Treball Fi de GRAU

**Grau en Enginyeria Electrònica Industrial i Automàtica**

**PROGRAMACIÓ DE ROBOT LEGO MINDSTORMS**



**Memòria i Annexos**

**Autor:** Pol Codinachs La Torre

**Director:** Nom del Director/a

**Co-Director**: Nom del Co-Director/a si n’hi ha

**Ponent:** Només si el Director/a no és professor/a de l’Escola

**Convocatòria:** Mes 20##

# Resum

Aquest projecte de final de carrera consisteix en el disseny i programació d’una impressora 3D a partir d’un kit de LEGO Mindstorms EV3 i una etapa controladora basada en una placa d’Arduino

La funció del robot, que ha estat programat en llenguatge C a partir del software ROBOTC, és imprimir en tres dimensions una sèrie de figures ja pre-dissenyades i seleccionables des de l’etapa controladora. A més a més, també és possible prendre el control del robot per tal de poder fabricar peces manualment sense la necessitat de ser dissenyades i llegides per el robot prèviament.

# Resumen

Escriure aquí el resum del TFG/TFM en castellà.

# Abstract

Escriure aquí el resum del TFG/TFM en anglès.

# Agraïments

Es poden incloure agraïments relatius a ajuts en la realització del treball i en la preparació del document. No és habitual agrair les contribucions com ara un control de rutina, un petit ajut o unes recomanacions de tipus general.

El reconeixement d’altres treballs emprats ha de fer-se en forma de referències. Els agraïments que fan referència a un text citat i a l’ús de taules i il·lustracions poden requerir el reconeixement de drets d’autor.

# Glossari

*Apartat opcional.* Quan un document conté signes, símbols, abreviatures, acrònims o termes que poden no ser compresos fàcilment i ràpidament pels possibles lectors haurien de definir-se en una o vàries llistes. L’existència d’aquestes llistes no justifica l’omissió d’una explicació sobre aquests elements quan apareixen per primera vegada en el text.

**Índex**

El següent índex es genera i s’actualitza automàticament amb l’opció “*actualizar campos*” que surt quan es prem el botó dret del ratolí a sobre d’ell. Feu que l’índex s’iniciï en una pàgina imparell.

[Resum i](#_Toc2855077)

[Resumen ii](#_Toc2855078)

[Abstract iii](#_Toc2855079)

[Agraïments v](#_Toc2855080)

[Glossari vii](#_Toc2855081)

[1. Prefaci 1](#_Toc2855082)

[1.2. Origen del treball 1](#_Toc2855083)

[1.3. Motivació 1](#_Toc2855084)

[1.4. Requeriments previs 1](#_Toc2855085)

[2. Introducció 3](#_Toc2855086)

[2.1. Objectius del treball 3](#_Toc2855087)

[2.2. Abast del treball 3](#_Toc2855088)

[3. LEGO® MINDSTORMS® EV3 5](#_Toc2855089)

[3.1. Introducció 5](#_Toc2855090)

[3.2. Història 5](#_Toc2855091)

[3.3. Contingut del kit 7](#_Toc2855092)

[4. Anàlisi de l’impacte ambiental [Ctrl + Shift + 1] 9](#_Toc2855093)

[Conclusions 11](#_Toc2855094)

[Pressupost i/o Anàlisi Econòmica 13](#_Toc2855095)

[Bibliografia 15](#_Toc2855096)

[5. Bibliografía 16](#_Toc2855097)

[Annex A 17](#_Toc2855098)

[A1. Text de l’apartat 1 de l’Annex 17](#_Toc2855099)

## Prefaci

*Capítol opcional.* Un prefaci pot considerar-se com una nota de presentació per definir l’estudi objecte de projecte, destacar-ne algun aspecte en particular, mostrar la seva relació amb treballs relacionats o traçar les circumstàncies històriques que han motivat el projecte. Alguns apartats poden ser:

## Origen del treball

## Motivació

## Requeriments previs

1. Introducció

La introducció ha d’establir breument l’abast i els objectius del treball que es descriuen en el document, la seva relació amb altres treballs i la metodologia emprada. No ha de repetir o parafrasejar el resum, ni donar detalls de la teoria, l’experimentació, el mètode o els resultats, ni anticipar les conclusions o les recomanacions. Pot incloure la informació del prefaci, si aquest no es presenta per separat. Alguns apartats poden ser:

## Objectius del treball

## Abast del treball

1. LEGO® MINDSTORMS® EV3

## Introducció

MINDSTORMS® és el kit d’experimentació i jocs basat en la robòtica educativa més potent actualment de la famosa empresa danesa de joguines LEGO®. El kit sencer consta d’una sèrie de peces, actuadors i sensors que poden ser fàcilment ensamblats i programats per qualsevol usuari gràcies al seu entorn de desenvolupament visual i guies de muntatge.

## Història

La història de LEGO Mindstorms pren sortida a finals del segle passat, concretament l’any 1986 moment en que la prestigiosa firma llança al mercat el primer producte capaç de ser controlat per un ordinador. Arrel d’aquest gran desenvolupament sorgeix un acord de col·laboració entre LEGO i el *Massachusetts Institute of Technology* amb l’objectiu de produir un “maó intel·ligent” el qual donarà vida a les creacions de LEGO a través de la programació a partir del computador.

Aquest producte va significar un gran avanç dins la societat tecnològica fet que va portar que la companyia LEGO arribés a un acord amb Dean Kamen, inventor de FIRST (*For Inspiration and Recognition of Science and Technology*) amb el propòsit d’impulsar la FIRST LEGO League, una competició de robòtica per alumnes de secundària on s’introdueix el kit LEGO Mindstorms. El primer Campionat del Món es va celebrar l’any 2005.

L’any 2013, coincidint amb el 15è aniversari de LEGO Mindstorms es va llançar al mercat el producte definitiu, precedit per el model NXT, el Mindstorms EV3 que avui en dia s’utilitza en centres educatius de robòtica i programació.

## Contingut del kit

El kit de LEGO® MINDSTORMS® compta amb diferents tipus de peces de muntatge, basades en un conjunt de sensors i actuadors que permeten interactuar el robot amb l’entorn i un *Brick* -maó intel·ligent- que actua com a cervell del conjunt.

#### Bloc intel·ligent EV3

Aquest bloc és l’element que serveix de centre de control. Utilitza el programa carregat en la memòria interna per poder posar en funcionament els motors i paral·lelament llegir la informació dels sensors. Per altra banda també realitza la funció de font d’alimentació per a tots aquests perifèrics que s’hi conecten.

