Institut Puig Castellar
Santa Coloma de Gramenet

PuigRobot

CFGS Administració de Sistemes Informàtics i Xarxes
CFGS Desenvolupament d'Aplicacions Multiplataforma

Nom e**studiant** Héctor Ramos Sedas

Curs acadèmic

8/03/2016

Llicències alternatives (triar alguna de les següents i substituir la de la pàgina anterior)

A) Creative Commons:

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-CompartirIgual 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-SenseObraDerivada 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-CompartirIgual 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement 3.0 Espanya de Creative Commons

B) GNU Free Documentation License (GNU FDL)

Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU Free Documentation License".

C) Copyright

Resum del projecte (màxim 250 paraules):

Abstract (in English, 250 words or less):

Paraules clau (entre 4 i 8): Robot, aprendizaje, programación, domótica, arduino

El proyecto se basa en la creación de un robot orientado al aprendizaje autodidacta de los más pequeños. Basándonos en la idea del proyecto LogoBot. Diseñaremos una alternativa más económica con la cual poder competir en versiones futuras del proyecto. La idea es adaptar el robot a las necesidades de aprendizaje de los niños, diseñando algo que no puedan romper, ni tampoco ser fácilmente dañado en el transcurso de su aprendizaje. Aún siendo una idea muy ambiciosa nos centraremos primero en hacer que el robot, intente integrar unos patrones los cuales, en este caso el alumno, deberá seleccionar para que el robot haga ciertas funciones. Como puede ser avanzar en una dirección u otra con el fin de enseñar a base de juegos a los más pequeños. Por ejemplo las direcciones, sumar... Todo esto intentando crear un cierto interés hacia la tecnología, usando el robot como herramienta de aprendizaje y ocio.

Índex

Llista de figures

1. Introducció

1.1 Context i justificació del Treball

Punt de partida del treball (Quina és la necessitat a cobrir?

- La necesidad principal del proyecto es atraer al público infantil la tecnologia y que se utilice como punto de apoyo para el aprendizaje.

Perquè és un tema rellevant?

- Porque acerca a los más pequeños a la tecnología, cosa que es cada vez es más importante saber utilizarla en la sociedad actual, además influenciar a que un futuro el alumno, quieran estudiar algo relacionado con tema .

Com res resol el problema de moment?) i aportació realitzada (Quin resultat es vol obtenir?)

- Hay alternativas en el mercado pero queremos buscar una más asequible y que se adapte más a las escuelas. Acercar la domótica a los más pequeños.

1.2 Objectius del Treball

- Pensar cual diseño podemos utilizar en nuestro prototipo.
- Crear prototipo del diseño del robot
- Programar el movimiento del robot.
- Probar funcionamiento del robot.
- Añadir funcionalidades opcionales (Controlar vía móvil o tablet)

1.3 Enfocament i mètode seguit

Indicar quines són les possibles estratègies per dur a terme el treball i indicar quina és l'estratègia triada (desenvolupar un producte nou, adaptar un producte existent, …). Valorar perquè aquesta és l'estratègia més apropiada per aconseguir els objectius.

Las estrategias serían crear un proyecto desde 0, sin seguir ningún patrón y ninguna idea concebida o la otra alternativa sería adaptar un proyecto existente y utilizarlo para nuestra finalidad.
Se trata de crear un proyecto nuevo basándonos en ideas alternativas. Puesto que ya hay alternativas que funcionan, adaptar nuestro proyecto será más fácil. Ahorrar la parte de estructura (hardware) de la domótica para centrarse en la interactividad con el usuario.

1.4 Planificació del projecte

Nombre	Fecha de inicio	Fe
Toma de requerimientos (Reunió...	24/02/16	24/02/16
Recopilación de información	25/02/16	8/03/16
Entrega de objetivo y planificació...	9/03/16	9/03/16
Creación de las piezas en máquin...	9/03/16	16/03/16
Comprar hardware/ Entrega	9/03/16	29/03/16
Probar código e realizar pruebas ...	9/03/16	8/04/16
Ensamblación y prueba de los dif...	30/03/16	8/04/16
Elección del prototipo ideal	9/03/16	30/03/16
Entrega toma de requisitos	8/04/16	8/04/16
Crear código y ajustar robot	11/04/16	27/04/16
Añadir elementos como carcasa ...	28/04/16	13/05/16
Entrega 3	14/04/16	14/04/16
Test finales	13/05/16	1/06/16
Entrega de Memòria final	3/06/16	3/06/16

2. Introducción al proyecto :

El proyecto se basa en la creación de un pequeño Robot para uso educativo. Añadiremos una pequeña introducción sobre qué es la robótica y una breve descripción de los diferentes componentes claves en nuestro proyecto.

Utilizaremos como parte “lógica” un chip basado en Arduino : Arduino Pro Mini y para la creación de las piezas utilizaremos una impresora 3D : 3D BQ Prusa i3 Hephestos.

2.1 Descripción de Robótica

La robótica es una ciencia o rama de la tecnología, que estudia el diseño y construcción de máquinas capaces de desempeñar tareas realizadas por el ser humano o que requieren del uso de inteligencia. Las ciencias y tecnologías de las que deriva podrían ser: el álgebra, los autómatas programables, las máquinas de estados, la mecánica o la informática.

2.1.1 Arduino

Arduino es una empresa de hardware libre, la cual desarrolla placas que integran un microcontrolador y un entorno de desarrollo (IDE), diseñado para facilitar el uso de la electrónica. El hardware se trata de una placa de circuito impreso compuesta con un microcontrolador, habitualmente basado en Atmel AVR, y diferentes puertos digitales y analógicos tanto de entrada como de salida, los cuales se pueden utilizar para conectarse con diferentes módulos (placas de expansión (shields) y que añaden mejoras y más características al funcionamiento de la placa de serie.
El software utilizado consiste en un entorno de desarrollo (IDE) basado en el entorno de Processing y lenguaje de programación basado en Wiring, así como el cargador de arranque que se ejecuta en la placa. El microcontrolador de la placa se programa a través de un ordenador, haciendo uso de diferentes medios, como USB, bluetooth o RS-232.

Tipos de placas Arduino;

En el mercado actual existen diferentes tipos de versiones de Arduino. Según nuestro proyecto podremos escoger una que se adapte mejor :

- **Placa Série :**

Es la placa básica y se utiliza para una interfaz RS232. Esta interfaz puede ser utilizada para programar la placa o para comunicarse con otros elementos externos que utilicen ese puerto.

- **Placa USB:**

Se trata de una placa que incluye un puerto USB para la comunicación externa con otros elementos como un ordenador.

- **Placa de prototipos:**

Se trata de una placa pensada para incorporar hardware adicional al diseño por defecto del Arduino. Contiene diferentes anclajes para poder incorporar nuestros hardware adicional. No dispone de puerto de serie ni USB y es necesario otra placa para poder programar el chip.

- **Bluetooth:**

La última versión que están desarrollando se basa en Bluetooth. Se trata de eliminar la necesidad de cables para la comunicación entre el PC o cualquier elemento con este hardware.

2.1.3 Microprocesador AVR

Los AVR son una familia de microcontroladores RISC del fabricante Atmel. Se trata de los microcontroladores integrados en el hardware de las placas Arduino. Utilizan la arquitectura AVR cuenta con bastantes aficionados por su sencillez de diseño y su fácil programación del chip. Se dividen en diferentes grupos :

- ATxmega: procesadores muy potentes con 16 a 384 kB de memoria flash programable, encapsulados de 44, 64 y 100 pines (A4, A3, A1), capacidad de DMA, eventos, criptografía y amplio conjunto de periféricos con DACs.
- ATmega: microcontroladores AVR grandes con 4 a 256 kB de memoria programable, encapsulados de 28 a 100 pines, conjunto de instrucciones extendido (multiplicación y direccionamiento de programas mayores) y amplio conjunto de periféricos.
- ATtiny: pequeños microcontroladores AVR con 0,5 a 8 kB de memória flash programable, encapsulados de 6 a 20 pines y un limitado set de periféricos.
- AT90USB: ATmega integrado con controlador USB
- AT90CAN: ATmega con controlador de bus CAN
- Tipos especiales: algunos modelos especiales, por ejemplo, para el control de los cargadores de baterías, pantallas LCD y los controles de los motores o la iluminación.
- AT90S: tipos obsoletos, los AVRs clásicos.

2.1.4 Programación en Arduino

Tras la instalación del Framework de arduino ya podremos empezar a trabajar con la placa. Al estar conectado directamente por vía USB a nuestro ordenador no tenemos la necesidad de alimentar nuestra placa para trabajar con ella.

2.1.4.1 Entorno de desarrollo

Para programar la placa es necesario descargarse de la página web de Arduino el entorno de desarrollo (IDE). Se dispone de versiones para Windows y para MAC, así como las fuentes necesarias para compilar el código en LINUX. En el caso de disponer de una placa USB es necesario instalar los drivers FTDI. Estos drivers vienen incluidos en el paquete de Arduino mencionado anteriormente. Existen en la web versiones para distintos sistemas operativos.

2.1.4.2 Estructura básica de un programa

La estructura básica de programación de Arduino es bastante simple y divide la ejecución en dos partes: setup y loop. Setup() constituye la preparación del programa y loop() es la ejecución. En la función Setup() se incluye la declaración de variables y se trata de la primera función que se ejecuta en el programa. Esta función se ejecuta una única vez y es empleada para configurar el pinMode (p. ej. si un determinado pin digital es de entrada o salida) e inicializar la comunicación serie. La función loop() incluye el código a ser ejecutado continuamente (leyendo las entradas de la placa, salidas, etc.).

```
voidsetup() {
  inMode(pin, OUTPUT);   //Establece 'pin' como salida
}
void loop() {
  digitalWrite(pin, HIGH);   // Activa 'pin'
  delay(1000);               // Pausa un segundo
  digitalWrite(pin, LOW);    // Desactiva 'pin'
  delay(1000);
}
```

Como se observa en este bloque de código cada instrucción acaba con ; y los comentarios se indican con //. Al igual que en C se pueden introducir bloques de comentarios con /* ... */.

