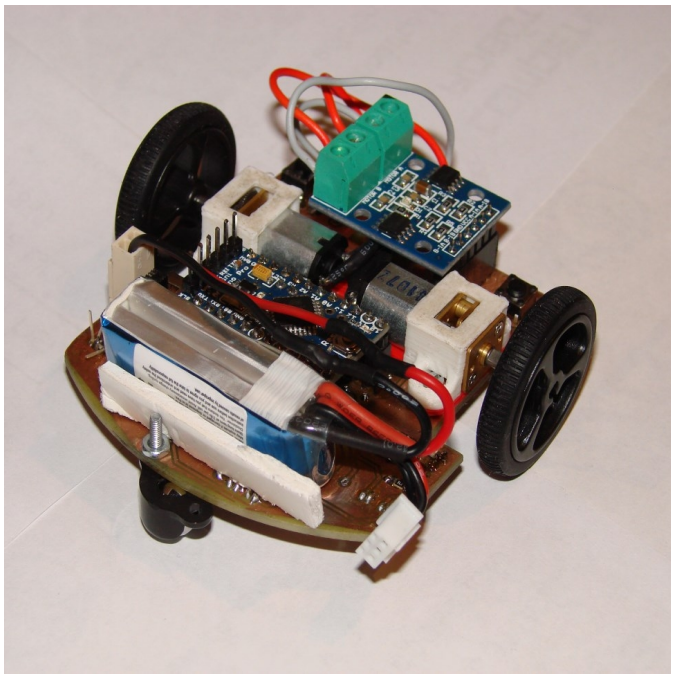
Competencia de robótica

JOHNNY 5

~2015~

Categoría: **Velocista**

Nombre robot: **Arión**



Institución: **Club de Robótica – Facultad de Ingeniería- U.B.A.**

Participantes: (Nombre completo y mail)

- **Martín Mello Teggia – martinmt139@gmail.com**
- **Damián Cudero - cudero@yahoo.com.ar**
- **Mateo Aragón - mateoaragon@gmail.com**
- **Nicolás Matsunaga - nicomatsu@gmail.com**

- **Gustavo Crescentini - guscres@gmail.com**
- **Joaquín Ulloa - joaquin.g.ulloa@gmail.com**

1. Introducción

1. Objetivos

En la mitología griega **Arión** (en griego antiguo Ἀρείων *Areíôn*, 'mejor', 'más fuerte', 'más valeroso') era un fabuloso caballo de pezuñas negras que poseía el don de la palabra y la inmortalidad.

Este nombre fue elegido para nuestro velocista para representar el resultado de mucho tiempo de trabajo, resultando en el mejor y mas rápido velocista hecho hasta el momento en el club de robótica.

2. Modelos de referencia

Para el presente robot nos inspiramos en las competencias de *micromouse*[1], originarias de Japón, en particular en el modelo conocido como Green Giant [2], en el cual se busca conseguir el menor peso y consumo posible, resultando en una menor inercia, y mejor respuesta del robot. La parte negativa de esto, es la baja autonomía, ya que el robot puede circular durante media hora antes que se agote completamente su batería.

3. Historia de desarrollo en el club

Existen numerosos velocistas realizados en el club de robótica a la largo de los años. Con cada uno de estos vehículos se fue aprendiendo distintas cosas, transmitidas luego a los nuevos integrantes del club, así como también a toda la comunidad.

Todo el trabajo realizado en el club es abierto, desde los diseños de los circuitos impresos, hasta los diseños 3D en los proyectos que los usan, y la documentación de los mismos. La idea es aprovechar el conocimiento obtenido previamente, y no reinventar la rueda para cada nuevo proyecto. Una lista completa de los proyectos del club puede encontrarse en el gestor de proyectos del mismo[3], en el cual puede verse la progresión de los distintos proyectos a lo largo del tiempo.

4. Robótica como ciencia interdisciplinaria

En todos los proyectos que desarrollamos de robótica, podemos distinguir aspectos de distintas áreas de conocimiento. En particular vamos a enfocar la siguiente documentación en 3 aspectos: La mecánica, la electrónica y la programación.