Tal i com es pot apreciar en les següents imatges, *Figura 3.2 i Figura 3.3,* en la superfície d’aquest mòdul de control s’hi distingeixen els següents elements:

1. Pantalla: Mostra tot el que succeeix a l’interior del Bloc EV3 i permet utilitzar-ne la interfície. També permet afegir text i respostes numèriques o gràfiques a la programació o experiments.
2. Botons: El conjunt de botons de l’EV3 permeten explorar i navegar per l’interior de la interfície. També poden utilitzar-se com activadors de programació.
3. Ports d’entrada (1-4), *Figura 3.2*: Els ports d’entrada s’utilitzen per a connectar sensors al bloc EV3.
4. Ports de sortida (A-D), *Figura 3.3*: Els ports de sortida s’utilitzen per a connectar actuadors al bloc EV3.



Figura 0.2. Bloc EV3 i vista detall dels ports de sortida i entrada USB. (Font: <https://www.robotix.com>).

|  |  |
| --- | --- |
| **Sistema Operatiu** | LINUX |
| **Processador** | ARM9 de 300 MHZ |
| **Memòria Flash** | 16 MB |
| **Memòria RAM** | 64 MB |
| **Resolució de la pantalla** | 178 x 128 blanc i negre |
| **Comunicació** | USB 2.0 a equip amfitrió fins a 480 Mbit/s  USB 1.1 a amfitrió fins a 12 Mbit/s |
| **Tarjeta MicroSD** | Admet SDHC, versió 2.0, 32 GB màx. |
| **Ports d’entrada i sortida** | Connectors RJ12 |
| **Alimentació** | 6 bateries AA |

Taula X.X. Especificacions tècniques *Brick EV3.* (Font: User Guide LEGO MINDSTORMS EV3 11 AII ES).

#### Sensors i actuadors

El sensor de color és un sensor digital capaç de detectar tant la tonalitat com la intensitat de llum que incideix per un petit orifici que es troba en la seva superfície. Pot ser utiltzat en tres modes: Mode color, Mode intensitat de llum reflexada i Mode intensitat de llum ambiental.

En el primer Mode, el sensor és capaç de reconèixer fins a set colors: negre, blau, verd, groc, vermell, blanc i marró. En canvi, pel que fa els altres dos modes d’intensitat, el seu *modus operandi* es basa en utilitzar una escala de 0 (molt fosc) fins a 100 (molt lluminós) que serveix per medir la llum incident.

Un tret diferencial entre ambdós modes de funcionament és que el primer medeix la intensitat reflexada des de una pròpia làmpada i el segon de la llum ambiental.



Figura X.X. Sensor de color (Font: <http://shop.lego.com>).

Dins del conjunt Mindstorms també hi trobem dos sensors de contacte, *Touch Sensor,* un sensor analògic capaç de detectar el moment en el que és pressionat o quan deixa de ser-ho. D’aquesta manera, pot ser programat en tres condicions: pressionat, llançat o en contacte i dóna un tipus de resposta booleana verdader o fals.



**Figura X.X.** Sensor de color (Font: <http://shop.lego.com>)**.**

També trobem l’anomenat sensor d’infrarrojos que consisteix en un receptor digital capaç de detectar llum infrarroja reflexada per objectes massissos i alhora detecta senyals de llum infrarroja enviades per la balisa infrarroja remota. De la mateixa manera que el sensor de color, pot ser utilitzat en tres modes diferents que són: Mode de proximitat, Mode de balisa i Mode remot.

El primer d’ells utilitza les ones de llum reflexades per un objecte i permet calcular la distància que hi ha entre el sensor i l’objecte. Aquest, informa de la distància a la que es troba l’objecte utilitzant una escala de 0 (molt proper) a 100 (molt lluny).

MOTORS

Pel que fa els motors, aquests juguen un paper molt important dins del kit Mindstorms ja que són els únics actuadors. N’hi ha de dos tipus, *Large* i *Medium*. El conjunt n’incorpora dos del primer i solament un del segon. Així doncs, la seva principal diferència com el seu nom indica és la potència que són capaços d’entregar cadascun d’ells i la manera que ho fan.

En primer lloc, el *Medium Motor*, es tracta d’un actuador ideal per aplicacions de menor càrrega que requereixen alta velocitat i molta precisió. El motor utilitza una retroalimentació per obtenir un control precís en un grau de precisió i disposa també d’un sensor de rotació integrat.

Seguidament, el *Large Motor*, és un motor que com el seu nom indica té unes dimesions superiors al motor anterior. Es tracta d’un actuador de gran potència (en comparació l’anterior) que també utilitza la retroalimentació per oferir una precisió d’un grau.

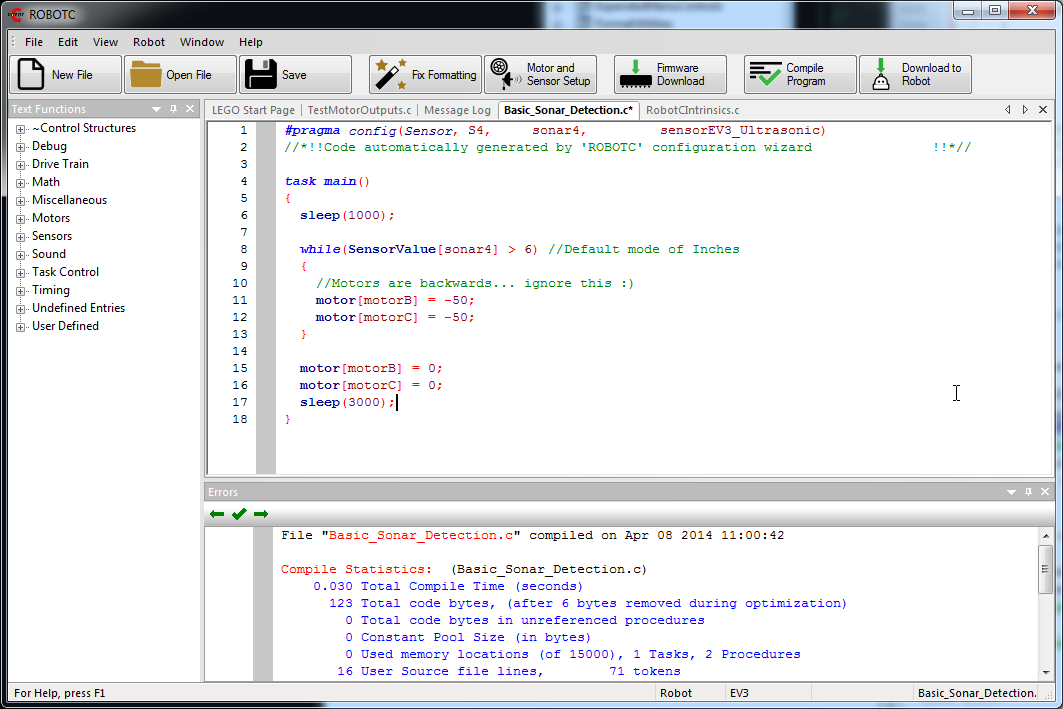


**Figura X.X.** Motors del kit Mindstorms. D’esquerra a dreta: *Medium Motor, Large Motor* (Font: <http://shop.lego.com>)**.**

## Programació en ROBOTC

El programari ROBOTC, és un potent software de programació basat en C des de Windows que permet escriure i depurar programes. Va ser creat per la *Robotics Academy de la Carnegie Mellon Univsersity de Pittsbur (EUA)* iajuda als estudiants a aprendre la programació en C que s’utilitza en aplicacions d’educació avançada i professionals.