2.2 Impresora 3d

Una impresora 3d se trata de una máquina que permite la creación de diferentes réplicas de diseños 3D, piezas o diferentes formas a partir de un diseño creado por ordenador o obtenido en Internet .
Surgen con la idea de convertir archivos de 2D en prototipos reales o 3D. Comúnmente se ha utilizado en el prototipado o en la pre-fabricación de piezas o componentes, en sectores como la arquitectura y el diseño industrial. En la actualidad se está extendiendo su uso.

Existen múltiples modelos comerciales:

- Sinterización láser, donde un suministrador va depositando finas capas de polvo de diferentes metales (acero, aluminio, titanio...) y un láser a continuación funde cada capa con la anterior.
- Estereolitografía, donde una resina fotosensible se curada con haces de luz ultravioleta, solidificando todo homogéneamente..
- Compactación, con una masa de polvo que se compacta por estratos.
- Adición, o de inyección de polímeros, en las que el propio material se añade por capas.

Según el método empleado para la compactación del polvo, se pueden clasificar en:

- **Impresoras 3D de tinta:** utilizan una tinta aglomerante para compactar el polvo. El uso de una tinta permite la impresión en diferentes colores.
- **Impresoras 3D láser:** Es un láser que transfiere energía al polvo haciendo que se polimerice. Después se sumerge en un líquido que hace que las zonas polimerizadas se solidifiquen.

Software

Para poder realizar los diferentes diseños de las piezas que se desean crear en 3D es necesario un software específico. El software utilizado se trata de CAD (diseño asistido por computadora) y existen diferentes opciones para poder elegir :

- DraftSight
- Blender
- Catia
- FreeCAD
- OpenSCAD
- SolidWorks
- Tinkercad
- AutoCAD

Estos programas contienen interfaces bastantes sencillas de utilizar por el usuario e incluyen herramientas capaces de ayudarnos a mejorar el rendimiento de nuestra creación, es decir si cumple o no con las características esperadas.

3. Requisitos de Hardware

Para la creación de nuestro proyecto basado en una arquitectura arduino, lo separaremos en dos partes, la parte CPU (Lógica) y la parte de la carrocería del robot. Para ello utilizaremos los siguientes componentes :

Parte Lógica del equipo (CPU) :

- Arduino modelo : Arduino Pro mini
- 2 Motores de Paso a paso de 5V. (PaP).
- KeyPad 3*4
- Portapilas de petaca de 9V.

Parte Carrocería :

- Para la carrocería la gran mayoría de las piezas utilizaremos la impresora 3D BQ Prusa i3 Hephestos, que nos facilitará mucho el trabajo.
- Velcro.
- Bridas.

Herramientas :

- Soldador.
- Ordenador con Windows o cualquier distribución Linux.

3.1 3D BQ Prusa i3 Hephestos

La impresora que utilizaremos para llevar a cabo las impresiones en 3D se trata de 3D BQ Prusa i3 Hephestos.

La impresora 3D Prusa i3 Hephestos es un proyecto libre diseñado y desarrollado por el departamento de Innovación y Robótica de bq. Hephestos toma la base de la Prusa i3 y añade varias mejoras extraídas de otras impresoras como la PowerCode, usuarios de la comunidad RepRap, modificaciones de estas piezas y diseños propios del departamento.

Con este proyecto se ha buscado ofrecer un diseño de impresora robusto que soluciona varias de las carencias de los diseños anteriores como por ejemplo la sujeción de los finales de carrera y el guiado de los cables.

Software:

-Firmware derivado de Marlin

-Entorno recomendado: Cura Software

-Archivos admitidos: .gcode

-OS compatibles:

1. Windows XP y superiores.
2. Mac OS X y superiores.
3. Linux

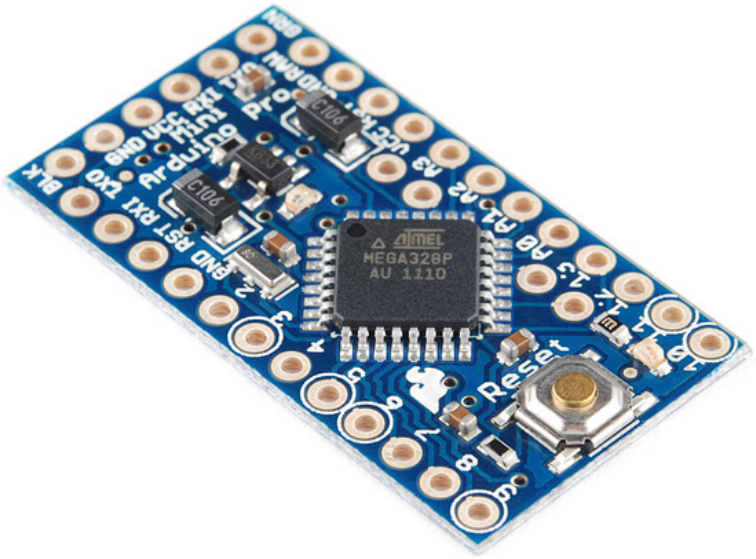
Firmware Marlin

Marlin se trata de un firmware para la electrónica de un solo procesador RepRap. Se trata de un proyecto apoyado por las empresas RAMPAS, Rambo, Ultimaker, BQ, entre otras empresas creadoras de impresoras 3D basadas en Arduino. Admite tanto USB como tarjetas SD para introducir los ficheros. Este proyecto esta contiene una licencia GNU GPL v3 y esta basado en el proyecto Sprinter. Existen diferentes promotores : thinkyhead , AnHardt , ErikZalm , DAID , Boelle , Wackerbarth , bkubicek , y Wurstnase, los cuales ayudan a la estabilidad del proyecto creando diferentes mejoras o parches si son necesarios.

Algúnas características del proyecto son :

- Movimiento basado interrupción con la aceleración lineal real .
- Alta steprate .
- Mire hacia adelante (mantener la alta velocidad cuando sea posible alta velocidad en las curvas.)
- Interrupción de protección de la temperatura en base . -Preliminary apoyo para Matthew
- Algoritmo de avance de Roberts.
- FULL Final de carrera apoyo.
- Soporte para tarjetas SD , incluyendo carpetas y nombres de archivo largos
- Soporte de LCD , tanto el carácter de base y gráfica . (20x4 Lo ideal o 128x64) .

3.2 Arduino Pro Mini



Ésta es una versión mejorada del Arduino Mini fabricada por Sparkfun que incluye un chip Atmega328 con 32Kb de ROM para programa.

Características:

- ATmega328 con una frecuencia 8MHz con resonador externo (0.5% tolerancia)
- Placa de baja tensión
- Conexión USB fuera de la placa.
- Soporte para auto reseteo
- Regulador de 3,3V
- Salida máxima de 150mA
- Protector de sobrecorriente
- Protector de polarización inversa
- Toma electrica de 3,3V hasta 12V
- LEDs de estado y potencia en la placa

Dimensiones:

- 0.7×1.3″ (18x33mm)
- Menos de 2 gramos

Desventajas :

- Al utilizar este modelo nos limita el número de conexiones posibles que podemos utilizar por defecto. Aún así, se le puede añadir un modulo para aumentar las conexiones. No podemos incluir : Led’s o buzzer para emitir sonido, ya que el motor ocupa 8 conexiones y el KeyPad 7.

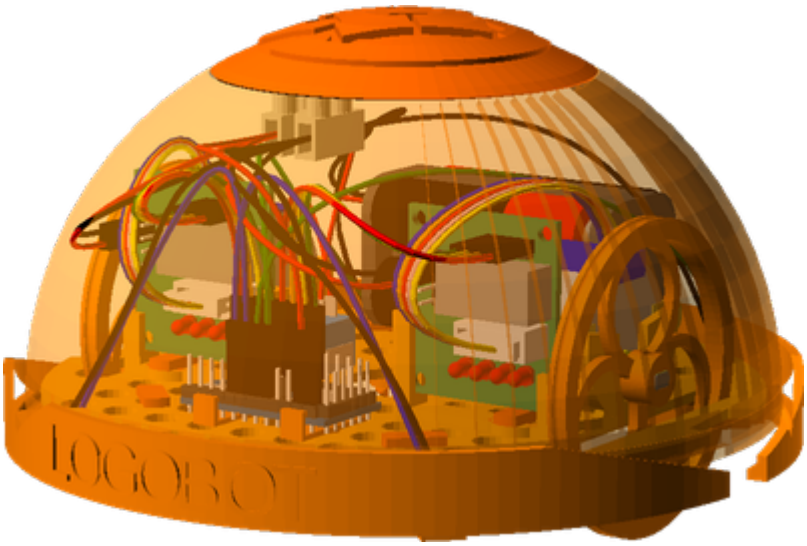
- Es necesario soldar las piezas para poder utilizarla.

Ventajas :

- Económico.
- Pequeño y versatil.

3.3 Diseños 3D

Para la creación de nuestro prototipo utilizaremos los siguientes diseños 3D que iremos cambiando y mejorando para adaptarlo mejor. Se trata del diseño del robot Logobot que tiene una licencia GPL.



Las características principales de este diseño son :

- Ayudar a enseñar a la electrónica , programación, diseño 3D y la impresión en 3D
- Altamente interactivo para atraer.
- Barato
- Para edades entre 6 - 100 años
- La simplicidad es la clave, cualquier pieza es fácilmente encajable y explicable a cualquier niño de unos 6 años de edad
- Sin soldaduras (Cosa que es práctica)
- Expandible (Le puedes añadir más módulos)
- Extremadamente personalizable, cosa que hace más atractivo el proyecto

A este proyecto se le hará una pequeña modificación para poder utilizar el Keypad.

4.1 Implementación

4.1.1 Introducción a la implementación

Realizaremos diferentes pruebas antes de soldar y llegar a cabo la unión entre nuestros diferentes componentes al robot. Para ello utilizaremos un modelo diferente de Arduino : Arduino Uno, un modelo de mayor tamaño pero que nos servirá perfectamente para nuestros ensayos y tener la versión final en perfecto estado.

