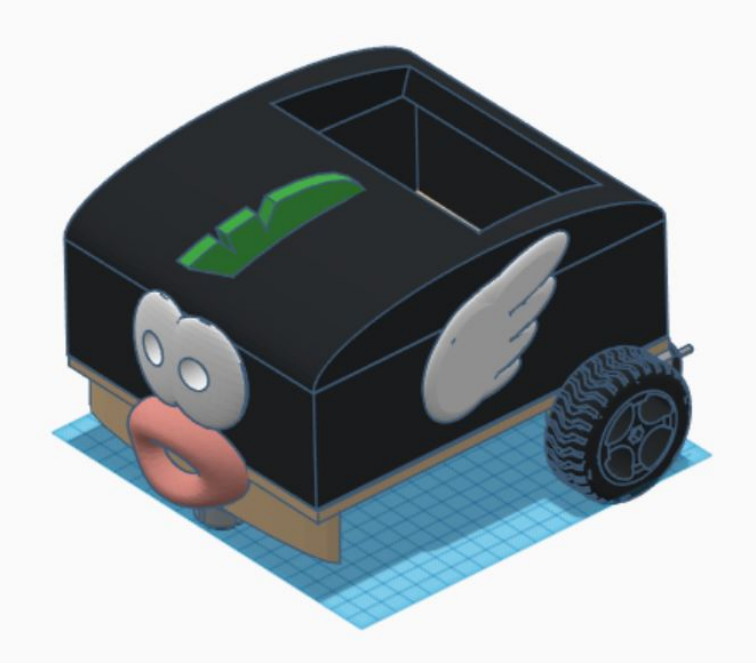

DRIVE ART

Sprint #4

23 Mayo 2019

Universitat Autònoma de Barcelona

Stefany Ariana Chóez Bolaños 1459134

Daniel Panadero Espinosa 1458674

César Valcarce Pagán 1388811

Marina Riera Velasco 1457466

Contents

Descripción del proyecto 2
Componentes electrónicos 2
Esquema 3
Arquitectura Software 4
Otros Esquemas: sketch del robot 5
Contribuciones 6
Componentes adicionales y piezas 3D 7
Riesgos previstos y plan de contingencia 8
Código 9
Aplicación 13
Filtering 15
    Curvas cuadráticas 15
    Curvas de Bezier 16
    Resultados 17
Sprint #4 18
Proyecto en el cual está inspirado 20

Descripción del proyecto

Drive Art es un coche robot controlado mediante los gestos que el usuario hace con la cabeza, que permite al usuario que lo controla realizar un dibujo físico (en el suelo). Drive Art dispone de un sensor de ultrasonidos frontal, que evitará que el coche choque con posibles obstáculos.

Este proyecto se ha pensado para tener una posible utilidad para las personas con paraplejia, por ese motivo presenta este tipo de control mediante gestos, y busca la interacción con el usuario para garantizar entretenimiento y jugabilidad.

Para tal efecto se desarrollará una aplicación móbil que recoja los gestos realizados con la cabeza y se comunique a tiempo real con el robot. Además, teniendo en cuenta que el dibujo final que se realizará con los gestos puede no quedar bien definido debido a ruido o incluso pequeños movimientos involuntarios, se realzará un filtrado digital al dibujo una vez acabado, para mejorar la definición del trazado, y se mostrará por pantalla el resultado final.

Componentes electrónicos

- Arduino nano
- HC-SR04 ultrasonic sensor
- Motor shield l298n
- Motores DC y ruedas
- Baterías AA recargables
- Base para baterías
- Módulo Bluetooth H-05 [1]
- Micro servo SG90
- Cables

[1] En los anteriores Sprint hemos trabajado con un módulo WiFi, pero por complicaciones técnicas se ha decidido reemplazarlo por un módulo Bluetooth, que es el que se usará de en adelante

Esquema Hardware

A continuación, se muestra el esquema de conexiones de nuestro robot.