La condición de interdisciplinaria de la robótica hace que en el club de robótica participe gente de distintas áreas. En el caso particular de este proyecto, el grupo esta conformado por 2 estudiantes de ingeniería electrónica, 2 estudiantes de ingeniería mecánica, 1 programador, y un ingeniero electrónico. Además de esto, se cuenta con el apoyo de los demás miembros del club, que aportan su conocimiento a medida que el mismo es requerido.

Sin mas preámbulos, vamos a realizar un repaso a través de las principales partes del robot.

2. **Mecánica**

1. Objetivo

El presente robot se planteó desde el inicio con ciertos objetivos referidos a sus especificaciones. El primero de los mismos fue el peso, con un valor objetivo de 80 g, considerando hasta 100 g un valor aceptable. Desde el punto de vista de la velocidad máxima lineal alcanzable, se tomaron como referencia valores de competencias pasadas, y se optó como objetivo conseguir una velocidad de 1m/s, quedando sujeto a la disponibilidad de elementos para poder conseguir esto. Por último, para las dimensiones, se definió usar como soporte la misma placa del circuito impreso a fin de reducir el peso de cualquier material agregado, y darle la rigidez necesaria a la estructura.

2. Peso

Para poder cumplir con el peso máximo aceptable, el primer aspecto que debíamos modificar era la batería utilizada. Esto es porque las baterías recargables disponibles en el club son de 2700 mAh, pero pesan 100 g cada una. En el mercado local[4] pudimos comprar 2 baterías de 300 mAh, con un peso de 17 g cada una.

La segunda decisión para reducir el peso fue diseñar soportes para los motores. Los mismos fueron impresos en la impresora 3D del club, ofreciendo una opción liviana para afirmar los motores.

Para la elección de los motores[5], se eligieron los *microgearmotor* de Pololu. Esto fue por ser los mismos livianos, así como también son favorables su reducido volumen y la caja de reducciones metálicas alcanzando un peso total de 9.7 g cada uno.

Al elegir la electrónica, Arduino Pro Mini[6] y el módulo de puente H[7], también se tuvo en cuenta este factor (ambos componentes suman un peso de 12 g), aunque se explicarán en mayor detalle más adelante.

El robot terminó con un peso final de 88.4 g totalmente armado, por lo que consideramos que se logró el objetivo de manera satisfactoria.

3. Velocidad

Al momento de elegir tanto los motores como las ruedas, la disponibilidad de los mismo jugó un rol fundamental (la relación entre estos es lo que define la velocidad del robot). Se optó por motores de potencia media (800 mA max), con caja de reducción de 30:1 metálica, y 730 revoluciones por minuto. El agregado de ruedas de 32 mm de diámetro dieron finalmente una velocidad máxima de 1.22 m/s.

Si bien se consideró la utilización de motores de alta potencia, se tuvo que pasar a la opción más económica en potencia, por la limitación de energía presentada por las baterías conseguidas.

3. Electrónica

1. Circuito impreso

El diseño del circuito impreso se planteó para cumplir 2 funciones de manera simultánea: controlar el comportamiento del robot, y servir de estructura base para el autito. Con este objetivo en mente, y la idea de poder realizar distintas placas con diferentes configuraciones de los componentes, se optó por utilizar módulos independientes que pudieran ser incorporados al robot, de forma tal que al migrar a otra placa, no se pierdan los componentes valiosos del robot.

La placa está conformada por pines hembra para enchufar el microcontrolador y el puente H, tiene 3 LEDs para permitirnos extraer información mientras el auto está funcionando, 3 botones para modificar el flujo del código, 5 sensores, y las resistencias necesarias para todos los componentes. También están contemplados en la placa la ubicación de los motores, así como también los soportes para los mismos, impresos en 3D.

2. Control central

Como módulo de control programable, se eligió el **Arduino Pro Mini**. El mismo está basado en un microcontrolador de la empresa Atmel (Atmega328), funciona con 5 V y tiene una velocidad de clock de 16 MHz. Como tiene un regulador de tensión incorporado, se puede alimentar con tensiones de hasta 12 V, permitiendo conectarlo directamente a la batería sin regular la tensión de la misma.