Para ello utilizaremos una placa protoboard, una placa para realizar pruebas de de electrónica.

Los componentes que principalmente deberemos controlar son :

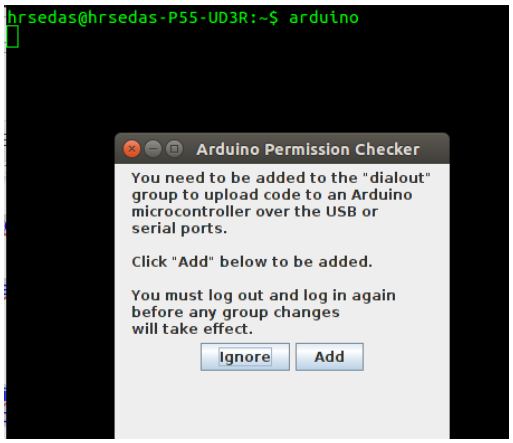
- Los motores 5 V.
- El Keypad.

Teniendo claro como poder realizar diferentes funciones con los dos, realizaremos el paso de unir los dos componentes en la placa protoboard y hacer test.
Conectaremos nuestra placa a nuestro ordenador y para poder realizar la comunicación entre placa y los diferentes componentes nos instalaremos el software de Arduino :

```
apt-get install arduino
```

Para poder sincronizar, además debemos añadir nuestro usuario al grupo dialout y cerrar sesión.

Lo primero será escoger el puerto USB que utilizaremos para conectarnos. Vamos a Herramientas -> Puerto Serial -> /dev/ttyACM0



4.2 Pruebas con Keypad

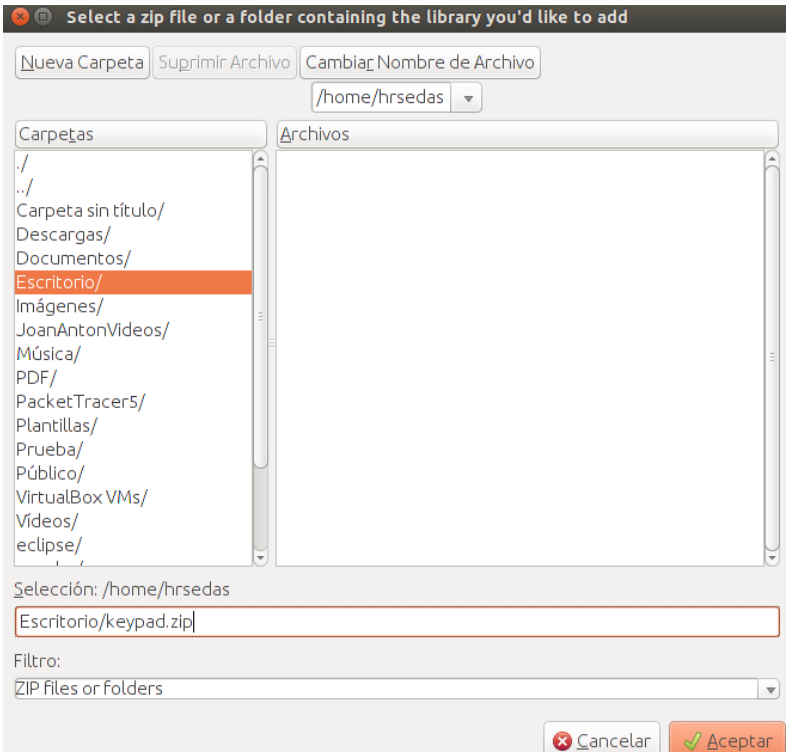
4.2.1 Importar librería Keypad

Comenzaremos a realizar pequeños tests para comprobar como funciona del Arduino.

Para poder utilizar el Keypad es necesario importar una librería que no viene por defecto en el IDE de Arduino. Accedemos a su página oficial e importamos :

http://playground.arduino.cc/code/keypad#Download

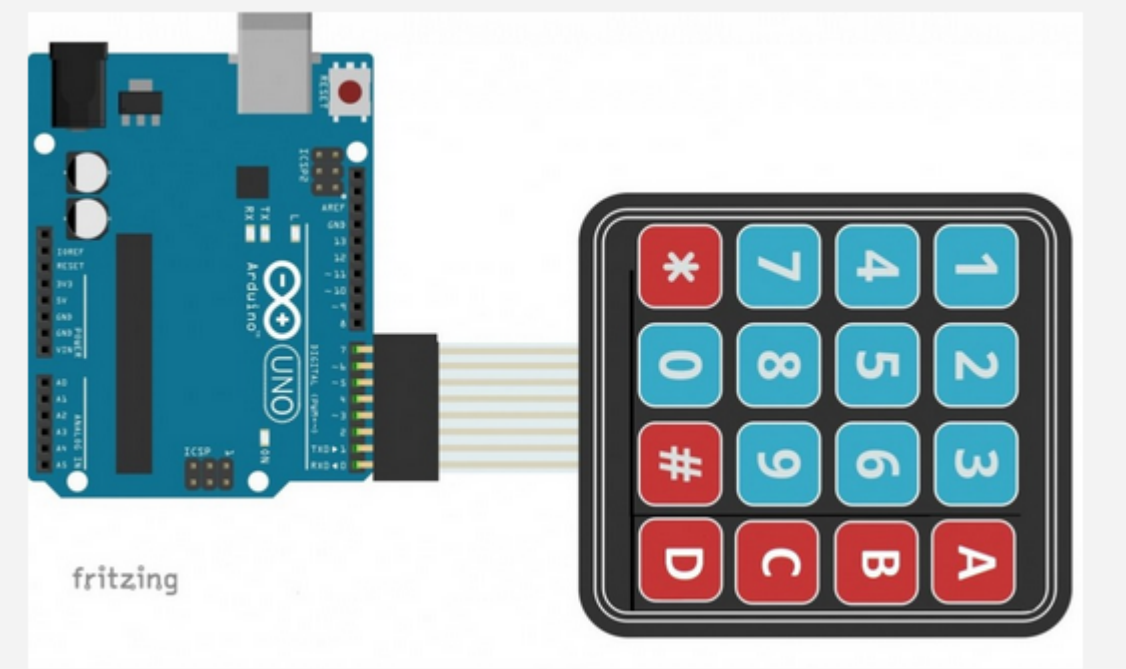
Para importar accedemos al IDE y clickamos en Sketch -> Importar librería y seleccionamos la ruta del archivo descargado :



Para poder utilizar la correctamente, reiniciamos el IDE.

4.2.2 Como utilizar la clase Keypad e implementar un pequeño código

El test para comprobar que nuestro keypad funciona seria conectarlo en esta posición :



Para utilizar la librería tan solo debemos incluir en nuestro código la librería :

```
#include <Keypad.h>
```

Y además definir un objeto Keypad, el cual es necesario pasarle 1 matriz,2 vectores , y dos variables.

- Matriz Teclado : Es donde defines los valores del teclado.
- Vector Pins_Fila : Es donde defines los pines que utilizarás de la placa (Destinados a las filas del teclado)
- Vector Pins_Fila : Es donde defines los pines que utilizarás de la placa (Destinados a las columnas del teclado)
- Filas = Número de filas del teclado.
- Columnas = Números de columnas del teclado.

```
#include <Keypad.h>
//Definimos las constantes que determinan el número total de filas y columnas que tenemos en nuetro KeyPad.

const byte Filas = 4;       //Definimos el número de filas
const byte Cols = 4;        //Definimos el número de columnas
```

```
//Los pines que usaremos en este básico ejemplo los separaremos por ( pines de columnas y pines de filas )


byte Pins_Filas[] = {9, 8, 7, 6};      //Pines Arduino a los que contamos las filas.
byte Pins_Cols[] = { 5, 4, 3, 2};      // Pines Arduino a los que contamos las columnas.

//Definimos los valores de las teclas del Keypad
char Teclas [ Filas ][ Cols ] =
{
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

//Declaramos el objeto Teclado y le añadimos los parametros necesarios para que funcione.
Keypad Keypad1 = Keypad(makeKeymap(Teclas), Pins_Filas, Pins_Cols, Filas, Cols);

void setup()
{      Serial.begin(9600) ; }

void loop()
{      char apretado = Keypad1.getKey() ;
      if (apretado != 0)                // Si la variable es 0 es que no se ha apretado o pulsado  ninguna tecla
              Serial.println(apretado);

}
```

Con este código conseguimos mostrar por la pantalla del terminal “virtual” del IDE de arduino los diferentes valores que vamos tecleando en el Keypad.

4.3 Motor Paso a Paso (PaP)

4.3.1 Motores DC

Los motores DC (proveniente del inglés “Direct Current”, corriente continua) son la clase de motores más simple. Contienen dos terminales, donde uno se conecta a la fuente de alimentación y el otro se conecta al punto de tierra, creando un giro hacia un sentido, sin embargo si se conectan al revés el giro será reverso (dirección contraria). Una de sus características es que cuanta más intensidad de corriente tenga el motor más rápido girará.
En general, los motores DC realizan un consumo eléctrico bastante elevado para conseguir la velocidad de giro adecuada. Esto quiere decir que muchas veces el pin de “5V” de nuestra placa Arduino no será suficiente, y el motor deberá ser alimentado a partir de una fuente externa, o bien mediante un amplificador de corriente (como lo es un transistor).

4.3.2 Introducción al PAP

Existen dos tipos de motor PaP :

- Unipolares
- Bipolares

Bipolares

Los bipolares se componen de 2 bobinas y los unipolares de 4 bobinas. Para diferenciarlos físicamente basta con observar el número de terminales de cada motor. Los bipolares siempre tienen 4 terminales, dos para cada bobina, y los unipolares normalmente tienen 6 terminales, dos para cada bobina y los otro dos son los comunes de estas. Hay motores unipolares con 5 terminales en que los dos comunes están unidos internamente.