ROBOTC presenta moltes avantatges pel que fa la programació de robots, doncs disposa d’una GUI (Interfície Gràfica d’Usuari) avançada que facilita les tasques de programació i unes eines de depuració de programa que permeten a l’usuari veure en temps real l’estat dels motors i sensors així com també el valor que prenen les variables locals i globals utilitzades segons l’execució del programa.



**Figura X.X.** Entorn de programació ROBOTC. (Font: <http://robotc.net>)**.**

La mateixa empresa desenvolupadora del software, RoboMatter, disposa d’una pàgina *help* sobre ROBOTC molt completa on s’hi poden trobar guies d’ús tant per la instal·lació com per la programació. Tanmateix, en la mateixa pàgina, també existeix una llibreria d’ordres per la programació de robots Mindstorms EV3 on s’explica cada ordre detalladament amb els corresponents paràmetres i funció que realitza.

1. Arduino

Arduino és una plataforma electrònica de codi obert (*open-source*) caracteritzada per la seva facilitat en l’ús tant a nivell de hardware com de software.

Es tracta d’una placa capaç de llegir senyals d’entrada (digitals o analògiques) i convertir-les en sortides, ja sigui activant un motor, un led o qualsevol altre actuador preparat per funcionar en aquest entorn.

Un dels avantatges d’utilitzar aquesta placa electrònica és que existeix un entorn de programació, desenvolupat per la mateixa empresa, que permet processar i programar les entrades i sortides físiques a través d’un llenguatge de programació pròpi.

## Característiques tècniques

El model més nou i dels més utilitzats per la comunitat d’usuaris d’Arduino és la placa UNO, disposa de 14 pins que poden configurar-se com entrada o sortida i als que es pot conectar qualsevol element capaç de transmetre o rebre senyals digitals de 0 i 5 V.

També disposa d’entrades i sortides analògiques i amb aquestes primeres som capaços d’obtenir dades de sensors en forma de variacions continuades d’un voltatge. Les sortides, solen utilitzar-se per enviar senyals de control en forma de senyals PWM.

Cadascún dels 14 pins digitals que poden ser utilitzats com entrada o sortida i funcionen amb una tensió de 5 V a nivell alt i 0 V a nivell baix i la capacitat de subministrar un corrent fins a 40 mA.

Els pins de la placa també poden ser utilitzats per a les comunicacions amb elements externs, ja sigui una altra placa, un computador... etc. A través dels ports Rx i Tx, l’Arduino pot enviar i rebre informació a través de transmissions serie de senyals TTL.

La placa d’Arduino pot ser alimentada de dues formes possibles. La primera d’elles és a partir del port USB tipus B i connectar-la al PC. La segona consisteix en utilitzar un petit transformador o, per exemple una pila de 9 V i utilitzar el connector de 2,1 mm que proporciona la placa.

## Parts de la placa

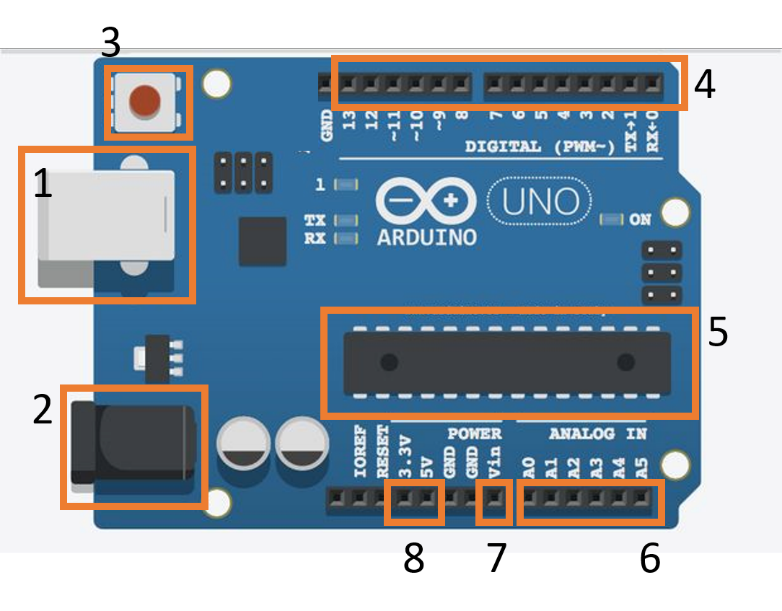


Figura X.X. Placa Arduino UNO (Font: Pròpia).

1. Port USB: Conector USB del tipus B que s’utilitza per conectar la placa a l’ordinador i així poder carregar-i el programa desenvolupat amb l’Arduino IDE. També pot ser utilitzat com a font d’alimentació a 5 V.
2. Jack d’alimentacio DC: Adaptador de AC a DC que pot alimentar la placa entre 7 i 12 V.
3. Botó de reset: Polsant aqest botó es conectarà temporalment el pin de reset a terra i es reiniciarà qualsevol programa que estigui carregat a la placa.
4. Regulador de tensió: La funció del regulador de tensió és, com el seu pròpi nom indica, controlar la tensió per tal de que no s’espatlli la placa.
5. Leds TX/RX: Aquests dos LEDS s’activen quan l’Arduino està enviant (TX) o rebent (RX) dades.
6. Port Digital: Consta de 14 pins d’entrades/sortides digitals, 6 dels quals són sortides PWM i 2 per les comunicacions sèrie Tx/Rx.
7. Processador: Atmel ATmega328P de 8 bits, 16MHz, 32KB de memòria Flash, 1KB de memòria EEPROM i 2KB de SRAM.
8. Port Analògic: Consta de 6 pins per entrades analògiques de 0 a 5 V i 10 bits ADC, anomenats bits de ressolució. La tensió que medeixen va de 0 a 5 V tot i que és possible canviar el seu rang utilitzant una funció amb el pin AREF.
9. Alimentació d’entrada: Pins per alimentar la placa a 9 V.
10. Alimentació de sortida: La placa proporciona dues sortides amb 5 V i 3,3 V respectivament però també pot utilitzar-se per alimentar un altre component amb una tensió regulada de 5 o 3 V.
11. Comunicació EV3 – Arduino (I2C)

El protocol I2C es tracta d’una comunicació sèrie que defineix les dades i les connexions físiques per transferir bits entre dos dispositius digitals.