El regulador incorporado del Arduino es utilizado en la placa como referencia para los receptores de los sensores, ya que la obtención de datos de los mismos se realiza mediante los conversores A/D, y es importante no superar el valor máximo de la referencia del controlador.

3. Control de motores y sensores

Para manejar los motores se utilizó un módulo de Arduino, basado en dos puentes H **HG7881CP**. Éste puede entregar una corriente de hasta 800 mA por canal, y es alimentado con la tensión de la batería.

Para sensar la pista, se utilizaron 5 sensores infrarrojos orientados hacia la pista. Se eligieron los **TCRT1000**[8] por su reducido tamaño. Los mismos están conformados por un LED infrarrojo y un fototransistor.

El uso de los sensores es sencillo de explicar: Se hace incidir un haz de luz infrarroja sobre la pista con el LED, y se mira cuanta luz vuelve con el fototransistor. La cantidad de luz que vuelve al sensor depende del color de pista, ya que el color blanco refleja una mayor componente que el negro. La misma es capturada mediante el Arduino, y con esto se puede saber qué tan blanca es la pista bajo el sensor. Al incorporar la información de los 5 sensores, se cuenta con una buena cantidad de información sobre la posición del robot.

4. Baterías

Las **baterías** utilizadas están formadas por 2 celdas de Li-Po, alcanzando una tensión máxima de carga de 8.4 V, y una tensión mínima (recomendable) de 7.4 V. Se cuenta con 2 unidades de la misma, que al tener 5 C de carga, pueden cargarse completamente en tan sólo 12 minutos (actualmente son cargadas a 500 mA).

Los cargadores de Li-Po sólo son capaces de cargar baterías que no se hayan descargado por debajo de 3.7 V por celda, razón por la cual se implementó en el robot un mecanismo para controlar la carga de las mismas, e informar mediante una codificación de LEDs cuando la tensión de la batería se acerca a un punto crítico, para poder ser reemplazada y cargada.

4. Programación

1. Lenguaje de programación

Como usamos la plataforma Arduino, el lenguaje de programación utilizado es C++, que dispone de algunas mejoras por sobre el lenguaje C que suele utilizarse en sistemas embebidos, a costa de una mayor complejidad. Afortunadamente, las bibliotecas de Arduino permiten disponer de todo el potencial de C++ sin mayores dificultades.

2. Facilidades de Arduino

El uso de un Arduino facilitó enormemente la realización de cosas como el movimiento de los motores mediante un PWM, ya que incluye bibliotecas específicas que permiten realizar este tipo de funciones de manera nativa sin necesidad de resolverlas para el hardware particular con que se cuenta.

Otro punto en el cual fue favorable el uso del Arduino, fue a la hora de utilizar los conversores A/D, para los cuales existe una sintaxis específica que marca los puertos como lectura, y realiza las conversiones de manera autónoma.

3. Sistema de control

Se eligió utilizar un sistema de control PID para el robot, por la velocidad de corrección que ofrece el mismo frente a una máquina de estados.

La información de los sensores se junta toda en una variable que promedia no sólo el valor que mide cada sensor, sino su posición sobre la línea relativa al centro del robot. Con este valor se puede saber qué tan alejado se encuentra el robot del centro de la línea.

La componente proporcional del sistema de control (P) se calcula con la distancia al centro de la línea del robot.

La componente integral del sistema (I) suma los errores del robot de cada lado de la línea, de forma tal de compensar si un motor funciona más lento que el otro a lo largo del tiempo.

La componente derivada del PID es la encargada de evitar los cambios bruscos en el robot, actuando de manera puntual con mucha intensidad, lo contrario del I, que actúa con mayor lentitud.

4. Prácticas de programación

Todo el código se realizó de forma parametrizada, de manera tal que si otro proyecto decidiera implementar el mismo sistema de control, solo es necesario cambiar los pines sobre los que esta referenciado el código, y el mismo lazo de control debería funcionar sin ningún otro agregado.