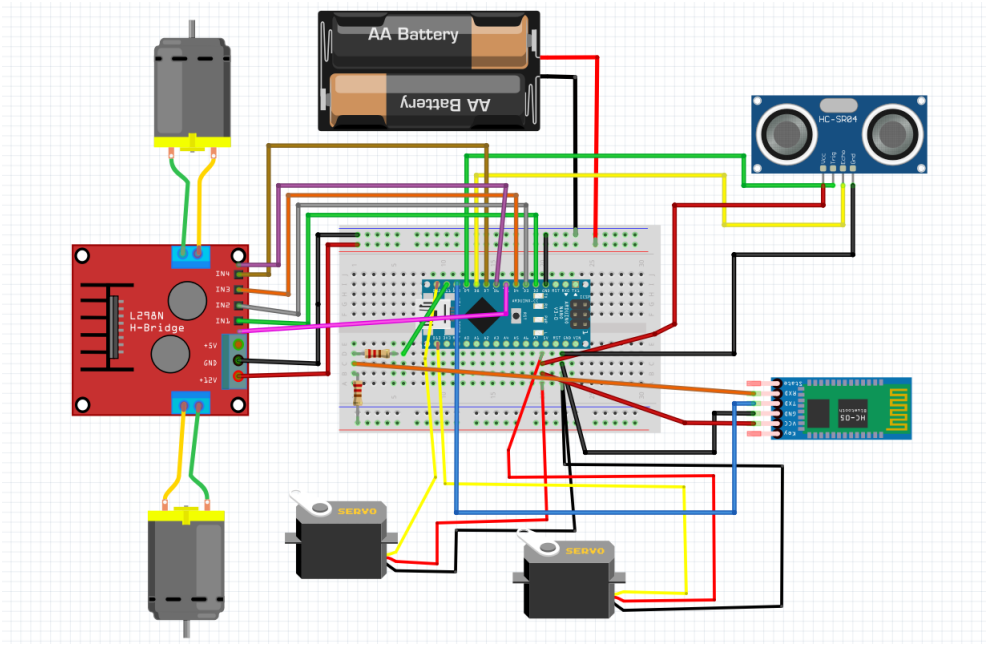

Figure 1: Esquema de connexiones de los componentes.

Arquitectura Software

A continuación, se muestran los distintos módulos software que usaremos y cómo se comunican entre ellos y con los distintos dispositivos.

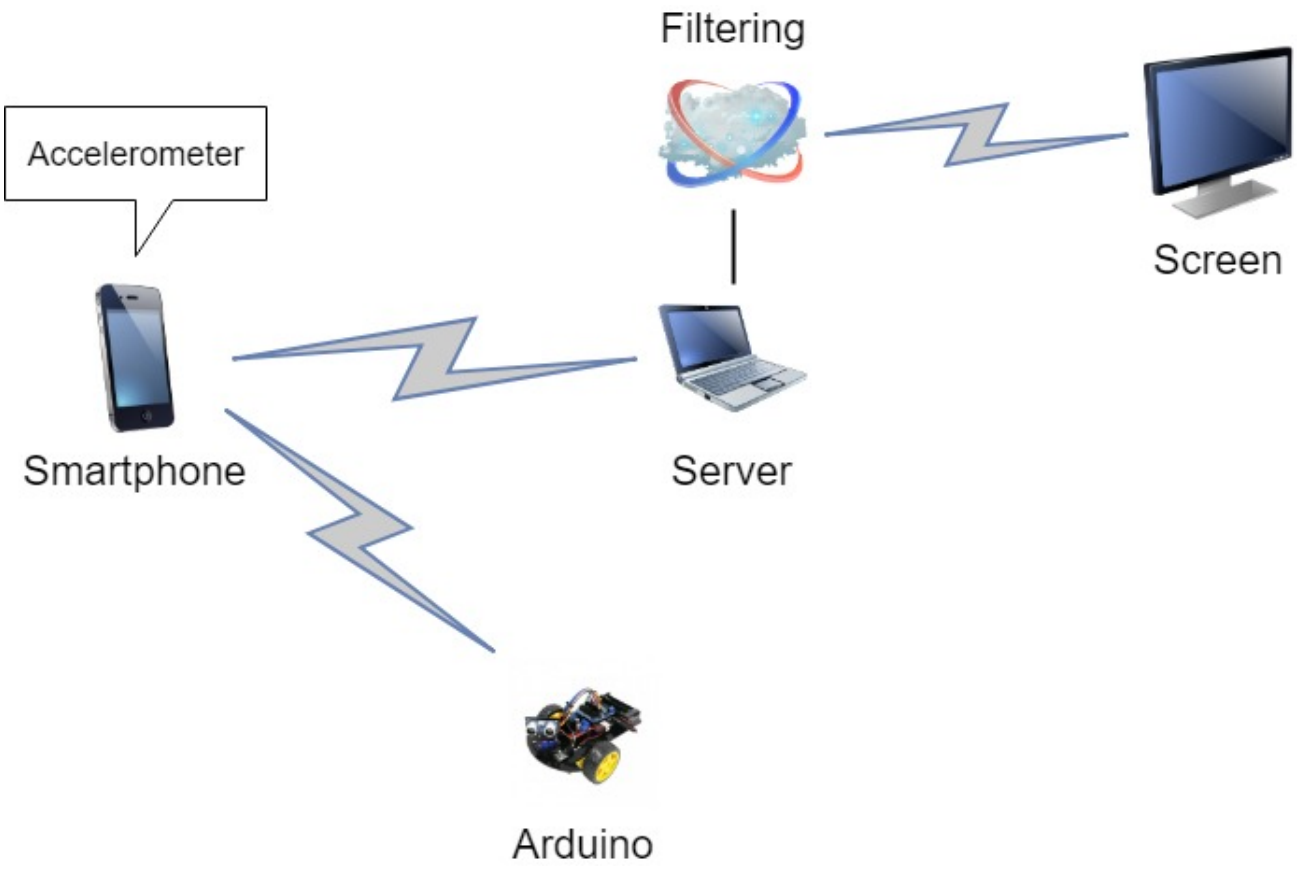

Figure 2: Esquema de módulos software.