Un port I2C qualsevol està format principalment per dos cables de comunicació que són SDA (*Serial Data)*, per la comunicació de dades, i SCL *(Serial Clock)*, que proporciona un senyal de rellotge per sincronitzar tots els dispositius connectats.

Tal i com es pot veure en la *Figura X.X,* la xarxa I2C sempre la componen un *Master* (Mestre) i un o més d’un *Slaves* (Esclaus). El mestre és l’encarregat de controlar la xarxa, doncs aquest és el que determina el senyal de rellotge per a tots els dispositius i fins que ell no dóna l’ordre no poden iniciar-se les comunicacions.

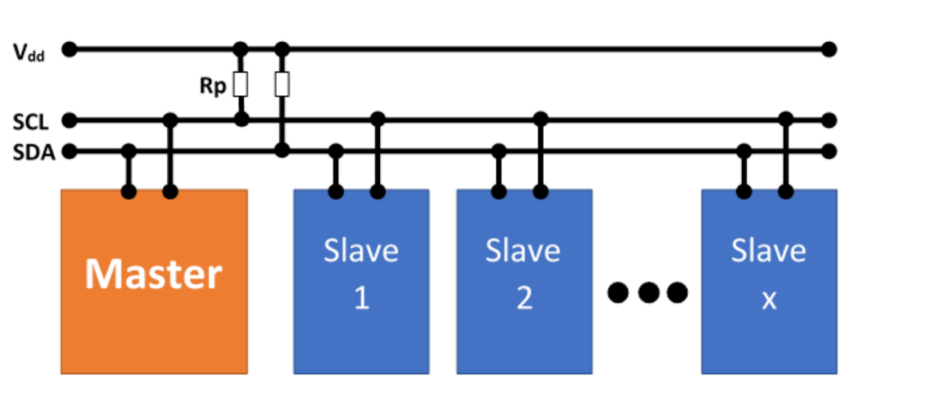


Figura X.X. Exemple de xarxa de comunicacions I2C. (Font: DexterIndustries).

En la *Figura X.X.* observem que els misstges que s’envien contenen una estructura determinada i estàndar per a tots.

En primer lloc, trobem un bit de *Start* que indica la condició d’inici del missatge fent canviar d’estat, de nivell alt a nivell baix, la línia SDA abans que ho faci el canal SCL.

Just després d’aquest bit, el missatge conté un espai reservat d’una mida de 7/10 bits amb la finalitat d’escriure-hi la direcció física de l’esclau al que es vol transmetre el missatge i seguidament dos bits més. El primer d’ells per determinar si el que es vol fer és enviar un missatge o bé sol·licitar-ne un. I el segon, anomenat ACK/NACK, que s’utilitza per confirmar si la informació ha estat llegida satisfactòriament pel receptor o no.

La informació que es vol transmetre al receptor s’emmagatzema en un *Data Frame de 8 bits*, en un espai del missatge que es troba just després del bit ACK/NACK.

Per últim, un bit de *Stop* és el que indica el final del missatge.

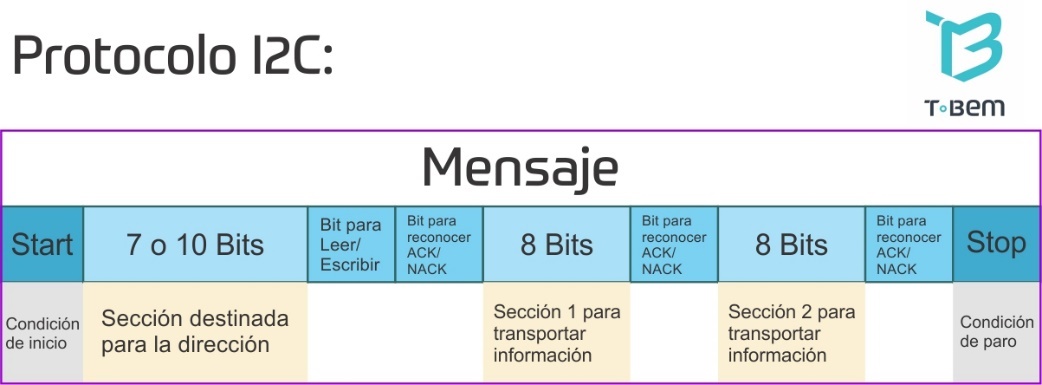


Figura X.X. Estructura d’un missatge I2C. (Font: TeslaBem).

Qualsevol missatge que es vulgui transmetre a la xarxa a través del bus I2C segueix una seqüència que és la següent:

El Mestre envia la condició de *Start* a tots els esclaus connectats a la línia SDA i en canvia el voltatge de nivell alt a nivell baix.

Tot seguit es procedeix al direccionament d’esclaus on, primer de tot, el mestre envia la direcció de l’esclau amb el que vol comunicar-se, conseqüentment tots els esclaus reben aquesta direcció i la comparen amb la seva pròpia. Si la direcció coincideix, l’Esclau de seguida envia un bit ACK amb nivell de voltatge baix al Mestre. Si pel motiu que fos la direcció no coincideix amb cap dels esclaus, el mestre no rep cap resposta es torna al nivell alt de voltatge a la línia SDA.

Un cop es té l’esclau identificat, el Mestre envia els *Frames* d’informació i seguidament el receptor enviarà un bit d’ACK al Mestre per notificar-li que la informació s’ha rebut satisfactoriament. Per concloure la transmissió d’informació, el Mestre envia a l’Esclau la condició de parada (stop) amb un nivell alt a la línia SDA.

En definitiva el protocol de comunicació I2C resulta molt funcional ja que tant sols utilitza dos cables per a la transmissió de dades i és capaç de suportar múltiples Mestres-Esclaus amb les configuracions adients en cada cas i a més, disposa del bit ACK/NACK que informa de que el missatge s’ha rebut correctament fet que resulta molt còmode a l’hora de realitzar comprobacions.

Enginyeria del disseny

## Disseny de l’estructura

El robot del present projecte utilitza una estructura previament dissenyada per l’autor bazmarc (<https://www.instructables.com/member/bazmarc/>) amb unes petites modificacions per tal d’adaptar-ho als recursos disponibles a l’hora de realitzar el treball.

L’estructura del robot està formada principalment per dos mòduls, A i B, que proporcionen moviment lineal i bidireccional dins l’espai X i Y a la base on s’imprimeixen les peces en 3 dimensions. Aquesta base (*Figura X.X*) és una peça plana i quadrada de LEGO amb unes dimensions de 6 x 6 (unitats de LEGO) equivalents a 4,8 x 4,8 cm.

Foto base

Per altra banda, el tercer mòdul, anomenat C, també proporciona moviment lineal i bidireccional dins l’espai Z al llapis d’impressió 3D (*Figura X.X*).