Unipolares

Estos motores suelen tener 5 o 6 cables de salida dependiendo de su conexión interna. Este tipo se caracteriza por ser más simple de controlar, estos utilizan un cable común a la fuente de alimentación y posteriormente se van colocando las otras líneas a tierra en un orden específico para generar cada paso, si tienen 6 cables es porque cada par de bobinas tienen un común separado, si tiene 5 cables es porque las cuatro bobinas tienen un polo común; un motor unipolar de 6 cables puede ser usado como un motor bipolar si se deja las líneas del común al aire.

En nuestro proyecto usaremos este tipo de motores.

Para conectar a nuestra Arduino el motor paso a paso utilizaremos la siguiente tabla:

	Pines	Pines	Pines	Pines
ARDUINO	12	11	10	9
Protoboard	IN1	IN2	IN3	IN4

Donde la energía del motor irá conectada al Arduino por los pines GND y la de 3,5 V. En nuestro caso al utilizar dos motores usaremos 5V para mayor estabilidad.

4.3.1 El programa de control

Los motores paso a paso están diseñados, a diferencia de los motores de corriente continua, para que cuando reciban corriente giren tanto como tu los configures.
Para gestionar el % que deseamos que se mueva, deberemos excitar las bobinas de nuestro motor y así poder mover lo de forma continua.

En este caso nuestro motor se trata de uno de 4 fases y lo podremos hacer mover de 3 maneras deferentes :

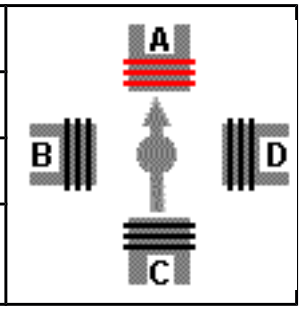
- **Excitando dos bobinas cada vez (*Suele ser lo que recomienda el fabricante*)**

Paso	Bobina A	Bobina B	Bobina C	Bobina D	
1	ON	ON	OFF	OFF	
2	OFF	ON	ON	OFF	
3	OFF	OFF	ON	ON	
4	ON	OF	OF	ON	

En este caso tendríamos máximo par, buena velocidad y alto consumo.

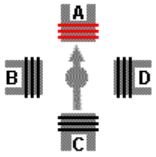
- **Excitando solo una bobina cada vez :**

Paso	Bobina A	Bobina B	Bobina C	Bobina D	

1	ON	OFF	OFF	OFF	
2	OFF	ON	OFF	OFF	
3	OFF	OFF	ON	OFF	
4	OFF	OFF	OFF	ON	

Que produciría un par menor (*Porque solo se activa una bobina en lugar de dos*) y consumo bajo.

- **Medios pasos (Intercambiando bobinas):**

Paso	Bobina 1	Bobina 2	Bobina 3	Bobina 4	
1	ON	OFF	OFF	OFF	
2	ON	ON	OFF	OFF	
3	OFF	ON	OFF	OFF	
4	OFF	OFF	ON	OFF	
5	OFF	OFF	ON	OFF	
6	OFF	OFF	ON	ON	
7	OFF	OFF	OFF	ON	
8	ON	OFF	OFF	ON	

El movimiento es más suave y lento que con los métodos anteriores, y el consumo y el par es también intermedio.

4.3.2 Implementación y diferentes librerías de PaP

Para realizar test con nuestros motores podremos utilizar 3 maneras diferentes.

- Sin utilizar librerías.
- Utilizando la librería Stepper.
- Utilizando la librería CustomStepper. (Es necesario descargarla e importarla).

Analizaremos una por una las opciones :

- 1.Sin librerías :

Se trata de crear una variable por cada pin del motor conectado al Arduino (En nuestro caso, puesto que el motor se maneja con 4 , creamos 4 variables).
Por cada tipo de movimiento que queremos que disponga nuestro motor le creamos una función.

```
// Definimos las variables que utilizará el motor. Variables → Pines :

#define motorPin1  8;  //Posición :  ----N1
#define motorPin2  9;  //Posición :  ----N2
#define motorPin3 10; //Posición :   ----N3
#define motorPin4 11; //Posición :   ----N4

//La activación de los pines depende depende del tipo de Control que deseamos utilizar
void gira_derecha(int delayTime)
{

// digitalWrite("Pin que deseamos mover", " ")*;

}
```

*La secuencia depende del tipo de Control que deseamos utilizar, el cual hemos descrito anteriormente.

```
// Configuramos que los pines 8,9,10,11 se utilizarán para salida.
void setup() {
pinMode(motorPin1, OUTPUT);
pinMode(motorPin2, OUTPUT);
pinMode(motorPin3, OUTPUT);
pinMode(motorPin4, OUTPUT);

}
```

```
// Con este for, definimos que el motor gire poco a poco. En este caso giramos en el sentido contrario al reloj.

void loop()

{
 for (int i=0; i <= 265; i++){
      gira_derecha(2);
   }
delay(2000)
}
```

- 2.Librería Stepper

// Esta clase es la predefinida por defecto para poder controlar los motores paso a paso.

```
#include <Stepper.h>

 // Definimos elnúmero de pasos en un minuto que realizará el motor en un minuto. Lo definiremos en una constante :

#define STEPS 200

//En el constructor del  tipo Stepper deberemos pasar el número de de pasos por minuto y además los pines que utilizaremos para el motor stepper.

Stepper stepper(STEPS, 8, 9, 10, 11);

void setup(){
  //definimos la velocidad que utilizará el motor paso a paso. En RPM's
  stepper.setSpeed(60);
}
```

//Con esta librería tan solo debemos definir el número …. y el motor girar hacia la posición que deseamos. El signo indica si será horario o antihorario el giro.

```
void loop(){
  stepper.step(100);
  delay(1000);
  stepper.step(-100);
  delay(1000);
}
```

- 3.Librería CustomStepper

```
#include <CustomStepper.h>

//Define los parametros caracteristicos del motor , como pines, definir el tipo de movimiento, número de vueltas, pasos por revolución y sentido del giro.
CustomStepper stepper(10, 11, 12, 13, (byte[]){8, B1000, B1100, B0100,
B0110, B0010, B0011, B0001, B1001}, 1, 200, CW);

void setup()
{
```

```
  //Define la velocidad del  motor
  stepper.setRPM(16);
  //Define el numero de pasos por vuelta
  stepper.setSPR(4075.7728395);
}

void loop()
{
 if (stepper.isDone())
 {
   delay(2000); //para que gire a intervalos de 90º, con pausas de 2 segundos
   //si se quita gira de forma continua

   //Define el sentido de rotación (CW = Sentido agujas del relo  o CCW = Inversa)
   stepper.setDirection(CCW);
   //Define el angulo de rotación
   stepper.rotateDegrees(90);

  }

  stepper.run();  // Ejecutamos esta función para iniciar y que funcione la librería.
}
```

Tras realizar varias pruebas con la librería Stepper y CustomStepper no llegaba a realizar el giro antihorario en mis motores Stepper.
En diversas webs observe que estas dos librerías presentaban problemas de compatibilidad de driver del controlador de los motores ULN2003 y finalmente utilizaré esta librería :

- AccelStepper:

Con ella podré manejar los motores tanto giro horario y giro antihorario.
Se trata de una librería diseñada por Adafruit y con ella se ha implementado dos clases. AccelStepper y MultiStepper. Con la primera podremos manejar 1 motor.

Con la segunda podremos manejar dos o más motores al mismo tiempo y logrando que aunque en este Arduino no es capaz de realizar acciones como Threads parecerá como si trabajaran al mismo tiempo.

Antes de mostrar un ejemplo analizaremos un par de funciones que contiene la librería :

Ejemplo de AccelStepper :

```
#include <AccelStepper.h>


#define HALFSTEP 4

// Definimos los pines que utilizaremos de la placa para el motor. Como se trata de una constante utilizaremos la palabra clave “define”, después el nombre de la variable que utilizaremos y después el pin que utilizaremos.

#define motorPin1  8    // IN1 para el controlador ULN2003
#define motorPin2  9    // IN2 para el controlador ULN2003
#define motorPin3  10  // IN3 para el controlador ULN2003
#define motorPin4  11  // IN4 para el controlador ULN2003

// Inicializamos la clase AccelStepper que es la encargada de poder controlar el motor 28BYJ-48. El constructor de AccelStepper se trata de 5 variables. La primera se decide con una variable de tipo entero y sirve para definir el número de pasos que realizara por .. Las 4 siguientes pondremos los pines que utilizará el el motor, para ello utilizamos las variables definidas anteriormente como motorPin1 hasta motorPin4
AccelStepper stepper1(HALFSTEP, motorPin1, motorPin3, motorPin2, motorPin4);

void setup() {
  //Definimos las características de
  stepper1.setMaxSpeed(1000.0);
  stepper1.setAcceleration(100.0);
  stepper1.setSpeed(200);
  stepper1.moveTo(2000);

}

// En el loop decidimos que el motor se mueva . Si la posición se
void loop() {

  if (stepper1.distanceToGo() == 0) {
        stepper1.moveTo(-stepper1.currentPosition());
  }
  stepper1.run();
}
```

4.4.1 Unión de Keypad y Pap

Finalmente una vez concluido los test con el Keypad y los motores, unimos las dos opciones. Con este ejemplo, al pulsar la tecla ‘A’ del Keypad moveremos el motor 1 hacia adelante y al pulsar la tecla ‘B’ moveremos el 1 motor hacia atrás.

```
#include <Keypad.h>
#include <AccelStepper.h>
#define HALFSTEP 4

// Motor pin definitions
#define motorPin1  14  //  IN1 para el controlador ULN2003
```

```
#define motorPin2  15  //  IN2 para el controlador ULN2003
#define motorPin3  16  //  IN3 para el controlador ULN2003
#define motorPin4  17  //  IN4 para el controlador ULN2003

const byte Filas = 4;
const byte Cols = 4;

byte Pins_Filas[] = {9, 8, 7, 6};
byte Pins_Cols[] = { 5, 4, 3, 2};


char Teclas [ Filas ][ Cols ] =
{
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

AccelStepper stepper1(HALFSTEP, motorPin1, motorPin3, motorPin2, motorPin4);


Keypad Keypad1 = Keypad(makeKeymap(Teclas), Pins_Filas, Pins_Cols, Filas, Cols);


void setup()
{       Serial.begin(9600) ;
        stepper1.setMaxSpeed(1000.0);
        stepper1.setAcceleration(100.0);
        stepper1.setSpeed(200);


}

void loop()
{
  char apretado = Keypad1.getKey() ;
  switch(apretado){
        //Mueves el motor 1
        case 'A':
        stepper1.moveTo(2000);

        break;

        case 'B':
        stepper1.moveTo(-2000);
        break;
  }
   stepper1.run();

}
```