5. **Bibliografía**

[1] https://en.wikipedia.org/wiki/Micromouse
[2] http://greenye.net/Pages/Micromouse/Micromouse2014-2015.htm
[3] http://labi.fi.uba.ar/chiliproject
[4] www.helitec.com.ar
[5] https://www.pololu.com/product/992
[6] https://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardProMini
[7] http://chioszrobots.com/2014/06/01/hg7881-h-bridge-stepper-motor-dual-dc-motor-driver-controller/
[8] http://www.farnell.com/datasheets/1915192.pdf

6. **Código**

```
// definición de pines del micro.
const int pwmMotorD = 11;
const int pwmMotorI = 10;
const int sentidoMotorD = 3;
const int sentidoMotorI = 5;
const int ledArduino = 13;
const int led1 = 9;
const int led2 = 8;
const int led3 = 12;
const int boton1 = 4;
const int boton2 = 6;
const int boton3 = 7;
const int sensor0 = A0;
const int sensor1 = A1;
const int sensor2 = A2;
const int sensor3 = A3;
const int sensor4 = A4;
const int batteryControl = A5;

const int cantidadDeSensores = 5;
int sensores[cantidadDeSensores];

// indices de array sensores
const int izq    = 0;
const int cenIzq = 1;
const int cen    = 2;
const int cenDer = 3;
const int der    = 4;

const int tolerancia = 5; // Margen de ruido al medir negro.
const int toleranciaBorde = 50; // Mínimo para decidir cuál fue el último borde

// velocidadMinima + rangoVelocidad <= 255
const int velocidadMinima = 0;
const int rangoVelocidad = 255;
int reduccionVelocidad;
int errP;
int errPAnterior;
int errI;
int errD;

int sensoresLinea = 0;
const int centroDeLinea = 2000;
const int coeficienteErrorP = 12;
const int coeficienteErrorI = 6000;
const int coeficienteErrorD = 3;

boolean estadoActualAdentro; // determina si se usa modo PID o modo "me fui"
// bordes para modo "me fui"
const int derecha = 1;
const int izquierda = 0;
int ultimoBorde;

void setup() {
  // como los motores se manejan con AnalogWrite,
  // no hace falta ponerlos como salida
  // pinMode(pwmMotorD, OUTPUT);
  // pinMode(pwmMotorI, OUTPUT);
```

```
  pinMode(sentidoMotorD, OUTPUT);
  pinMode(sentidoMotorI, OUTPUT);

  // como los sensores y el batteryControl se leen con AnalogRead,
  // no hace falta ponerlos como entrada
  // pinMode(sensor0, INPUT);
  // pinMode(sensor1, INPUT);
  // pinMode(sensor2, INPUT);
  // pinMode(sensor3, INPUT);
  // pinMode(sensor4, INPUT);
  // pinMode(batteryControl, INPUT);

  pinMode(ledArduino, OUTPUT);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);

  pinMode(boton1, INPUT);
  pinMode(boton2, INPUT);
  pinMode(boton3, INPUT);

  for (int i = 0; i < cantidadDeSensores; i++) {
    sensores[i] = 0;
  }

  apagarMotores();

  errP = 0;
  errI = 0;
  errD = 0;
  errPAnterior = 0;
  ultimoBorde = izquierda;
  estadoActualAdentro = true;

}

boolean apretado(int boton) {
  return (digitalRead(boton) == LOW);
}

void esperarReboteBoton() {
  delay(5);
}

inline void obtenerSensores() {
  // carga en el array de sensores las lecturas AD de cada sensor
  // este proceso lleva 500 us
  sensores[izq]    = analogRead(sensor0);
  sensores[cenIzq] = analogRead(sensor1);
  sensores[cen]    = analogRead(sensor2);
  sensores[cenDer] = analogRead(sensor3);
  sensores[der]    = analogRead(sensor4);
}

void mostrarSensor(int sensor) {
  if ((sensor >= cantidadDeSensores) || (sensor < 0)) {
    return;
  }