Foto llapis

Primerament, el Mòdul A (*Figura X.X*) està compost principalment per un *Medium Motor* i un actuador lineal. El motor genera un moviment circular i el transmet, a través d’un eix, a l’actuador lineal que transforma la rotació en un moviment lineal del pistó intern que posseeix. Amb la base d’impressió subjectada al cap del pistó s’aconsegueix el desplaçament d’aquesta en l’eix X.

Plànol mòdul A

Seguidament, els elements principals que componen el Mòdul B (*Figura X.X*) són els mateixos que ho fan en l’anterior (*Medium Motor* i actuador lineal) amb la diferència que en aquest cas al cap del pistó hi ha fixat tot el Mòdul A. Així s’aconsegueix desplaçar tot el Mòdul A bidireccionalment en l’eix Y i conseqüentment la base d’impressió.

Plànol Mòdul B

Finalment, el Mòdul C (*Figura X.X*) és totalment diferent als dos anteriors ja que l’element principal d’aquest és un *Large Motor* que fa moure una estructura de subjecció del llapis d’impressió en tres dimensions a través de l’eix Z de coordenades cartesianes.

Plànol Mòdul C

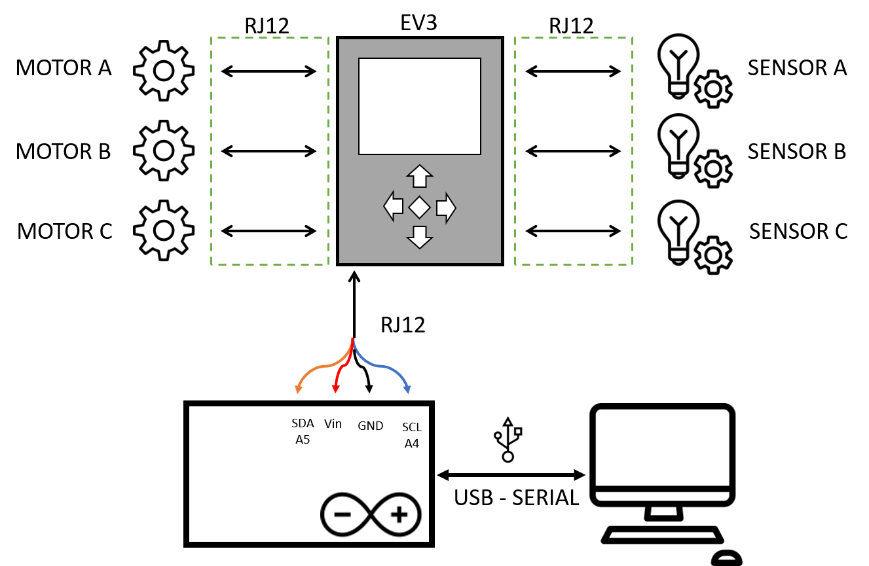
El motor d’aquest Mòdul genera un moviment circular a un conjunt d’engranatges de l’estructura que es troben en contacte amb la superfície dentada de dues torres de X cm d’alçada. L’acoplament entre els engranatges i les torres permet el moviment en l’eix Z de tot el Mòdul C.

## Disseny del sistema

La següent imatge (*Figura X.X.*) és una representació gràfica de les parts que componen tot el projecte i els enllaços entre elles.

Per una banda, i com a element més important trobem el maó intel·ligent EV3 que actúa com a cervell del sistema i en els seus ports d’entrades i sortides és on es connecten tots els motors i sensors del conjunt a través d’un cable RJ12.

Per altra banda tenim la placa d’Arduino que fa de passarel·la de comunicacions entre el maó intel·ligent i l’ordinador. Aquesta es connecta a l’EV3 a través d’una adaptació del cable RJ12 que permet la comunicació via I2C i al PC amb un cable USB per utilitzar la comunicació sèrie. D’aquesta manera s’estableix l’enllaç i transmissió de dades entre el robot i el software de control del computador.



## Figura X.X. Diagrama de blocs del sistema dissenyat. (Font: Pròpia).

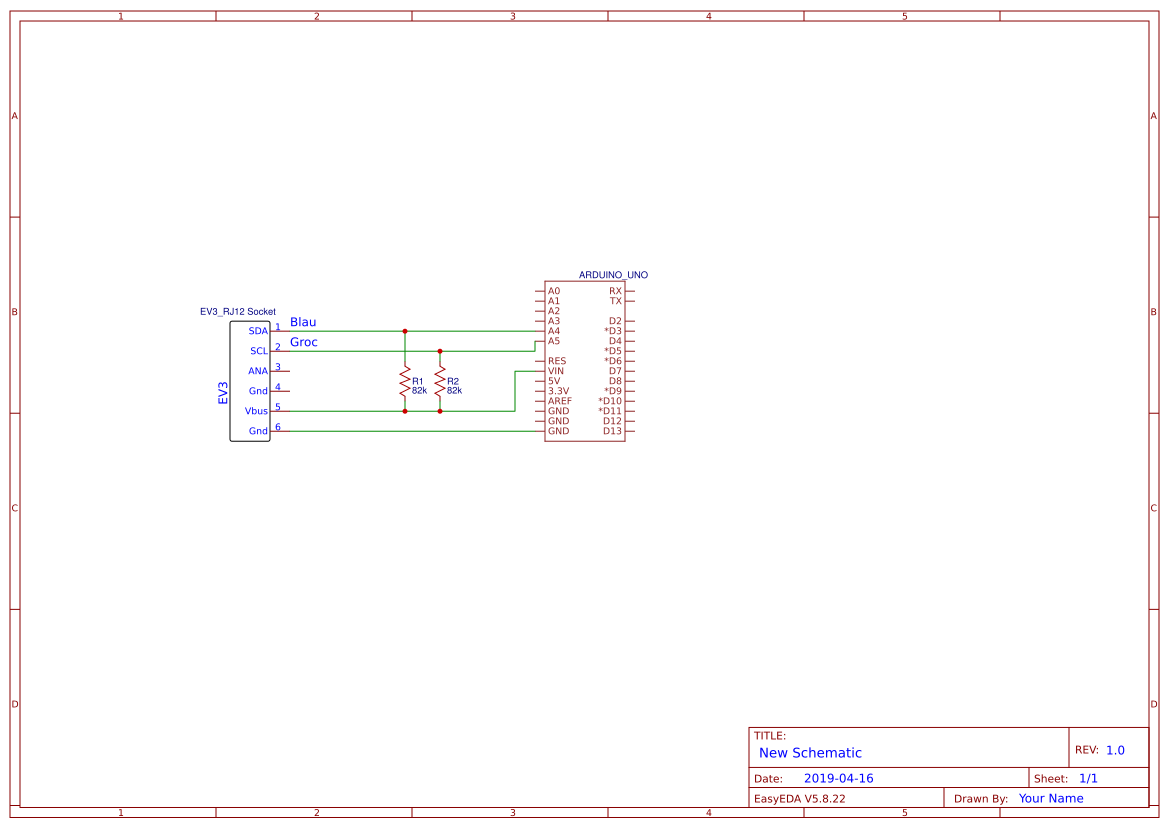
Enginyeria del detall

## Comunicació EV3 – Arduino

Per tal de comunicar aquests dos elements satisfactòriament s’utilitza del cable RJ12, *Figura X.X*, que serveix per connectar el maó intel·ligent amb els perfifèrics del kit Mindstorms (motors, sensors, etc...). La composició interna d’un RJ12 està formada per sis cables més petits: SDA, SCL, ANA, GND, Vbus, GND tal i com s’observa en la *Figura X.X*. Cadascú d’ells realitza una funció específica i en aquest projecte nomes han estat utilitzats els de dades *SDA i SCL* i els d’alimentació *Vbus i GND.*

