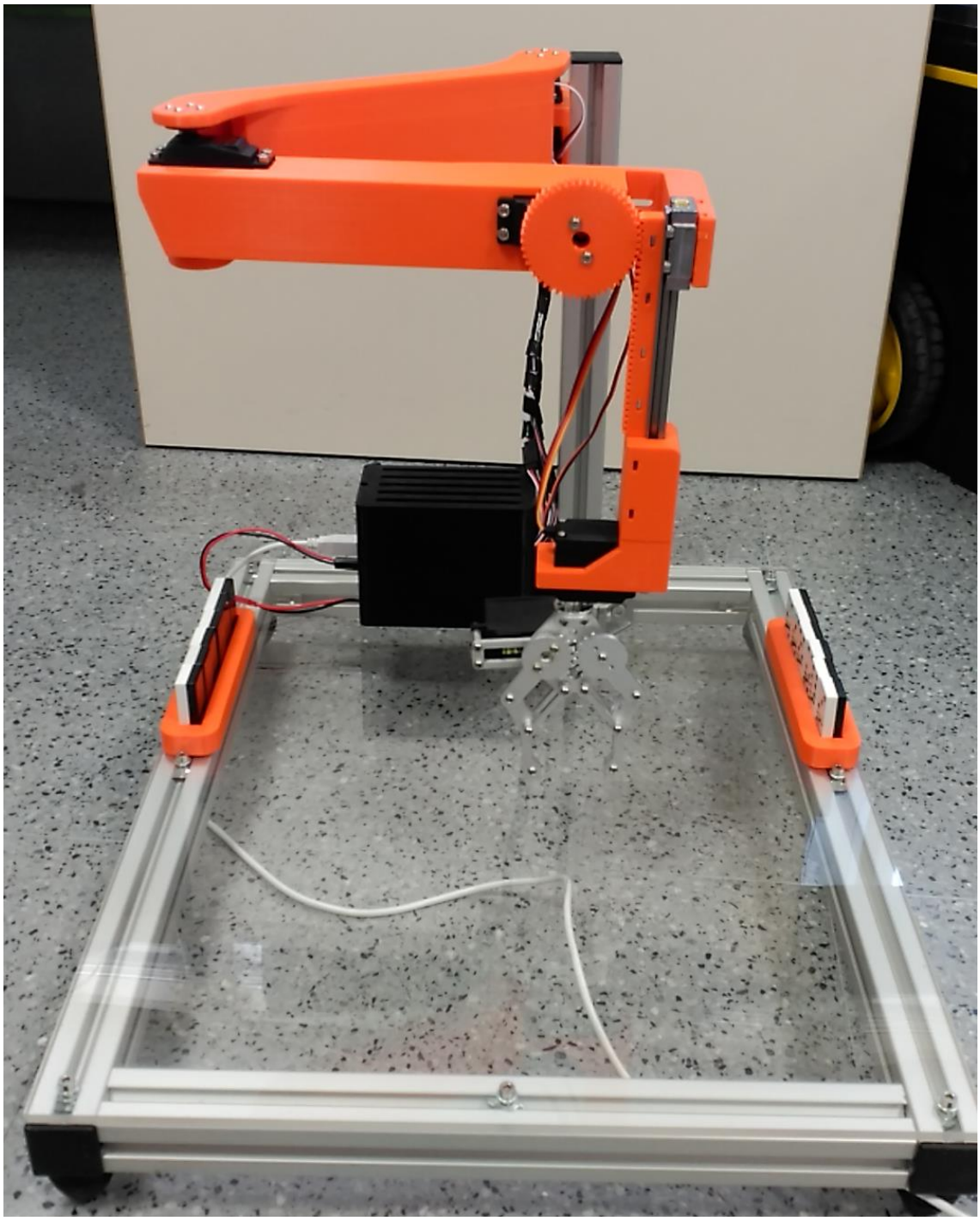


Ilustración 113: imagen real SCARA