Otros Esquemas: sketch del robot

A continuación, se muestra un sketch inicial del concepto de nuestro robot, con las medidas de los componentes.

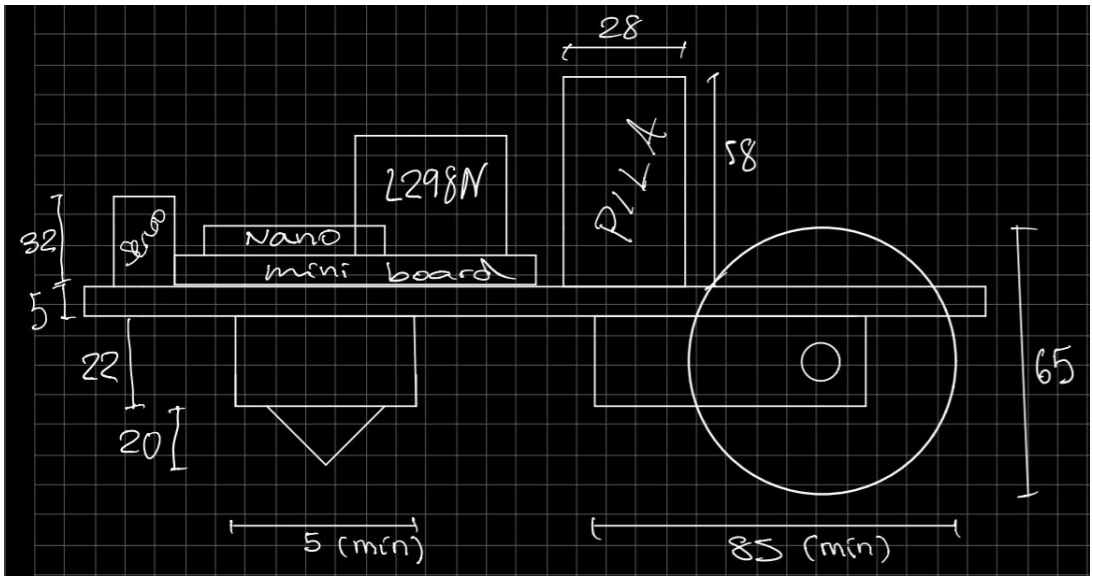

Figure 3: Sketch inicial del robot (cotas en milímetros).

Contribuciones

Como se ha comentado en la descripción del proyecto, DriveArt consiste en un coche robot que permita dibujar, y que se controla mediante los gestos de la cabeza. Está basado no en un sólo proyecto, sinó en una combinación de varios (consultar sección referencias). De modo que nuestra principal contribución ha sido la de unificar todos estos proyectos en uno solo, y al mismo tiempo permitir que el robot dibuje en físico.

Por otra parte, otra contribución notable será la de aplicar un filtrado al dibujo resultante (similar al proyecto "Fishes Can Drive Too", de RLP 2018) y mostrarlo por pantalla. El filtrado puede consistir en eliminación de ruido (especialmente al ser para personas con paraplejía, que pueden realizar algunos movimientos involuntários) y suavizar las líneas del dibujo.

Para el control del robot, hemos decidido comunicar el robot con una aplicación móvil mediante Bluetooth, la cual realizará las siguientes funciones:

- Por una parte, enviará al robot los datos de los gestos realizados con la cabeza, recogidos mediante un accelerómetro interno. Así mismo, también permitirá al usuario seleccionar el color del dibujo (en físico y en pantalla).
- Por otra parte, actuará como lienzo digital donde se irá generando un dibujo. Cuando el usuario haya acabado, la misma aplicación es la que se encargará de aplicar el filtrado descrito.

Finalmente, la nota a la cual aspiramos, cumpliendo con todos los objetivos y funcionalidades que nos hemos planteado, es el 10, porque consideramos que el proyecto es muy completo tanto a nivel hardware como a nivel software, y además se han propuesto varias contribuciones muy significativas, que añaden valor al proyecto, haciendo que destaque y se distinja de los que se pueden encontrar en internet.

También hemos relacionado el proyecto con otra asignatura de nuestra mención (APC, Aprendizaje Computacional), de la cual aprovecharemos algunos conocimientos para realizar algún filtraje tipo "smoothing" o de regresión (ya sea lineal o polinómica). Asimismo, también se aprovecharán conocimientos de otra asignatura fuera de la mención (Web) para desarollar la aplicación móvil.