Foto RJ12

Per un costat, la solució adoptada a nivell de hardware per efectuar la comunicació ha estat realitzar el muntatge i soldadura d’un connector femella RJ12 adaptat a Mindstorms, dues resistències de 82 k i X pins en una placa de topos tal i com es mostra en l’esquema electrònic de la següent *Figura X.X*.



**Figura X.X**. Circuit de l’adaptació del cable Mindstorms (Font: Pròpia).

Foto placa soldada

Per altra banda, també cal dissenyar una solució a nivell de software. El primer que cal fer en el ROBOTC és definir en quin port de l’EV3 es troba el cable que es troba connectat a la placa soldada i conseqüentment a l’Arduino i definir també, l’adreça de l’Arduino. Tal i com es mostra a continuació:

#define ARDUINO\_PORT S1

#define ARDUINO\_ADRESS 0x08

Abans de seguir amb la configuració del programa per la comunicació, és necessari definir prèviament les funcions que ROBOTC utilitza per tal d’entendre el seu funcionament:

* **sendI2CMsg(Port on es troba l’Arduino, Missatge a enviar, Mida de la reposta):** Aquesta sentència, tal i com el seu nom indica, s’utilitza per a enviar missatges des de l’EV3 a través del protocol I2C.
* **readI2CReply(Port on es troba l’Arduino, Variable on guardar el missatge, Mida del missatge a rebre):** Igual que l’ordre anterior, a través d’aquesta es pot llegir la resposta enviada per l’Arduino.

Seguidament es creen dues variables que serviran per emmagatzemar els missatges enviats i rebuts.

byte mensajeI2C[22];

char respuestaI2C[20];

A partir d’aquí, es defineix una funció parametritzada *msg\_i2c* que rebrà la direcció de l’Arduino, la mida del missatge a enviar, la mida de la resposta que s’espera i els cinc *bytes* que es volen enviar.

int msg\_i2c(byte dir\_arduino, int tam\_mensaje, int tam\_respuesta, ubyte byte0, ubyte byte1, long byte2, ubyte byte3, ubyte byte4)

{

//Es prepara el byte respuestaI2C posant-lo a zero i copiant-li la mida que ja s'havia definit.

    memset(respuestaI2C, 0, sizeof(respuestaI2C));

    tam\_mensaje = tam\_mensaje + 3;

//Es defineixen cadascún dels bytes que el programa enviarà.

    mensajeI2C[0] = tam\_mensaje;

    mensajeI2C[1] = dir\_arduino;

    mensajeI2C[2] = byte0;

    mensajeI2C[3] = byte1;

    mensajeI2C[4] = byte2;

    mensajeI2C[5] = byte3;

   mensajeI2C[6] = byte4;

//S'executa l'ordre d'enviar el missatge, amb els paràmetres del port on es troba l'Arduino, el missatge i la mida que s'espera de la resposta.

    sendI2CMsg(ARDUINO\_PORT, &mensajeI2C[0], tam\_respuesta);

    wait1Msec(20);

//S'executa l'ordre de llegir la resposta de l'Arduino amb els mateixos paràmetres que l'ordre anterior.

    readI2CReply(ARDUINO\_PORT, &respuestaI2C[0], tam\_respuesta);

//És necessari llegir el primer byte

    int x = respuestaI2C[0];

    wait1Msec(35);

    return x;

}

A partir d’aquí, enviar un missatge a l’Arduino i obtenir-ne resposta resulta tant senzill com cridar la funció *msg\_i2c* amb els paràmetres corresponents.

Per altra banda, l’Arduino també necessita unes línies de codi explícites per tal de poder rebre missatges i enviar la resposta via I2C.

El primer que cal fer és declarar la llibreria *Wire* que conté les directives per tal de poder establir la comunicació.

#include<Wire.h>

Llavors, en la funció de *setup* es defineixen quines funciones s’executaran quan l’Arduino rebi un missatge o necessiti enviar-ne la resposta.

void setup()

{

Wire.begin(0x04); // Determinar l'adreça de l'esclau

Wire.onRequest(requestEvent); // Enviar la resposta

Wire.onReceive(receiveI2C); // Rebre informació

Serial.begin(9600);

}

I finalment es defineixen les següents funcions.

void receiveI2C(int bytesIn)

{

read\_byte = bytesIn;

byte\_count = 0;

while(1 < Wire.available())

{

read\_byte = Wire.read();

instruction[byte\_count] = read\_byte;

//Cada byte llegit es guarda en una posició de la variable instruction

byte\_count++;

}

Un petit detall que cal tenir en compte és que si ens fixem en l’adreça d’esclau definida al programa del robot (ROBOTC) i en el programa de l’Arduino, es pot veure com no concorda. La primera pren un valor de 0x08h (1000b) i la segona de 0x04h (0100b).

Aquest canvi és fruit del protocol I2C que el que fa és moure cap a l’esquerra un bit la direcció de l’esclau, d’aquesta manera, realitzant el moviment i afegint un zero a la dreta, passa de 0100b a 1000b.

Falta la funció per enviar I2C

## Programació en ROBOTC

Tal i com s’ha anat mencionant al llarg d’aquest treball, l’objectiu principal consisteix en programar en C un robot LEGO Mindstorms que s’ha decidit que sigui per mitjà del software ROBOTC.

#### Definició i ús dels motors i sensors

És necessari declarar tots els motors i sensors que s’utilitzen en el projecte dins del menú *Robot --> Motors and Sensors Setup* i el programa automàticament generarà una línia de codi per cada element declarat amb les característiques proposades.