Tras este paso uniremos los 2 motores con el Keypad.
Este paso es muy importante puesto que gracias a él podremos mover el robot a nuestro antojo. Decidiendo a nuestro gusto el número de centímetros que se moverá el robot al pulsar cada tecla. Finalmente, como deseamos que funcionen dos motores y el keypad deberemos realizar una pequeña modificación y colocar los pines del segundo motor en un puerto analógico. Al ser de salida podremos utilizarlo sin problemas. Para tratar los puertos analógicos en el código los definiremos como 14, 15 ,16 y 17, y los trataremos igual que las otras variables.

```
#include <AccelStepper.h>
#include <Keypad.h>
#define HALFSTEP 4

// Definimos los pines que utilizara la placa para el motor

#define motorPin1  10  // IN1 para el controlador ULN2003
#define motorPin2  11  // IN2 para el controlador ULN2003
#define motorPin3  12  // IN3 para el controlador ULN2003
#define motorPin4  13  // IN4 para el controlador ULN2003

#define motorPin1  14  // IN1 para el controlador ULN2003
#define motorPin2  15  // IN2 para el controlador ULN2003
#define motorPin3  16  // IN3 para el controlador ULN2003
#define motorPin4  17  // IN4 para el controlador ULN2003
```

//Definimos el número de columnas y filas que tiene el Keypad

```
const byte Filas = 4;
const byte Cols = 4;
```

//Las guardamos en un array los pines que utilizará el Keypad.

```
byte Pins_Filas[] = {9, 8, 7, 6};
byte Pins_Cols[] = { 5, 4, 3, 2};
```

//En una matriz guardamos los valores que contienen el Keypad. No tienen que coincidir con la realidad, es un valor realativo, que cuando se pulsa esa fila y esa columna se interpreta como ese valor.

```
char Teclas [ Filas ][ Cols ] =
{
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
```

//Inicializamos los constructores

```
AccelStepper stepper1(HALFSTEP, motorPin1, motorPin3, motorPin2, motorPin4);
AccelStepper stepper2(HALFSTEP, motorPin5, motorPin7, motorPin6, motorPin8);
Keypad Keypad1 = Keypad(makeKeymap(Teclas), Pins_Filas, Pins_Cols, Filas, Cols);

//Otorgamos valores a los diferentes atributos  de las clases.
void setup()
{      Serial.begin(9600) ;
       stepper1.setMaxSpeed(1000.0);
       stepper1.setAcceleration(100.0);
       stepper1.setSpeed(200);
       stepper2.setMaxSpeed(1000.0);
       stepper2.setAcceleration(100.0);
       stepper2.setSpeed(200);
```

```
}
void loop()
{
  char apretado = Keypad1.getKey() ;
  switch(apretado){
       //Mueves el motor 1
       case 'A':
       stepper1.moveTo(2000);

       break;
       //Mueves el motor 2
       case 'B':
       stepper2.moveTo(-2000);
       break;
  }
   stepper1.run();
   stepper2.run();
}



#include <Keypad.h>
#include <AccelStepper.h>
#include <MultiStepper.h>
#define HALFSTEP 4

// Definimos los pines que utilizara la placa para el motor

#define motorPin1  10  // IN1 para el controlador ULN2003
#define motorPin2  11  // IN2 para el controlador ULN2003
#define motorPin3  12  // IN3 para el controlador ULN2003
#define motorPin4  13  // IN4 para el controlador ULN2003

#define motorPin1  14  // IN1 para el controlador ULN2003
#define motorPin2  15  // IN2 para el controlador ULN2003
#define motorPin3  16  // IN3 para el controlador ULN2003
#define motorPin4  17  // IN4 para el controlador ULN2003

//Definimos el número de columnas y filas que tiene el Keypad

const byte Filas = 4;

const byte Cols = 4;

//Las guardamos en un array los pines que utilizará el Keypad.

byte Pins_Filas[] = {9, 8, 7, 6};

byte Pins_Cols[] = { 5, 4, 3, 2};

byte i = 0;


//En una matriz guardamos los valores que contienen el Keypad. No tienen que
coincidir con la realidad, es un valor realativo, que cuando se pulsa esa fila y
esa columna se interpreta como ese valor.

char Teclas [ Filas ][ Cols ] =
{
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

//Definimos el array donde guardaremos los movimientos máximo 120 movimientos
```

```
char mov [120];

//Inicializamos los constructores

Keypad Keypad1 = Keypad(makeKeymap(Teclas), Pins_Filas, Pins_Cols, Filas, Cols);

AccelStepper stepper1(HALFSTEP, motorPin1, motorPin3, motorPin2, motorPin4);

AccelStepper stepper2(HALFSTEP, motorPin5, motorPin7, motorPin6, motorPin8);

MultiStepper multi;

//Array que determina la la posición donde esta guardado el Stepper dentro del
obejto MultiStepper.

long position [2];

//Definimos la variable donde guardaremos el valor que nos proporciona el Keypad.
char apretado;

//Otorgamos valores a los diferentes atributos  de las clases.

void setup() {
         Serial.begin(9600) ;
         stepper1.setMaxSpeed(200);
          stepper2.setMaxSpeed(200);
          multi.addStepper(stepper1);
          multi.addStepper(stepper2);
}
//Función para ir hacia adelante

void forward(char mov [],int x){
    mov[x]=' ';
    position[0]=position[0]+500;
    position[1]=position[1]-500;
    multi.moveTo(position);
    multi.runSpeedToPosition();
}
//Función para ir hacia atrás.
void behind(char mov [],int x){

  mov[x]=' ';
  position[0]=position[0]-500;
  position[1]=position[1]+500;
  multi.moveTo(position);
  multi.runSpeedToPosition();
}



//Función para mover hacia la izquierda 90ª aproximadamente

void left(char mov [],int x){

  mov[x]=' ';
  position[0]=position[0]-1000;
  position[1]=position[1]-1000;
  multi.moveTo(position);
  multi.runSpeedToPosition();
}

//Función para mover hacia la derecha 90ª aproximadamente

void right(char mov [],int x){
  mov[x]=' ';
  position[0]=position[0]+1000;
  position[1]=position[1]+1000;
```

```
  multi.moveTo(position);
  multi.runSpeedToPosition();
}

void loop(){
       //Estoy constantemente comprobando si se ha pulsado un botón
       apretado = Keypad1.getKey();
       switch(apretado) {
              case 'A':     //Hay cuatro movimientos posibles (adelante, atrás,
izquierda, derecha --> 'A','B','C','D') a guardar
              case 'B':
              case 'C':
              case 'D':
                     mov[i]=apretado;
                     i = i+1;
                     break;
              case '*':     //Ejecuto el movimiento (recorriendo de principio a
fin el array)
                     byte numElemArray=sizeof(mov)/sizeof(char);
                     for(int x=0;x<numElemArray;x++){ //A cada elemento reseteo el
array, indico el movimiento del motor y lo realizo
                            if(mov[x] == 'A') { forward(mov,x);}
                            if(mov[x] == 'B') { right(mov,x);}
                            if(mov[x] == 'C') { left(mov,x);}
                            if(mov[x] == 'D') { behind(mov,x);}
                     }
              i = 0;  //Sitúo el índice al principio para poder volver a empezar
a rellenar el array de nuevo
               break;
      }
}
```

5.1 Creación de las piezas utilizando la impresora 3d Hephestos

Una vez que tenemos el código implementado y que funciona correctamente nos centraremos en crear el chasis de nuestro, para ello utilizaremos la impresora 3d. Para poder utilizar la impresora 3D utilizaremos gcode un lenguaje de programación que se usa especialmente en entornos de automatización, es decir , un lenguaje utilizado para determinar mediante un ordenador los diferentes movimientos que debe de realizar una máquina/herramienta.

El proyecto que utilizamos como base tiene por defecto las instrucciones en el tipo de ficheros still y deberemos transformar el formato a gcode, ya que es el formato que entiende el firmware de la impresora.

El núcleo principal de nuestra impresora 3D se trata de una placa arduino y que contiene un shield especifico para controlar los motores stepper de gran tamaño. Este Arduino tiene como firmware Merlin, un proyecto de licencia GPL v3 ya comentado en el apartado de la impresora 3d.

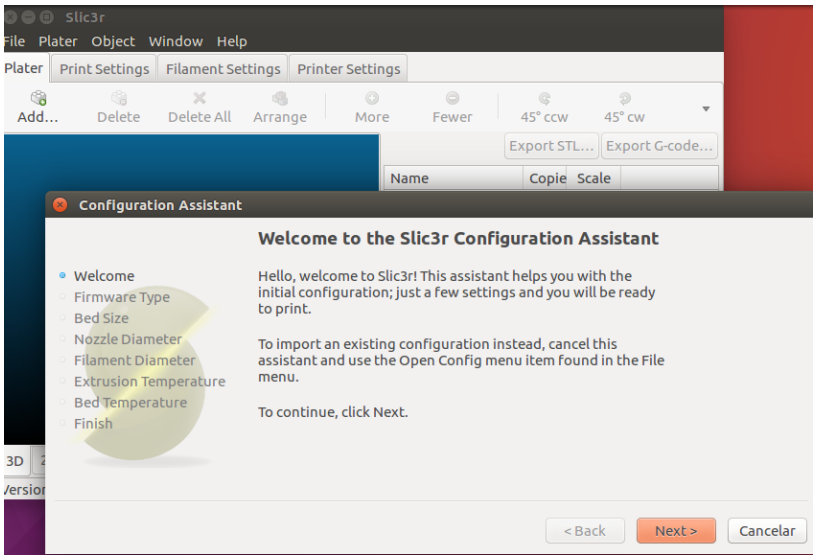
Para proceder a transformar los archivos a gcode utilizaremos el software Slic3r.