  digitalWrite(led1, ((sensores[sensor] / 256) ? HIGH : LOW));
  digitalWrite(led2, ((sensores[sensor] / 64) ? HIGH : LOW));
  digitalWrite(led3, ((sensores[sensor] / 16) ? HIGH : LOW));
```

```
}

void apagarMotores() {
  analogWrite(pwmMotorD, 0);
  analogWrite(pwmMotorI, 0);
}

void loop() {

  // hasta que se presione el botón, espera,
  // y muestra en los leds el valor del sensor central
  while (!apretado(boton2)) {
    obtenerSensores();
    mostrarSensor(cen);
  }
  esperarReboteBoton();

  // inicialización de todo
  setup();

  // hasta que se suelte el botón, espera
  while (apretado(boton2));
  esperarReboteBoton();

  // arranque progresivo, de 0 a 250 en 75 ms
  for (int i = 1; i <= 25; i++) {
    analogWrite(pwmMotorD, i * 10);
    analogWrite(pwmMotorI, i * 10);
    delay(3);
  }

  // ejecuta el ciclo principal hasta que se presione el botón
  while (!apretado(boton2)) {
    obtenerSensores();

    if (sensores[izq] > toleranciaBorde) {
      ultimoBorde = izquierda;
    } else if (sensores[der] > toleranciaBorde) {
      ultimoBorde = derecha;
    }

    // si me fui, entro en modo "corrección máxima"
    if ((sensores[izq]    < tolerancia) &&
        (sensores[cenIzq] < tolerancia) &&
        (sensores[cen]    < tolerancia) &&
        (sensores[cenDer] < tolerancia) &&
        (sensores[der]    < tolerancia)) {
      estadoActualAdentro = false;
    } else {
      estadoActualAdentro = true;
    }

    if (estadoActualAdentro) {
      // modo pid
      // linea = (0 * s0 + 1000 * s1 + 2000 * s2 + 3000 * s3 + 4000 * s4) / (s0 + s1 + s2 + s3 + s4)
      // 0 a 4000, donde 2000 es el centroDeLinea
      sensoresLinea = (
        sensores[izq]    * 0 +
        sensores[cenIzq] * 1000 +
        sensores[cen]    * 2000 +
        sensores[cenDer] * 3000 +
        sensores[der]    * 4000
```

```
      ) / (
        sensores[izq]    +
        sensores[cenIzq] +
        sensores[cen]    +
        sensores[cenDer] +
        sensores[der]
      );

      errP = sensoresLinea - centroDeLinea;
      errI = errP;
      errD = (errP - errPAnterior);
      errPAnterior = errP;

      reduccionVelocidad = errP * ( 1 / coeficienteErrorP) + errI * (1 / coeficienteErrorI) + errD *
(1 / coeficienteErrorD);

      reduccionVelocidad = constrain(reduccionVelocidad, -rangoVelocidad, rangoVelocidad);

      if (reduccionVelocidad < 0) {
        // a la derecha de la linea
        analogWrite(pwmMotorI, velocidadMinima + rangoVelocidad - reduccionVelocidad);
        analogWrite(pwmMotorD, velocidadMinima + rangoVelocidad);
      } else {
        // a la izquierda de la linea
        analogWrite(pwmMotorI, velocidadMinima + rangoVelocidad);
        analogWrite(pwmMotorD, velocidadMinima + rangoVelocidad - reduccionVelocidad);
      }

    } else {
      // modo me fui
      if (ultimoBorde == izquierda) {
        analogWrite(pwmMotorI, 0);
        analogWrite(pwmMotorD, 20);
      } else if (ultimoBorde == derecha) {
        analogWrite(pwmMotorI, 20);
        analogWrite(pwmMotorD, 0);
      }
    }
  }
  esperarReboteBoton();

  // inmediatamente después de presionar el botón para salir del ciclo,
  // se apagan los motores
  apagarMotores();

  // hasta que se suelte el botón, espera
  while (apretado(boton2));
  esperarReboteBoton();

}
```