En el nostre cas, només s’utilitzen 1 *Large Motor*, 2 *Medium Motor* i 3 *Touch Sensor.*  Així es veu en les següents línies:

#pragma config(Sensor, S1, TIR, sensorI2CCustom)

#pragma config(Sensor, S2, Touch1, sensorEV3\_Touch)

#pragma config(Sensor, S3, Touch2, sensorEV3\_Touch)

#pragma config(**Motor**, motorA, MotorB, tmotorEV3\_Medium, PIDControl, encoder)

#pragma config(**Motor**, motorB, MotorB, tmotorEV3\_Medium, PIDControl, encoder)

#pragma config(**Motor**, motorC, MotorC, tmotorEV3\_Large, PIDControl, encoder)

El software utilitzat facilita una serie d’ordres ja establertes tant pels motors com pels sensors les quals permeten llegir o escriure’n informació segons la directriu utilitzada.

Pel que fa els motors, s’utilitzen les següents ordres:

* moveMotorTarget(nMotorIndex, nPosition, nSpeed): Defineix una distància al motor des de la seva posició relativa a una velocitat determinada.
* resetMotorEncoder(nMotorIndex): Escriu un zero en el valor de l’encoder del motor.
* setMotorTarget(nMotorIndex, nPosition, nSpeed): Defineix una distància al motor des de la seva posició absoluta a una velocitat determinada.
* stopAllMotors(): Ordre per aturar tots els motors connectats al robot.
* stopMotor(motorPort): Ordre per aturar el motor indicat en el paràmetre de la funció.
* motor[motorX] = YY: Posa en marxa el motor X a la velocitat YY.

Per altra banda, pels sensors que s’utilitzen en aquest projecte n’hi ha prou amb llegir si estan en contacte o no amb l’ordre SensorValue[Port del sensor] i retornarà un valor booleà segons l’estat en que es trobi.

#### Algoritmes

Esquema dels algoritmes

## Interfície de control del robot

Un dels principals objectius d’aquest treball ha estat dissenyar i desenvolupar una interfície de control per al robot que permetiaccionar cada motor individualment, seleccionar la figura a imprimir i també poder dissenyar noves figures a imprimir des de l’ordinador.

La solució triada per assolir aquest objectiu ha estat utilitzar el llenguatge de programació C# (C Sharp) i una tecnologia anomenada *Windows Forms* que forma part de Microsoft .NET Framework i proporciona accés a tots els elements que componen la interfície gràfica del sistema operatiu Windows. Amb aquesta eina es poden desenvolupar aplicacions d’escriptori molt completes gràficament que tenen la capacitat de manipular recursos locals del computador com ara fitxers i el Port Sèrie.

La interfície ha estat distribuïda en X formularis: Principal, Control Manual, Disseny, Building.

El primer d’ells s’ha establert com a *Parent* per tal fer de contenidor de la resta anomenats *Child,* i que sigui el que sempre estigui actiu. Tal i com s’observa en la següent imatge, *Figura X.X, el Parent* disposa d’una barra lateral a la part esquerra on s’allotgen tres botons per tal de poder accedir als respectius formularis *Child* que es mostraran a la part central del formulari. També una barra superior amb un menú selector per escollit el Port Sèrie on es troba allotjada la placa d’Arduino i establir-ne la comunicació.

Foto Form Principal

El funció del Port Sèrie de l’aplicació està definida en una classe pública dins del codi del formulari *Parent* i d’aquesta manera és accessible des de qualsevol formulari amb les següents instruccions:

Funció de definició del Port Sèrie

Principal.SerialPortClass.serialPort1.ReadLine(); - per llegir les dades que es reben.

Principal.SerialPortClass.serialPort1.Write(); - per escriure dades en el Port.

En segon lloc trobem el formulari anomenat Control Manual que ha estat definit com a *Child* i dissenyat de la següent manera (*Figura X.X*).

Foto Form Control Manual

La funció principal d’aquest formulari és, tal i com el seu nom indica, permetre a l’usuari de la interfície controlar els moviments de cada motor del robot individualment.

Constantment es llegeixen les dades rebudes per el Port Sèrie des de l’Arduino amb la següent estructura:

A,B,C,D

Es tracta d’una cadena de tipus *string* amb diferents caracters separats per una coma i la lectura es repeteix tres vegades, una per cada motor. La primera posició de la cadena pot prendre un valor entre 1 i 3, ambdós inclosos, i fa referència a cada motor del robot; A, B o C. Seguidament, en la segona posició es llegeix la velocitat a temps real en que s’està movent el motor que s’ha indicat a l’inici de la cadena. Aquesta velocitat ha de ser tractada matemàticament i restar-li el valor enter de 256. En tercer lloc es llegeix encoder. I per últim, la quarta posició de la cadena ens indica amb un 1 o un 0 si el *Touch Sensor* que actúa com a final de cursa està en contacte o no. Tota aquesta informació s’emmagatzema per separat en llistes i posteriorment es mostren per pantalla en la interfície.

Per altra banda, l’aplicació també té la capacitat d’enviar dades a través del Port Sèrie al robot passant per l’Arduino. Aquesta tasca es duu a terme a partir d’una funció parametritzada amb tres variables anomenada *enviarDades*. El primer paràmetre de la funció és del tipus enter i serveix per indicar el motor al que es vol fer referència; 1, 2 o 3. Seguidament, un altre paràmetre de tipus enter que espera un valor entre 1 i 4, ambdós inclosos. Per fer avançar o retrocedir el motor s’utilizen els valors 1 i 2. I el 4 per pausar el moviment. Com a últim paràmetre s’utilitza una variable del tipus double per determinar la velocitat amb que ha de funcionar el motor indicat anteriorment.

public void enviarDades(int motor, int moviment, double velocitat)

El tercer formulari de la intefície, anomenat Disseny (*Figura X.X*), s’utiltza únicament per a seleccionar quina de les figues ja programades es vol imprimir. Cada figura conté un número identificador que s’envia mitjançant la funció *enviarDades*.

public void enviarDades(int figura)

Foto formulari Disseny

El quart i últim formulari, anomenat Building (*Figura X.X*) permet a l’usuari de la interfície utilitzar la llibreria *Graphics de C#* per dibuixar figures sobre una quadrícula que simula la base d’impressió del robot. Aquest formulari no utilitza les comunicacions via Port Sèrie, doncs la seva funció és crear un fitxer de text i escriure-hi les directrius que serveixen al robot per imitar les línies dibuixades.

Foto Formulari Building

Per al disseny de les figures s’utilitzen dos mètodes de la llibreria *Graphics*, anomenats *DrawLine* i *DrawArc,* que com el seu nom indica, permeten dibuixar línies i arcs en un formulari de *Windows Forms*.

La definició del primer mètode és una funció parametrizada amb el punt d’orígen, el punt final i l’estil gràfic de la línia.

DrawLine(penName, pointA, pointB)

Aquests paràmetres de la funció necessiten ser declarats abans d’utilitzar el mètode. En primer lloc, cal definir l’estil de la línia especificant-ne el color i el tamany de la següent manera:

Pen penName = new Pen(Color.color, width);

I seguidament fer el mateix amb les coordenades cartesianes de cada punt.