Se trata de un programa de software libre que puede generar código gcode a partir de diferentes tipos de archivos como CAD en 3D (STL o OBJ).

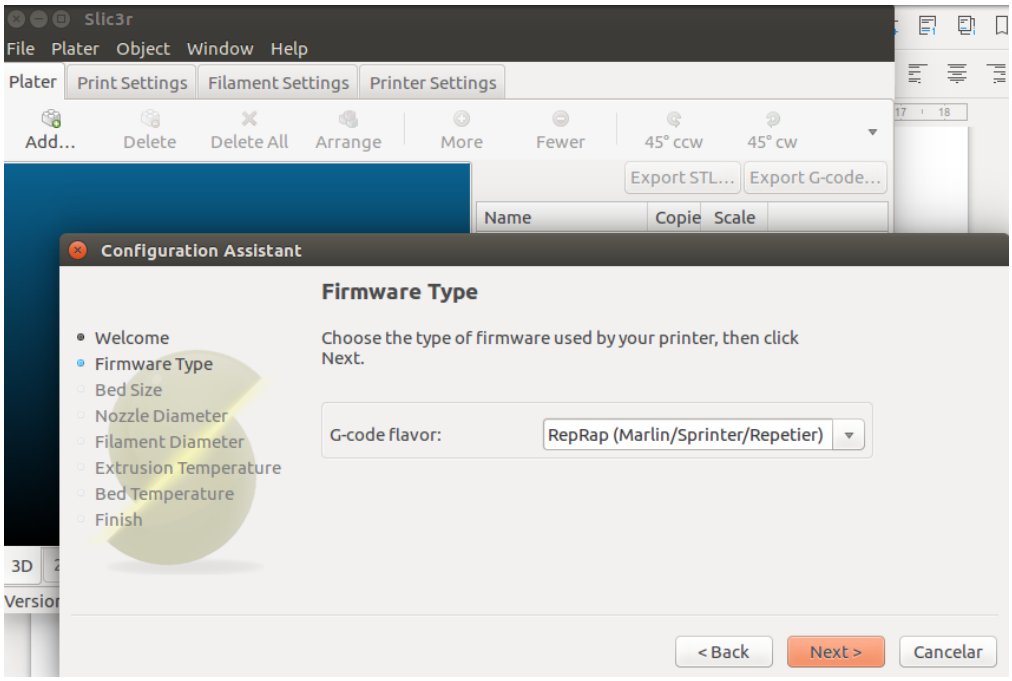
Nos podemos descargar el software desde los mismos repositorios de Ubuntu, en otras versiones también está disponible vía la web de los creadores.

Tras la instalación procedemos a la configuración según la impresora que dispongamos. Tenemos 2 opciones : descargarnos e importar configuraciones creadas por diferentes usuarios o también configurar nosotros mismos la impresora. Decidimos configurar nosotros mismos a través de las diferentes características disponibles en la web del fabricante.

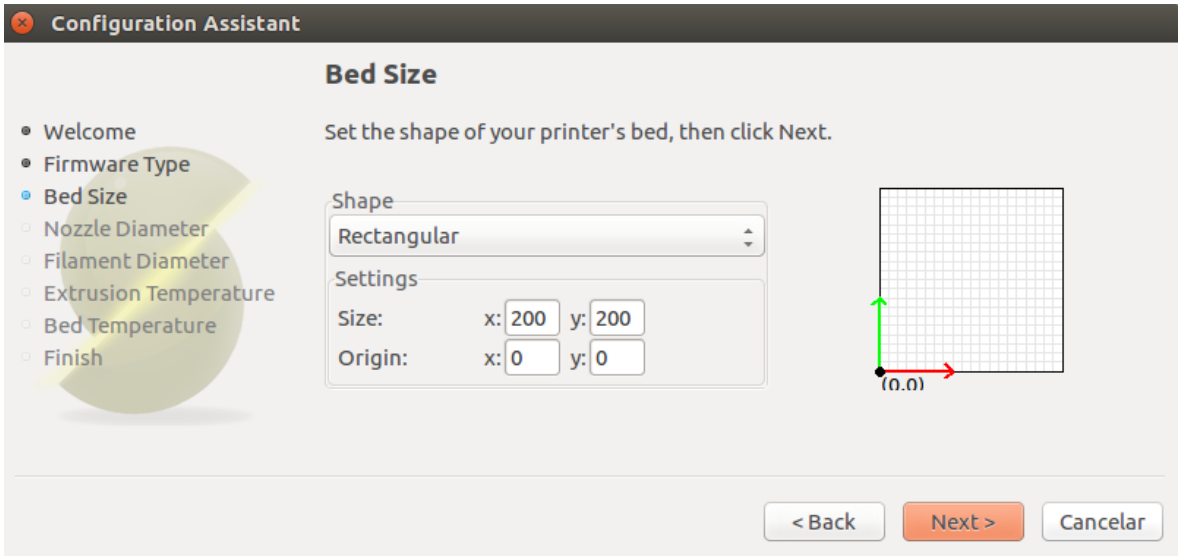
Utilizaremos el configurador, donde paso a paso introduciremos todos los datos que nos pida.



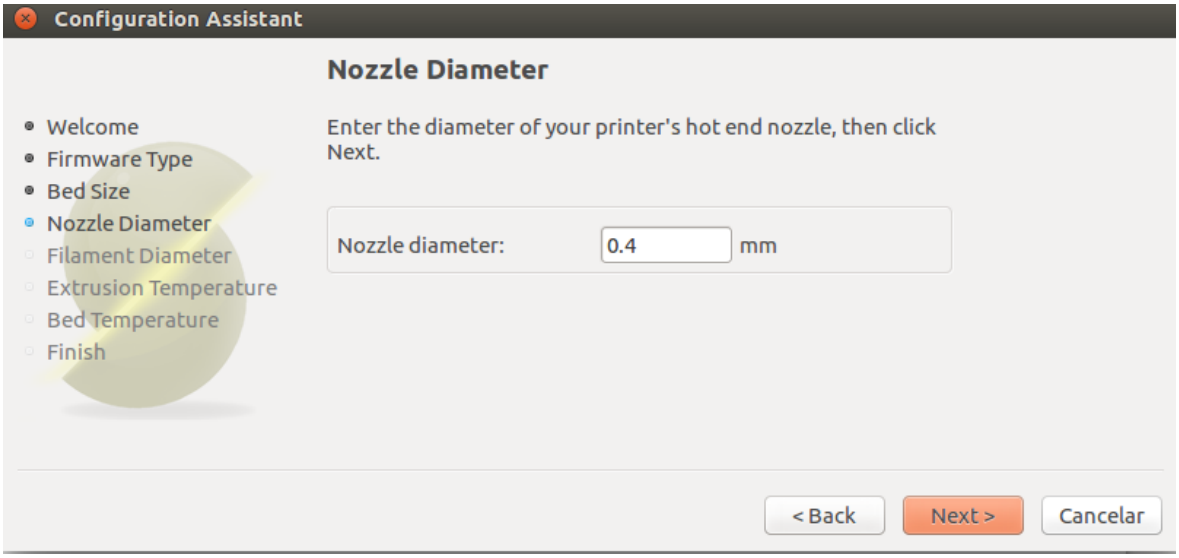
- Firmware utilizado por la impresora :