Componentes adicionales y piezas 3D

- Chasis del coche a medida (rediseñado y adaptado varias veces a lo largo del proyecto) [2]

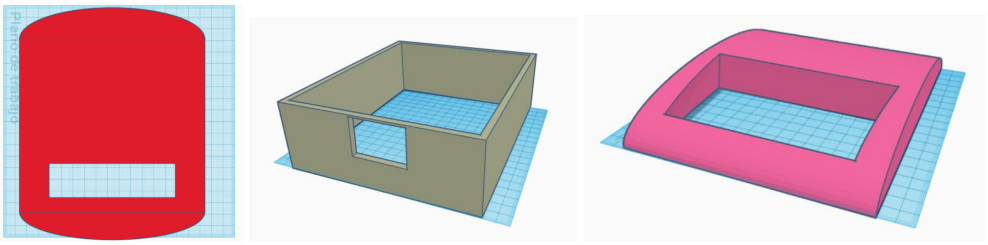

Figure 4: Modelo 3D del chasis del coche robot

- Modelos 3D de los diferentes detalles para tunear el coche (impresos con la impresora 3D que nos ha facilitado un compañero de clase)

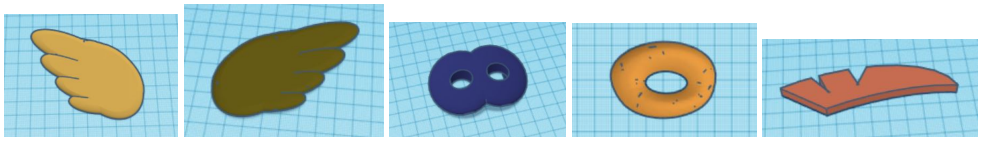

Figure 5: Alas, ojos, boca, y cresta del robot.

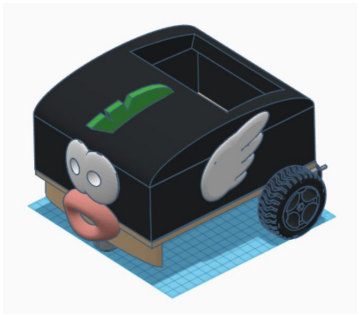

Figure 6: Reconstrucción completa en 3D del coche.

- Estructura de sujeción de los bolis, montada en cartrón
- Rueda loca (comprada)
- Mini breadboard para realizar todas las connexiones

[2] Presentamos un modelo 3D para el chasis, pero debido a que el tiempo requerido para la impresión es muy elevado, hemos decidido substituirlo por un chasis de madera, igualmente firme y resistente

Riesgos previstos y planes de contingencia

A continuación, analizamos los riesgos que se prevé que podrían ocurrir en el proyecto, recogidos en la siguiente tabla:

Riesgo	Descripción	Probabilidad	Impacto	Plan contingencia
1	Desviación del coche provocada por diferente rendimiento entre los dos motores	Media	Alto	Habrá que calibrar los motores previamente, adaptando la velocidad de cada uno para que el coche se mueva en línea recta.
2[3]	Cambio color no funcional (a causa de la transmisión de los engranajes o por falta de potencia del servo)	Baja	Medio	Simular el cambio de color sólo en pantalla y no en físico (rebajando ligeramente la ambición inicial).
3[4]	En caso de que se necesite un pin adicional para RST del módulo Bluetooth, nos faltaria un pin	Media/Alta	Bajo	Eliminar algún componente no esencial, como el sensor de ultrasonidos secundario (el de detrás).
4	Falta de precisión al realizar el dibujo (ya sea por cómo se recogen los gestos con el accelerómetro, o por falta de precisión del robot)	Media	Bajo	Aplicar un filtrado más potente/agresivo para suavizar, aunque el dibujo no quedará tan bien.
5[5]	No poder realizar el filtrado a tiempo real	Media/Alta	Bajo	El dibujo en físico es más visual, de modo que no supone un inconveniente si el filtrado se realiza una vez acabado el dibujo y se puestra por pantalla al final.

Table 1: Tabla de riesgos y plan de contingencia.

[3] Debido a que el sistema de engranajes y ruedas pensado inicialmente era algo inestable, se ha rediseñado varias veces a lo largo del proyecto, finalmente se utiliza un servo independiente para seleccionar cada color.

[4] Para poder conectar el nuevo módulo, ha habido que precindir de uno de los sensores ultrasonido. A cambio, su servo se ha usado para el segundo boli.

[5] Para no relantizar la comunicación y mejorar la interacción con el usuario, se ha decidido realizar el filtrado una vez acabado el dibujo.

Códigos

En este apartado se muestra el código del robot, que ya le permite moverse siguiendo las indicaciones del usuario con los gestos, dibujar en físico, y proyectar una versión del dibujo (sin filtrado).

Para ello, se ha estado desarrollando la aplicación móvil que capta los gestos de la cabeza y se comunica con el robot. El Arduino está constantemente escuchando los datos que le llegan por el Bluetooth, y los interpreta para moverse en la dirección adecuada, para cambiar de color o detenerse.