PointF pointA = new PointF(pointA(X), pointA(Y));

Per últim, el segon mètode, *DrawArc,*  ve donat per set paràmetres:

* Pen: Color i amplitud de l’arc.
* X: Coordenada X de la cantonada superior-esquerra del rectangle que defineix l’elipse.
* Y: Coordenada Y de la cantonada superior-esquerra del rectangle que defineix l’elipse.
* Width: Amplada del rectangle que defineix l’elipse.
* Height: Alçada del rectangle que defineix l’elipse.
* startAngle: Angle en graus mesurat en sentit horari des de l’eix X fins al punt d’inici de l’arc.
* sweepAngle: Angle en graus mesurat en sentit horari des de l’angle d’inici fins al punt final de l’arc.

DrawArc(Pen, X, Y, Width, Height, startAngle, sweepAngle);

Start Angle: 0º

Height

Width

(X,Y)

Sweep Angle

*Figura X.X:*

**Algoritme**

El primer pas a seguir per dissenyar una nova figura és concretar les coordenades X i Y del punt inicial a partir d’on el robot començarà la impressió. Les coordenades d’aquest punt seran introduides en la primera posició de dues llistes del tipus *double* i s’incrementarà un contador anomenat *contadorPunts* de valor inicial igual a zero utilitzat com a índex de la llista.

List<double> puntX = new List<double>();

List<double> puntY = new List<double>();

**Línies simples**

Amb el punt de partida ja definit, cal introduïr la distància de la línia que es vol dibuixar en el corresponent *Text Box* i polsar un dels botons: Amunt, Avall, Esquerra o Dreta. Com s’ha explicat anteriorment, el mètode *DrawLine* necessita les coordenades X i Y del punt d’orígen i del punt final. En aquest cas, el programa agafa per al punt d’orígen les coordenades que es troben en la posició *contadorPunts*-1 de les llistes *puntX i puntY* i calcula el punt final en funció del botó polsat:

* Amunt: La coordenada X del punt final es manté igual a la del punt d’orígen i es calcula la Y mitjançant la següent operació: .
* Avall: La coordenada X del punt final es manté igual a la del punt d’orígen i es calcula la Y mitjançant la següent operació: .
* Esquerra: La coordenada Y del punt final es manté igual a la del punt d’orígen i es calcula la X mitjançant la següent operació: .
* Dreta: La coordenada Y del punt final es manté igual a la del punt d’orígen i es calcula la X mitjançant la següent operació: .

Un cop calculades les coordenades del punt final, aquestes s’introdueixen a les llistes *puntX* i *puntY* en la posició equivalent al valor *contadorPunts.* Finalment, s’executa el mètode *DrawLine* i s’incrementa el contador.

**Línies diagonals**

En el cas de voler dibuixar una línia amb un cert angle d’inclinació α respecte els eixos de coordenades, cal introduïr-ne la distància de X i Y als respectius *Text Box* indicant el sentit amb el signe.

α

Y

X

Figura X.X.:

El programa, igual que en el cas anterior, utilitzarà les coordenades de la posició *contadorPunts-1* de les llistes *puntX i puntY* per determinar el punt d’orígen i calcularà les coordenades del punt final de la linia de la següent manera:

Tot seguit insereix les coordenades calculades a les llistes i incrementa el contador.

**Arcs i circumferències**

Per dibuixar arcs i circumferències el programa utilitza el mètode *DrawArc*. Que tal i com ha estat explicat anteriorment, requereix set paràmetres. L’algoritme defineix les variables alçada (*height*) i amplada (*width*) del quadrilàter que conté la circumferència, amb el valor del radi introduït en el *TextBox.* Llavors calcula les coordenades X i Y del vèrtex superior esquerre del quadrat a partir de les que es troben en la posició *contadorPunts – 1* de les llistes *puntX* i *puntY.* Aquests càlculs varien en funció de l’angle incial escollit i les mides del rectangle:

|  |  |
| --- | --- |
| **Angle incial** | **Equació** |
| 0 º | x = puntX[contadorPunts - 1] - width; |
| y = puntY[contadorPunts - 1] - height / 2; |
| 90 º | x = puntX[contadorPunts - 1] - width / 2; |
| y = puntY[contadorPunts - 1] - height; |
| 180 º | x = puntX[contadorPunts - 1]; |
| y = puntY[contadorPunts - 1] - height / 2; |
| 270 º | x = puntX[contadorPunts - 1] - width / 2; |
| y = puntY[contadorPunts - 1]; |

Taula X.X

Els càlculs de les coordenades X i Y

**Export**

Com s’ha mencionat anteriorment, la funció principal d’aquest formulari és traduir la figura dibuixada a moviments del robot i exportar-ho en un fitxer de text. Per això cal introduir un nom dins del *TextBox* name i seguidament escollir la ruta on es guardarà l’arxiu. Per acabar, s’especifiquen el nombre de capes que es volen imprimir i fill i es fa click en el botó *Send.*

Aquest algoritme utilitza el mètode *System.IO.StreamWriter* que permet escriure *strings* en un fitxer que es trobi en una ruta prèviament especificada. Primer de tot,

# Bibliografía

La bibliografia és la llista de documents consultats durant la preparació del TFG i s’ha d’incloure al final del cos del treball. S’ha de fer referència a aquesta llista en els llocs adequats en el text ([citacions dins el text](http://publica.upc.edu/ca/estil/bibliografies))

És molt recomanable anar confeccionant la bibliografia a mesura que es va elaborant la documentació del TFG i es van realitzant les cites a aquestes referències.

L’estil de citació que es recomana seguir és Chicago Manual of Style o ISO 690 en català o castellà.

* Com es presenta una bibliografia i se cita un text

http://publica.upc.edu/ca/estil/bibliografies

* Com fer una bibliografia amb Mendeley

<http://publica.upc.edu/ca/estil/bibliografies#Mendeley_catala>

https://www.youtube.com/watch?v=eVfYMUbfpeI

# Annex A

En cas de necessitar posar annexos, aquests es posaran a partir d’aquesta secció

En els Annexos s’hi poden incloure si s’escau:

* Plec de Condicions i/o Manual d’Usuari. Que ha d’incloure:
  + El Plec de Condicions Tècniques Particulars / Manual d’Usuari/a / Manual d’Operacions i Manteniment.
  + El Plec de Condicions Econòmiques Generals (quan sigui procedent).
  + Detall de la normativa aplicable al treball (quan sigui procedent).
  + Programa i pla de fabricació o execució i/o instruccions de muntatge.
* Càlculs de dimensionat o comprovació.
* Càlculs justificatius de l’elecció de les solucions adoptades.
* Fulls de característiques de materials subministrats pels fabricants.
* Llistats d’ordinador.
* Plànols
  1. Text de l’apartat 1 de l’Annex