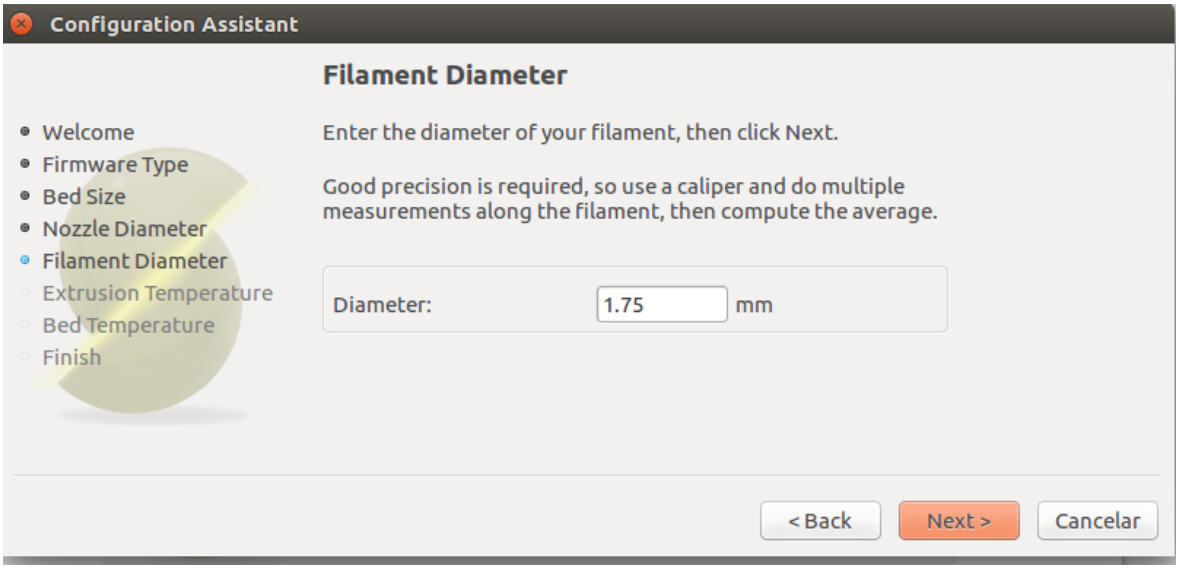
- Tamaño del plato



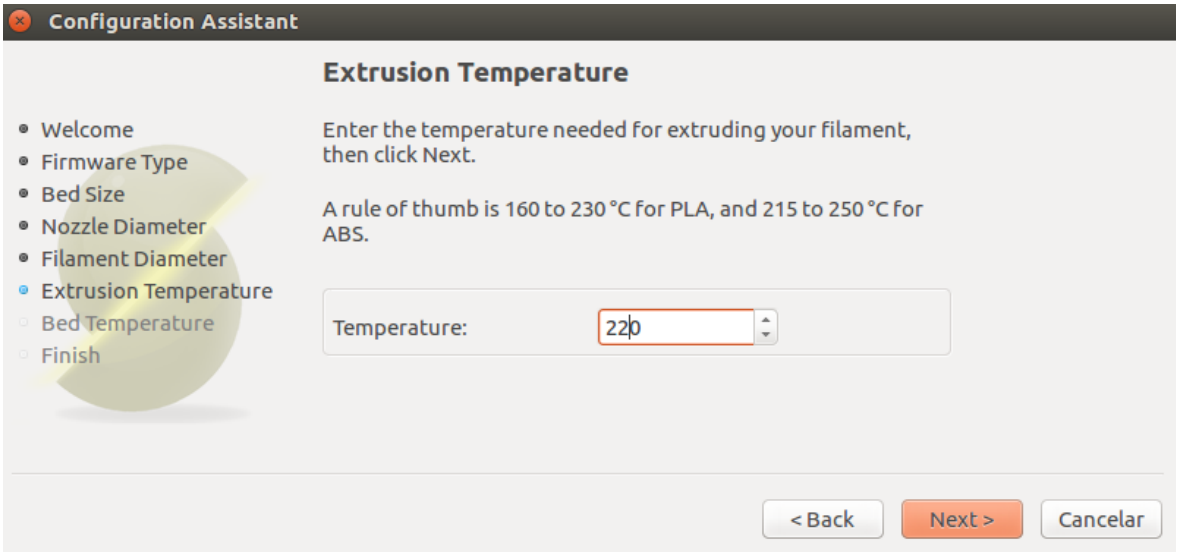
- Diámetro de la boquilla :



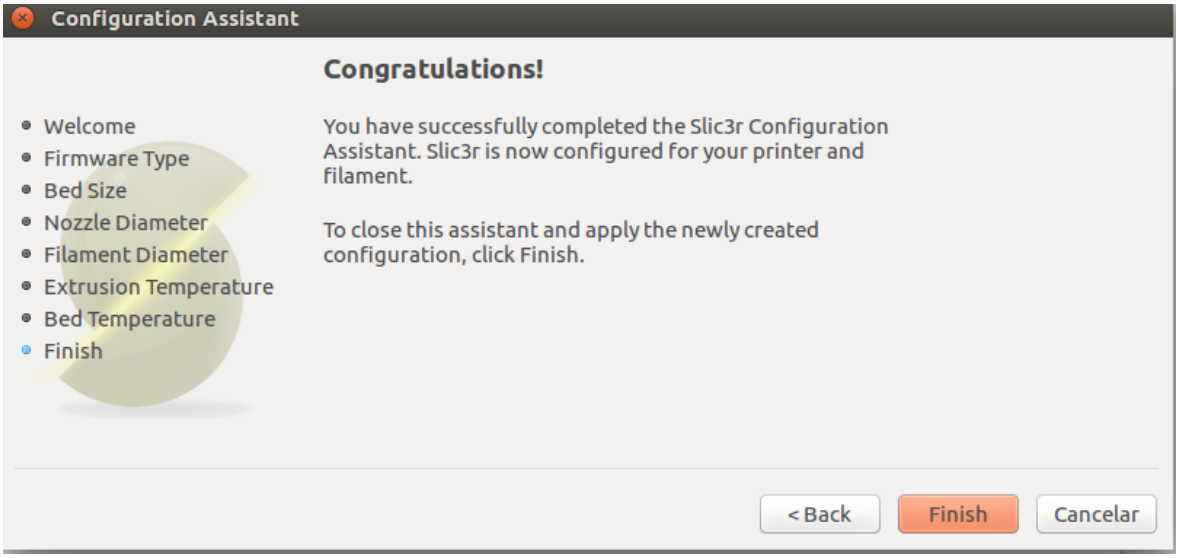
- Diámetro del filamento :



- Temperatura :



- Finalmente tenemos nuestro programa configurado :



Tras estos pasos podremos importar los archivos stil y transformarlos a gcode.

Una vez exportados , los pasamos a una tarjeta SD y la introducimos en la impresora. En ella escogemos el modelo que deseamos crear y tras realizar unos cálculos la impresora empezará a crear la pieza que hemos elegido.

6.2 Ensamblar componentes con las piezas creadas con la impresora 3D.

Tras la creación de todas las piezas proseguimos a ensamblar. Para ello seguiremos el tutorial existente del proyecto logobot , sin embargo haremos pequeñas modificaciones.

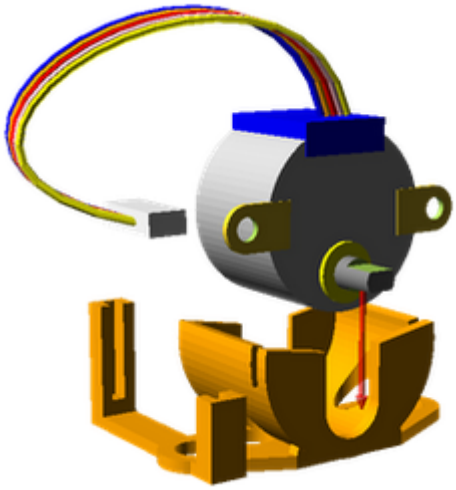
Los pasos a seguir en la guía oficial son estos :

http://rawgit.com/swindonmakers/LogoBot/master/hardware/docs/LogoBotAssemblyGuide.htm

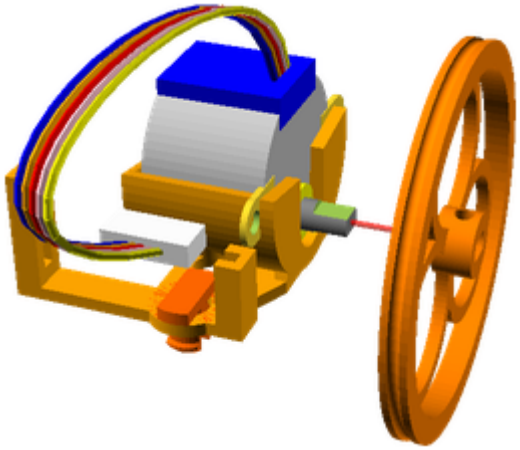
De esos pasos hemos seguir la unión de los motores con el stepper, los demás pasos se han modificado puesto se ha utilizado el Arduino modelo Uno en vez del Pro mini, por tema de ahorrarnos tiempo soldando .

- Unión motores stepper con su soporte (tanto lado izquierdo como derecho) :

  - Introducimos dentro de la pieza, el motor.

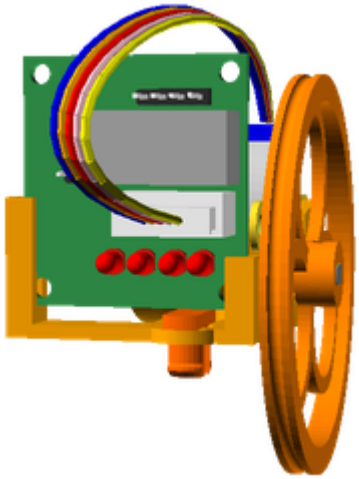


- Unimos la rueda con el motor



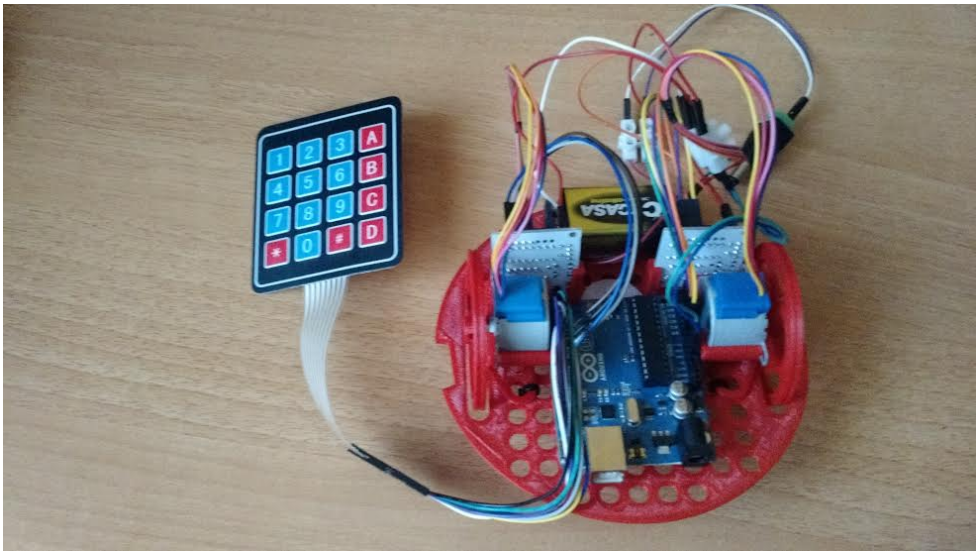
Cabe destacar en este caso que la rueda no entra con soltura en la clavija del motor y es necesario o modificar el código de la pieza en 3D o limar por dentro un poco para poder encajar las dos piezas.

- Colocamos en su soporte el controlador del motor.

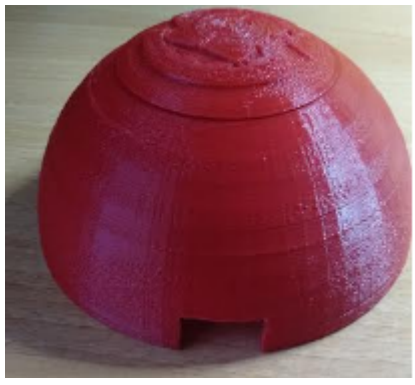


Para unir la pieza que contiene el motor y su soporte con la base , he utilizado bridas puesto que las piezas que se ha imprimido para unir las dos partes son demasiado duras y da la sensación de poder romperse.