A continuación el código del Arduino:

```
#include <SoftwareSerial.h>
#include <Servo.h>

// declaraciones de PINs y otras variables.
Servo myservo;
SoftwareSerial BTSerial(10, 11); // RX | TX
char dato; // dato que leemos del bluetooth
int color = 0; //color del dibujo

// motor1
int IN1 = 2;
int IN2 = 8;
int motor1_ena = 5;

// motor2
int IN3 = 4;
int IN4 = 7;
int motor2_ena = 6;

// ultrasonidos
int trigPin = 12;
int echoPin = 13;
long duration, cm, inches;

// inicializaciones
void setup()
{
```

```
  myservo.attach(3);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode( motor1_ena , OUTPUT);
  pinMode( motor2_ena , OUTPUT);
  pinMode (IN1, OUTPUT);
  pinMode (IN2, OUTPUT);
  pinMode (IN4, OUTPUT);
  pinMode (IN3, OUTPUT);
  analogWrite( motor1_ena , 128);
  analogWrite( motor2_ena , 128);
  pinMode(9, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(9, HIGH);
  Serial.begin(9600);
  Serial.println("Ready");
  BTSerial.begin(38400); // comunicacion bluetooth
}

// funcion que detiene el robot
void Mover_Stop()
{
  digitalWrite( IN1, LOW);
  digitalWrite( IN2, LOW );
  digitalWrite( IN3, LOW);
  digitalWrite( IN4, LOW );
}

// codigo principal
void loop()
{
  if (BTSerial.available())
  {
    dato = BTSerial.read(); // leemos dato bluetooth
    Serial.write(dato);
    if(dato=='1'){ // giro derecha
```

```
  digitalWrite( IN1, HIGH);
  digitalWrite( IN2, LOW );
  digitalWrite( IN3, HIGH);
  digitalWrite( IN4, LOW );
}
if(dato=='2'){ // giro izquierda
  digitalWrite( IN1, LOW);
  digitalWrite( IN2, HIGH);
  digitalWrite( IN3, LOW);
  digitalWrite( IN4, HIGH);
}
if(dato=='3'){ // marcha atras
  digitalWrite( IN1, HIGH);
  digitalWrite( IN2, LOW );
  digitalWrite( IN3, LOW);
  digitalWrite( IN4, HIGH );
}
if(dato=='4'){ // adelante
  // leer ultrasonidos
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = (duration/2) / 29.1;
  Serial.write(cm);

  // comprueba distancia de seguridad para evitar que choque
  if (cm>10){
    digitalWrite( IN1, LOW);
    digitalWrite( IN2, HIGH );

    digitalWrite( IN3, HIGH);
    digitalWrite( IN4, LOW );
```

```
      }
    }
    if(dato=='5'){ // seleccion color1
      myservo.write(100);
    }
    if(dato=='6'){ // seleccion color2
      myservo.write(40);
    }
  }
  delay(300);
  Mover_Stop(); // detener coche
}
```

Aplicación

Paralelamente al coche-robot, se ha estado desarrollando la aplicación móvil en Angular/Ionic3, la cual se comunica con el robot mediante Bluetooth. Debido a su larga extensión, el código de la aplicación no se incluye en este report, pero se puede consultar online en el siguiente GitHub:

`https://github.com/marinarierav/DriveArt`

Ionic es una tecnología que permite diseñar aplicaciones móbil híbridas y que se pueden llevar a cualquier plataforma (Android, iOS, Web), que se programan utilizando Typescript (sobre JavaScript), HTML5 y CSS. A continuación el enlace a la documentación:

`https://ionicframework.com/docs/v3/native/`

La aplicación permite, por un lado, el control del robot para el dibujo físico, capturando los movimientos que el usuario realiza con la cabeza, y pasándoselos al Arduino una vez tratados para que actúe conforme. También incorpora diversos botones para la selección del color que usa el robot para dibujar.

Por otro lado, proyecta el dibujo realizado en un lienzo digital, que se va completando a tiempo real. Una vez acabado el dibujo, permite también aplicar un filtrado (como se verá mejor en la siguiente sección del informe).

Para ello, se han utilizado, principalmente, las siguientes extensiones de cordova (además de las básicas que se encontrarían en cualquier aplicación):

- **Bluetooth Serial** - Para permitir la comunicación via Bluetooth.
  (`https://github.com/don/BluetoothSerial`)
- **Device Motion** - Para capturar los datos del accelerómetro del móvil, que sirven para recoger los gestos realizados con la cabeza.
  (`https://github.com/apache/cordova-plugin-device-motion`)

A continuación se muetran dos pantallazos del funcionamiento de la aplicación:

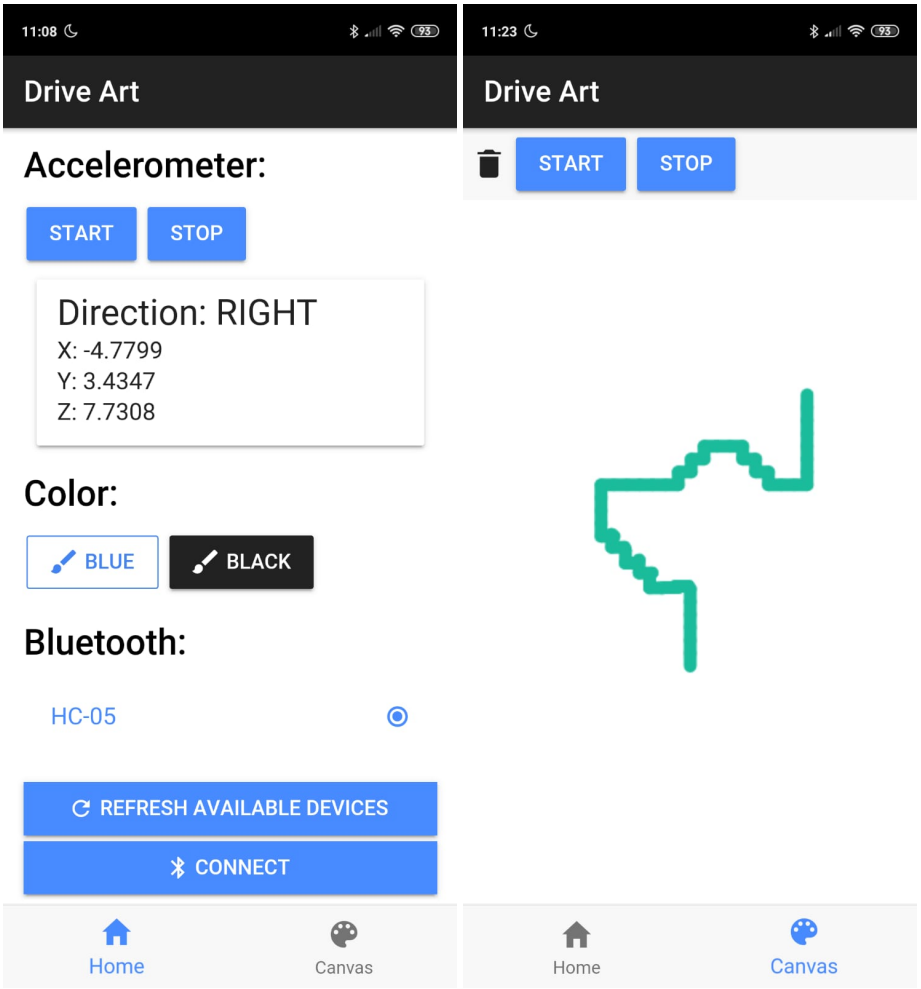

Figure 7: A la izquierda, pantalla principal de la aplicación, donde se muestran los datos del acelerómetro y como se traducen a una dirección, la opción para cambiar de color, y la parte de la configuración del Bluetooth (a nivel de usuario). A la derecha, pantalla del lienzo, donde se realiza el dibujo digital a medida que el robot se mueve.

Filtering

En este apartada se analizan y comparan las diferentes técnicas que se han probado para hacer el filtrado del dibujo final acabado.

Curvas cuadráticas

Este tipo de filtrado utiliza curvas cuadráticas, las cuales se definen usando 3 puntos: inicio y final, más un punto de control que define la curvatura de la curva.

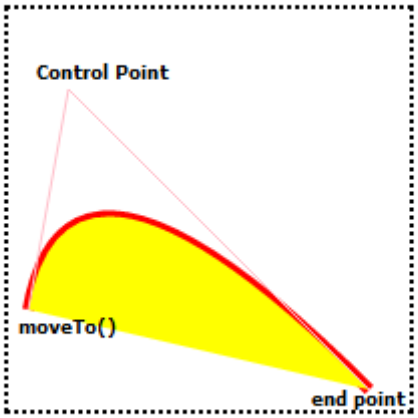

Figure 8: Curva cuadrática.

Sintaxi:

```
moveTo(startX, startY);
quadraticCurveTo(cx, cy, endX, endY);
```

Donde:

- (endX, endY) son las coordenadas del punto de inicio de la curva.
- (cx, cy) son las coordenadas del punto de control.
- (endX, endY) son las coordenadas del punto final de la curva.

Curvas de Bezier

Otro tipo de filtrado que se ha implementado es el que usa curvas de Bezier. Este tipo de curvas son parecidas a las cuadráticas, pero en vez de utilizar un sólo punto de control, usan dos puntos de control, por lo que en algunos casos, permiten definir mejor la curvatura de la curva.