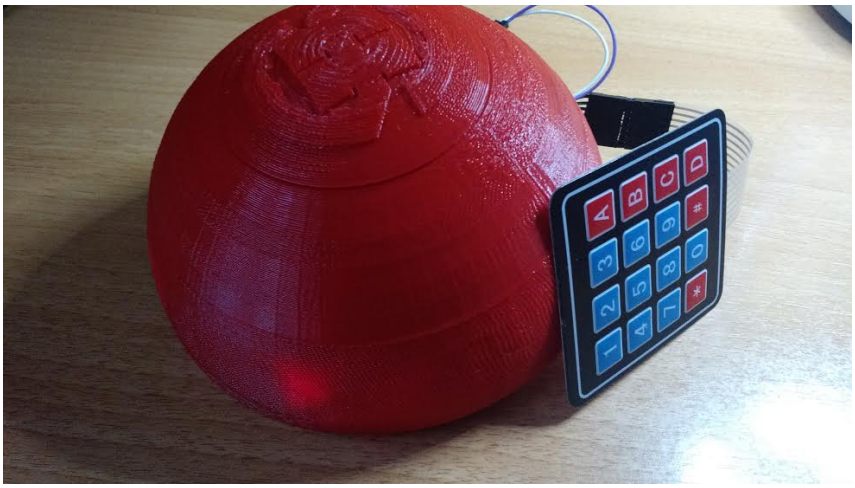
Tras estos pasos la unión del Arduino en la base se coloca en medio. Para fijar el Arduino se ha utilizado “velcro” para poder ser más fácil su extracción si es necesario realizar modificaciones en el código o incluir algún nuevo modulo para realizar nuevas tareas. (algún shield o algún controlador).



Finalmente para dar energía a la placa y los motores se ha optado por una pila de 9v. Se ha realizado un pequeño corte en la parte trasera del casco o “protector” para poder extraer el keypad y además como no estaba enfocado a esta placa era necesario ya que por volumen el cable de corriente del Arduino hacia la batería no cabía.



** Una opción es añadir una pequeña rueda que no este conectada a ningún motor para dar más sostenibilidad a la estructura del robot. Yo he añadido una pequeña bola para que la base no toque directamente el suelo.* Se ha realizado un pequeño agujero para poder conectar y desconectar la corriente de la placa y además poder sacar al exterior el Keypad para su uso.



7. Estudio de rentabilidad

Tras la realización del montaje e implementación del código haré una pequeña tesis donde calculando las ventajas/inconvenientes, coste de los productos y su facilidad de poder obtenerlos , y sí sale rentable para crear una posible la producción a pequeña escala de este proyecto con respecto a los productos existentes en el mercado .
Empezare por la disponibilidad de los productos.
El núcleo de nuestro proyecto se trata del Arduino , tanto el modelo Uno como el Pro mini, y es cierto que vía web se puede encontrar fácilmente, pero a gran escala es mejor tener una tienda física donde poder reclamar por ejemplo el tema de la garantía o otros aspectos que vía web son más difícil de solucionar. Existen tiendas físicas en Barcelona donde venden estos tipos de componentes , sin embargo muchas de ellas te venden packs ya dedicados y no solo la placa especifica. Las tiendas físicas que ofrecen estos productos encarecen bastante el precio tanto de las placas y de los motores.
Estos motores me han sido bastante dificultosos encontrarlos. No existen muchos promotores que ofrezcan este producto, a pesar de que en todos los tutoriales que existen vía web lo recomiendan por tema de precio y tamaño. Hablando con el propietario de una tienda me informo que estos motores lo exportan directamente de china y en un principio no hay problema de tener disponibilidad pero estas dependiendo de no saber si a la larga dejarán de fabricar este modelo y es un riesgo bastante grande si el día de mañana no dispones de un material tan importante.

Los precios de los productos finalmente son :

- Arduino Pro mini : vía web – 4 € vía tienda física – 14,46 € ó
- Arduino Uno : vía web – 4 € vía tienda física – 14,46 €
- Motores Stepper con controlador *2 : vía web –7,95 vía tienda física 9,45 €
- Pila de 9V : vía web – 1,5 € vía tienda física – 4,5 €
- Porta pilas para Arduino : vía web 1,95 € – vía tienda física 2,50 €
- Keypad : vía web 2,20 € – vía tienda física 7,5 €
- Cables para unir los diferentes componentes eléctricos : 3 €
- Pack bridas : vía web 1,12 € – vía tienda física 2 €
- Bloques de terminales : vía tienda física 0,40 €
- Pack material para fabricar piezas en 3D : 22,26 vía web = vía física

Total vía web :~ 30 € (Precio relativo según tienda online + gastos de transporte)
Total vía Física :~ 50 € (Precio relativo según tienda física *)

*Es posible llegar a encontrar más barato las placas Arduino pero precio medio es este.

El precio de los robots comerciales están sobre 60 ~ 70 euros. Contando que los productos los compráramos vía web el resultado sería aproximadamente unos 30 euros , si lo compráramos vía tienda física 50 €, sin contar el % que se dedica de la bobina a cada robot.

Vía web tenemos la desventaja del tiempo en recibir los materiales, pero en física si no disponen en stock tenemos el mismo problema además del tema de aumentar los precios.

Si decidimos producir productos a pequeña escala, sería un poco dificultoso si nos topamos con el problema de necesitar todo los materiales de un día para otro, y si la escasez de los motores nos pasa factura, podríamos encontrarnos en un pequeño bucle que se podría solucionar modificando los motores stepper por unos DC, siendo estos menos precisos. Además habría que modificar el código para que logrará funcionar igual.

Tras el tema de componentes tenemos que también se ha modificado un poco la estructura física de la cúpula. Si se procede a realizar mediante el Arduino pro mini tendríamos la dificultad de tener que soldar las piezas. Sin embargo si utilizamos el Arduino Uno deberíamos modificar la cúpula para poder introducir todos los componentes correctamente dentro.

Creo que es posible realizar el proyecto en pequeña escala y producir 3 o 4 robots, pero nos podríamos encontrar con el problema de escasez de motores stepper que podrían ser sustituidos por motores DC. Si comparamos con los proyectos comerciales saldría a cuenta producir por nosotros mismos.

8. Conclusions

Aquest capítol ha d'incloure:

- Una descripció de les conclusions del projecte: Quines lliçons s'han aprés del projecte?

    - Programación para Arduino desde 0.
    - Unir diferentes librerías en Arduino.
    - Saber crear un pequeño estudio de viabilidad.
    - Entender el concepto de motor DC y motores Stepper.

- Una reflexió crítica sobre l'assoliment dels objectius plantejats inicialment: Hem assolit tots els objectius? Si la resposta és negativa, per quin motiu?

    - Se ha llegado a realizar el prototipo del robot, sin embargo no se ha logrado añadir ciertas características como se tenía planeado como controlar vía bluetooth el robot y crear su propio programa en Android. También las ruedas chocan un poco con con la carrocería por el tema de que no esta orientado a este tipo de motores y ha sido necesario ajustar un poco. Intentando ajustar las ruedas lo máximo posible se ha roto la rueda y el motor y ha sido necesario la creación de una nueva y encontrar una tienda que tuviera un motor Stepper de las mismas caracteristicas.

- Una anàlisi crítica del seguiment de la planificació i metodologia al llarg del producte: S'ha seguit la planificació? La metodologia prevista ha estat prou adequada? Ha calgut introduir canvis per garantir l'èxit del projecte? Per què?

    - Se ha seguido la planificación, pero al tener una ruptura en los componentes del equipo, el peso de todo el trabajo a recabado todo solo a una persona. Pienso que si el trabajo se hubiera repartido proporcionalmente se podría haber realizado una todos los objetivos y realizado un estudio más elaborado del tema. Se ha realizado un par de cambios. En vez de utilizar la placa Arduino pro mini , se ha intercambiado por la placa Arduino Uno, con las mismas características pero de un tamaño mayor, ya que con eso nos hemos ahorrado soldar las piezas y encontrarnos con posibles fallos. La estructura , es decir las piezas 3D , se han tenido que modificar para poder adaptarse a ese cambio de placa. Hay que incluir que la impresora 3D no se consiguió tener a punto en el plazo que nos hubiera gustado tenerla para realizar más pruebas y poder crear nuestro propio modelo siguiente como modelo el proyecto "Logobot".

- Les línies de treball futur que no s'han pogut explorar en aquest projecte i han quedat pendents.

    - Realizar un pequeño script para poder manejar con un controlador bluetooth los motores Stepper.
    - Poder modificar y crear nuestro propio modelo 3D e imprimirlo.
    - Ajustar el modelo en 3D más orientado a los nuevos componentes y ajustar el problema del roce de las ruedas con el chasis.
    - Conseguir un método más efectivo para apagar y encender el Arduino como un interruptor.

9. Glossari

Definició dels termes i acrònims més rellevants utilitzats dins la Memòria.

- Arduino : * *Observar punto 2.1.1*

- Motor Paso a paso : * *Observar punto 4.3.2*

- Motor DC : * *Observar punto 4.3.1*

- Programación : La programación informática o programación algorítmica, acortada como programación, es el proceso de diseñar, codificar, depurar y mantener el código fuente de programas computacionales.

7. Bibliografia

Se ha utilizado las siguientes referencias :

- **Curso Práctico de Formación sobre Arduino :**

Autor: Oscar Torrente Artero.
Nº de páginas: 588 págs.
Encuadernación: Tapa blanda
Editorial: RC LIBROS (SC LIBRO)
Lengua: CASTELLANO
ISBN: 9788494072505

- **Wikipedia :** https://www.wikipedia.org/

- **Cursos de Arduino :** http://www.prometec.net/

-**Logobot:**
http://rawgit.com/swindonmakers/LogoBot/master/hardware/docs/index.htm