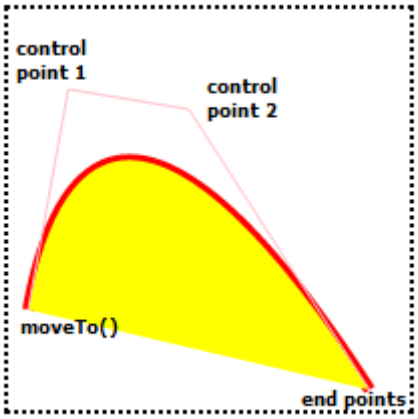

Figure 9: Curva de Bezier.

Sintaxi:

```
moveTo(startX, startY);
bezierCurveTo(cx1, cy1, cx2, cy2, endX, endY);
```

Donde:

- (endX, endY) son las coordenadas del punto de inicio de la curva.
- (cx1, cy1) son las coordenadas del primer punto de control.
- (cx2, cy2) son las coordenadas del segundo punto de control.
- (endX, endY) son las coordenadas del punto final de la curva.

Resultados

A continuación se muestran unas imágenes de ejemplo de un dibujo proyectado a la aplicación, antes y tras el filtrado. Se ha aplicado un filtrado que utiliza las curvas de Bezier, porque de los diversos métodos analizados, es el tipo de filtrado que ha dado unos mejores resultados. Se puede ver como permite suavizar los trazados del dibujo, pero sin llegar a perder la definición de los contornos.

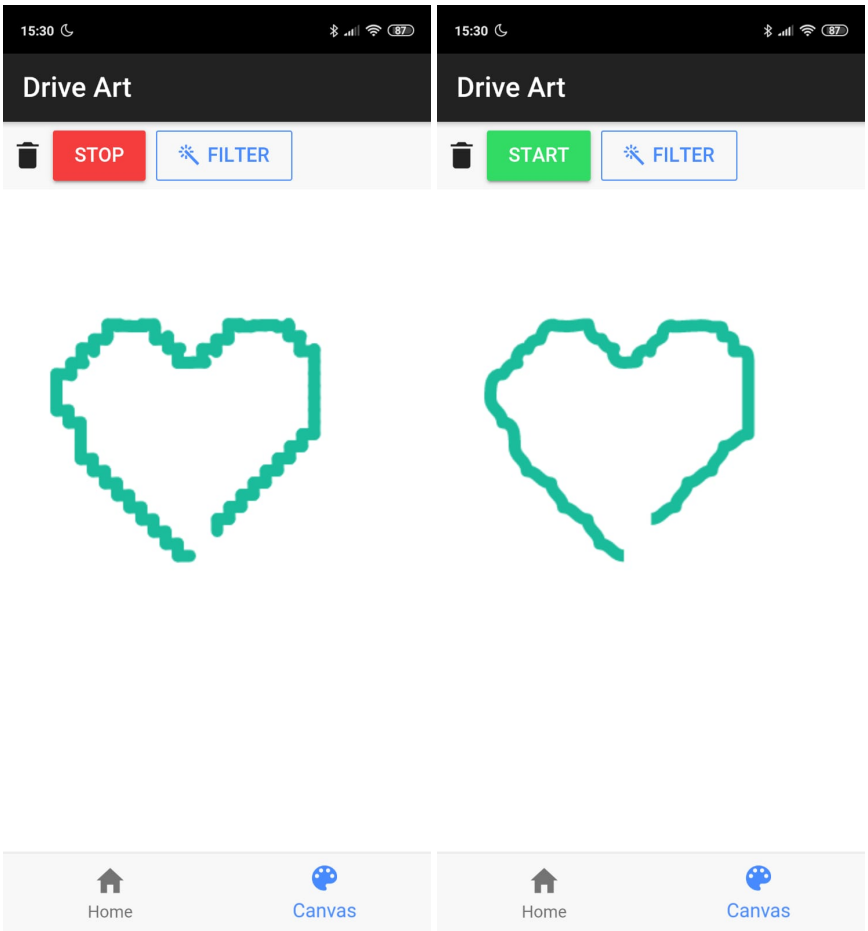

Figure 10: Comparación de los dibujos sin filtrar (izquierda) y con filtrado de Bezier (derecha).

Sprint #4

El Sprint 4 ha consistido en seguir puliendo aquellos detalles que nos habían quedado pendientes para finalizar el prototipo final.

Por lo referente al robot, se ha implementado un nuevo sistema de cambio de color, de manera que cada boli se controla con un botón de la aplicación, que acciona un servo independiente para cada boli. Además, se han acabado de imprimir todas las piezas 3D para tunear el robot y darle un aspecto más amistoso:

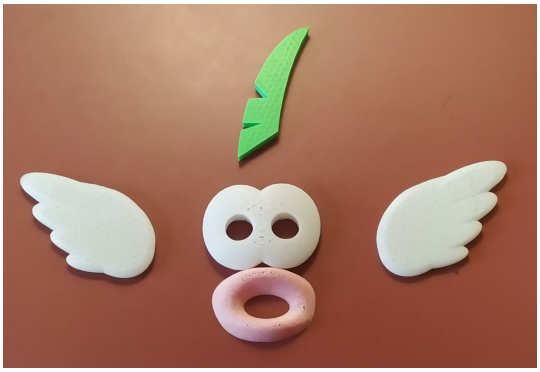

Figure 11: Piezas 3D ya impresas.

Por otra parte, se ha dedicado parte del tiempo a intentar mejorar la interacción usuario-robot, ya que el prototipo del anterior Sprint era un poco limitado en este aspecto, el delay que había entre los movimientos del robot hacía que el proyecto en general mancara de interactividad y jugabilidad. Esto suponía un problema, sobretodo recordando las aspiraciones iniciales del proyecto, y teniendo en cuenta que el público al que se destina son personas con discapacidad, que deben poder disfrutarlo y manejarlo de forma intuitiva.

Debido a las limitaciones hardware (tiempo de reacción del Arduino, falta de precisión de los motores, ...) y software (tiempo de lectura del accelerómetro, delay entre envío y lectura de estos datos, saturación del buffer del serial Bluetooth, ...) no ha sido sencillo poder mejorar este aspecto de interactividad, y se han evaluado diferentes posibilidades para afrontar el problema:

- Constantemente vaciar el buffer para que no se llene (cuando hay overflow, el bluetooth se desconecta y por lo tanto es inadmisible), cosa que incrementa aún más el tiempo de respuesta del robot.

- Augmentar un poco el tiempo que se le deja al robot para actuar, mandándole menos datos para que no se sature.

Al final se ha optado por la segunda opción, ya que es mucho más rápida (aunque tiene mayor riesgo), y de este modo se espera haber podido incrementar la interactividad del robot.

El entregable que presentamos en este Sprint ya cumple todas las funcionalidades y objetivos que el grupo ha establecido que el robot debe cumplir:

- Permite la comunicación de la aplicación con el Arduino mediante Bluetooth.
- Recoge los gestos realizados con el acelerómetro, los traduce a direcciones y se las pasa al robot.
- Permite que el robot realice un dibujo en físico.
- Permite el cambio de color de los bolis.
- Realiza también el dibujo en un lienzo digital.
- Permite también aplicar un filtrado sobre el dibujo final acabado para eliminar ruído y suavizar los contornos.

A continuación se muestra una demo del funcionamiento del robot:

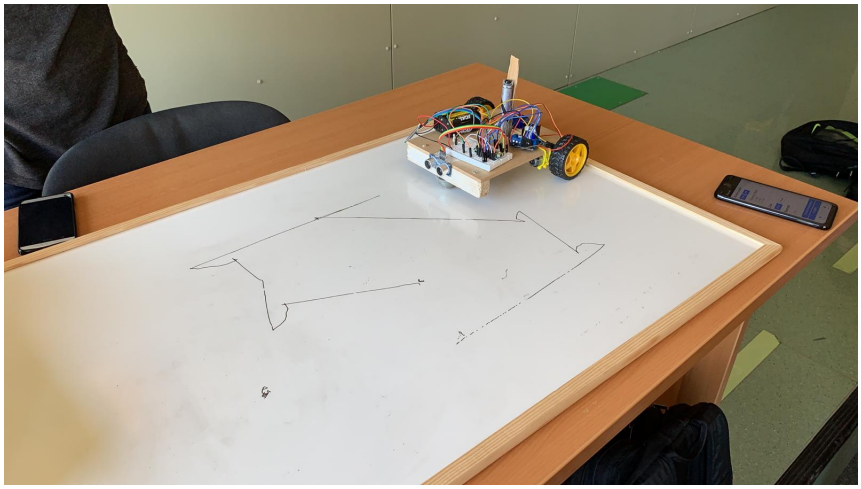

Figure 12: Demo realizada el dia 15 de mayo (en clase de teoría), donde se demostró el funcionamiento del coche, controlado via nuestra app, para realizar un pequeño dibujo sobre una pizarra de Vileda.

Se dedicará estos últimos dias a terminar la documentación, pulir detalles de la aplicación, y a tunear la maqueta final del coche-robot.

Referencias

El proyecto ha estado inspirado, principalmente, como una combinación de los siguientes proyectos de internet:

1. Human Head to Robot Head: https://create.arduino.cc/projecthub/jegatheesan/human-head-to-robot-head-364bfd?ref=\search&ref_id=head%20control&offset=5

2. Obstacle Avoiding Robot: https://www.instructables.com/id/Arduino-Ultimate-Obstacle-Avoiding-Robot/

3. CNC Drawing Arm: https://www.instructables.com/id/CNC-Drawing-Arm/

4. Fishes Can Drive Too: https://rlpengineeringschooluab2018.wordpress.com/2018/05/29/fishes-can-drive-too/